7

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLAN"

DESIGNER EN EL MODELADO DE SISTEMAS

T E S I N A

QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

JULIO CESAR CRUZ ESPINAL

ASESOR: ING. RUBEN ROMERO RUIZ

ENERO 2002

PAGINACION DISCONTINUA

AGRADECIMIENTOS.

Doy gracias a mis padres por su incondicional y desinteresado apoyo, por su paciencia y su alentador carácter. A mi familia que siempre ha confiado en mi.

A mi esposa por su paciencia y su apoyo, a pesar del tiempo que le quite para dedicárselo al presente trabajo, Te Amo.

Ingeniero Rubén Romero Ruiz, mi asesor en ésta tesina, por su colaboración para hacer posible la misma.

A mis maestros de la Honorable ENEP ACATLAN, porque sin ellos no hubiera sido posible este trabajo.

Finalmente, agradezco el apoyo recibido por parte de Mario Santiago, Gerente en MSG Consultoría en Sistemas de Información, quien contribuyó a la instalación y configuración de las herramientas utilizadas. Aun cuando estos temas no son el objetivo del trabajo, fueron la base para poder realizarlo.

CONTENIDO

OBJETIVO

Analizar las técnicas de modelado de sistemas empleando Designer/2000 como herramienta CASE y aplicando un caso práctico.

INTRODUCCION

Hoy en día, debido al alto grado de competitividad, las empresas requieren de sistemas de información cada vez más eficientes, que les permitan soportar su operación diaria para optimizar tiempos y costos. Así mismo, las necesidades de servicios de informática van en aumento, al igual que la eficiencia y calidad exigidas en cada uno de ellos.
Dentro del campo del desarrollo de sistemas de información, es importante seguir una metodología que apoye el logro de los objetivos de un proyecto y en consecuencia asegurar el éxito. Realizar cada actividad siguiendo una secuencia definida, junto con tareas de monitoreo y control, permite alcanzar las metas esperadas en menos tiempo y a menor costo. Hacer las cosas sin el uso y guía de una metodología es encaminarse directamente al caos, a la perdida de control y al aumento innecesario de las inversiones; se debe involucrar, más que buenos deseos, la habilidad de la comunicación y administración de la metodología. Esto requiere la aplicación de técnicas especificas y herramientas en el ciclo de vida de los sistemas, a fin de asegurar un eficiente mantenimiento de los sistemas una vez que se encuentren en producción.

En el área de sistemas, las siglas CASE (Computer Aided Software Engineering) se utilizan para hacer referencia a la ingeniería de sistemas asistida por computadora.
La herramienta CASE de Oracle denominada Designer/2000, tiene como objetivo dar soporte a todo el ciclo de vida de los sistemas de información, permitiendo la construcción en forma automatizada.
Designer/2000 forma parte de una familia de productos orientados a la creación de aplicaciones Cliente/Servidor, escalables, portables y fáciles de desarrollar tanto al nivel de grupos de trabajo, como al nivel departamental o corporativo, de la forma más productiva posible. Soporta diseños complejos con Reingeniería de Procesos de Negocios (BPR) y a partir de esto generar sistemas orientados al manejo de información almacenada en Bases de Datos (BD) de cualquier proporción. Cuenta con un repositorio de información común, un ambiente de desarrollo cliente/servidor unificado y una gran cantidad de herramientas gráficas para el diseño de modelos y generación de grandes aplicaciones.

El propósito de esta tesina es analizar el *Modelo Entidad Relación* y el *Modelo Jerárquico Funcional*, como técnicas de modelado de sistemas. Se utiliza Designer/2000 como herramienta CASE para automatizar los aspectos clave del proceso de diseño y construcción.
La metodología *CASE*Method* de Oracle proporciona técnicas para la ejecución de varias actividades. Incorpora técnicas de análisis, diseño y funciones de comprobación, las cuales garantizan la construcción de sistemas de alta calidad y desempeño. Debido a que Designer/2000 es totalmente compatible con la metodología en mención, el análisis del Modelo Entidad Relación y del Modelo Jerárquico Funcional está basado en ésta metodología.
El trabajo está dividido en 5 capítulos. El primer capitulo contiene una introducción a la metodología *CASE*Method*, se mencionan las fases que la componen y el objetivo de cada una de

estas fases; también se hace una descripción del conjunto de diagramadores y utilidades que forman parte de Designer/2000, herramientas indispensables en el modelado de sistemas.
En los capítulos 2 y 3 se realiza el análisis del *Modelo Entidad Relación* y del *Modelo Jerárquico Funcional*, respectivamente; estos capítulos contienen los detalles de los elementos disponibles para la construcción de estos modelos. Además se proporcionan los detalles de las herramientas de Designer/2000 que soportan y automatizan la construcción de estos modelos.
Por último, el propósito del capítulo 4 y 5 es mostrar como el diseño y desarrollo de sistemas es más flexible utilizando una herramienta CASE. Para ello, realicé el modelo de un sistema a partir del Diagrama Entidad Relación (DER) y del Diagrama Jerárquico Funcional (DJF), para obtener finalmente las tablas que podrán integrarse a una Base de Datos y las pantallas que serán parte de un sistema utilizado para leer archivos planos. Además, se incluye como anexos el DER, el DJF, el código para construir los objetos de la BD y los módulos (formas y reportes) generados con la herramienta CASE.

Además de Designer/2000 V6, las herramientas que se utilizaron para el desarrollo de este trabajo, son las siguientes:

- Personal Oracle 8 como Base de Datos (BD). Está diseñado para soportar el manejo de varios tipos de información, cuenta con una extensa gama de datos, lo que permite que crezca la capacidad de manipulación y administración de los mismos.

- Developer/2000 V6. Herramienta cliente/servidor e Internet para desarrollar aplicaciones que requieren de pantallas (formas), reportes o gráficas.

- SQL Plus. Herramienta para controlar, definir y manipular los datos dentro de la BD de una manera sencilla, con poderosas funciones.

Estas herramientas son productos de Oracle que trabajan e interactúan con la versión de Base de Datos Oracle 8. Y se encuentran en un ambiente de Windows NT versión 4.0.

CAPITULO 1.
DESIGNER/2000

"...el objetivo a largo plazo de las herramientas CASE es automatizar los aspectos clave de todo el proceso de desarrollo, desde el principio hasta el final. Para aquéllos que emplean el término *ingeniería de software asistida por computadora*, hacemos mención de que el desarrollo de una aplicación comienza con la especificación de requerimientos, no con la codificación del software...."
En general, las herramientas de tipo CASE incluyen los siguientes cinco componentes: herramientas para diagramación, un depósito de información, generadores de interfaces, generadores de código y herramientas de administración[1.1].

Este capitulo contiene una panorámica de la *metodología CASE*method*. También se describen las características principales de la herramienta *Oracle Designer/2000* y la forma en que está dividida.

1.1. METODOLOGIA DE DISEÑO DE SISTEMAS

El propósito de los sistemas de información es automatizar las funciones principales de las organizaciones que los emplean, agilizar y optimizar el procesamiento de la información para mejorar su productividad. Sin embargo, si no se realiza un análisis previo de las necesidades de la organización y un diseño de los sistemas adecuado tanto a las funciones de la organización como a la tecnología empleada, y aún teniendo la tecnología más moderna, la automatización puede ser ineficiente, no operativa y representar una mala inversión. Por lo tanto, cuando se decide o se tiene la necesidad de iniciar con un proyecto para desarrollar un sistema, es necesario contar con una metodología de desarrollo de sistemas.

No hay ninguna magia en el ciclo de vida del desarrollo de sistemas (SDLC, System Development Life Cycle). Aparece documentado en todos los libros de diseño y análisis de los últimos 20 años. Con la excepción de algunos cambios radicales, como la metodología del prototipado de mediados de los 80, no ha habido ningún cambio en el SDLC desde su aparición hace varias décadas. Todo ello se reduce, en esencia, a que todas las metodologías, independientemente de si se desarrollan internamente o se adquieren a una consultoría o gestora de software de proyectos, dirigen una estructura para planificar, analizar, diseñar, construir e implementar aplicaciones de negocios. Lo que las diferencia es su nivel de detalle, técnicas y su integración con el personal de procesos y tecnología[1.2].

Dentro de las metodologías utilizadas en el ciclo de vida del desarrollo de sistemas se encuentra la propuesta por *Richard Barker* conocida como *método CASE* (CASE*Method). Richard Barker fue una de las influencias impulsoras de la metodología CASE y de los productos CASE de Oracle.
El método de *Barker* plantea una secuencia de etapas detalladas y para cada etapa proporciona su descripción, definición de objetivos, productos de la etapa y la lista de tareas que conviene realizar. A continuación se mencionan las etapas básicas de éste método y una breve explicación de cada una de ellas, Figura 1.1:

[1.1] Senn, James. *Análisis y diseño de sistemas de información*, p. 293.
[1.2] Dorsey, Paul y Peter Koletzke. *Manual de Oracle Designer/2000*, p. 4.

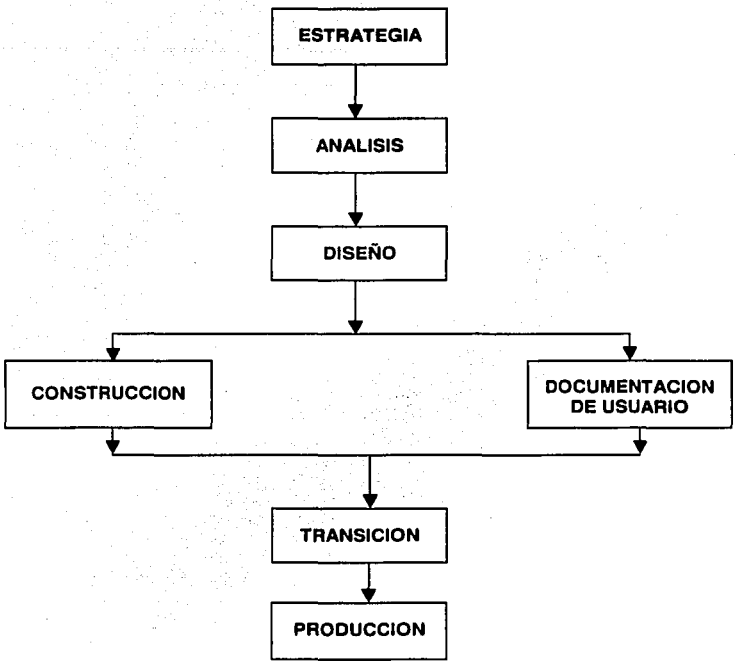

Figura 1.1. Etapas de la metodología CASE (CASE*Method): estrategia, análisis, diseño, construcción, documentación, transición y producción.

Estrategia.
El objetivo de la fase de estrategia es producir, bajo la dirección del usuario, un conjunto de modelos de negocio, un conjunto de recomendaciones y un plan acordado para el desarrollo de sistemas de información.
En esta etapa se debe lograr un entendimiento claro de las necesidades de la organización y del ambiente en que operará el sistema o sistemas a implantar. Con el fin de tener una visión desde el punto de vista de la dirección corporativa, se analizan las diferentes funciones que realiza la organización y sus necesidades de información a todos los niveles, para esto se realiza una serie de entrevistas con la dirección y los responsables de los departamentos. Así a partir de esta información se elabora un primer modelado de los requerimientos del sistema de información adecuado a las necesidades de la organización. Posteriormente para la definición de una primera versión de la arquitectura del sistema, además de los requerimientos antes obtenidos, se toman en cuenta la tecnología disponible y los sistemas de información que se encuentran en operación.
Los resultados de esta etapa son: un conjunto de modelos de la empresa, un conjunto de recomendaciones, un plan de desarrollo para los sistemas de información y una evaluación de la estrategia.

Análisis.
La fase de análisis consiste en obtener todas las especificaciones de usuario para el proyecto y detallar completamente todos los procesos del negocio que estarán implicados en el diseño del sistema. Barker describe el análisis como una extensión de la estrategia para asegurar la precisión y viabilidad del negocio y una sólida base para el diseño.
La etapa de análisis expone en forma más detallada los resultados obtenidos en la etapa de estrategia. Para asegurar la precisión de los modelos de la empresa y para especificar un fundamento sólido para el diseño, durante ésta etapa se realiza otra serie de entrevistas, ya no a un nivel directivo, sino a un nivel operativo y técnico. Con la participación de los responsables de la operación y de las funciones que serán automatizadas, se realiza un análisis detallado de sus requerimientos específicos en cuanto a objetivos, subfunciones, datos, etc.

El documento de análisis consta de las siguientes partes: flujos de procesos lógicos (¿Cuáles son las tareas?, ¿Qué tareas preceden a otra?, etc.), documento de requisitos, evaluación del análisis y se generan los modelos básicos del sistema:

- *Modelo Entidad Relación*, que modela mediante relaciones lógicas todos los datos involucrados en el sistema, de tal manera que cualquier tipo de explotación (consulta o modificación) sea posible.
- *Modelo Funcional*, que modela los diferentes servicios que ofrecerá el sistema mediante una organización y clasificación de las diversas funciones y subfunciones que fueron identificadas en el análisis.

Diseño.
La fase de diseño es aquella en la que se dejan los anteproyectos para empezar a construir el sistema. Se deben afinar todos los detalles antes de empezar la generación.
En esta etapa se toman los requerimientos y los modelos de la etapa de análisis para determinar la mejor manera de satisfacerlos, logrando niveles de servicios acordados. Es decir, del diseño conceptual se pasa al diseño final que será utilizado para la implantación, por ejemplo, en esta etapa el *Modelo Entidad Relación* será transformado en un *diseño de base de datos* y el *Modelo de Funciones* será transformado en *módulos y manuales de procedimientos*.

La fase de diseño se divide en dos partes:

- Diseño de la base de datos (BD). El diagrama entidad relación de diseño es el diagrama entidad relación final antes de la construcción y es recomendable construirlo con la asistencia del administrador de la base de datos (DBA)
- Diseño de la aplicación, con las especificaciones de los programas.

Construcción.
La fase de construcción comprende dos áreas: la base de datos y las aplicaciones. Si todas las fases anteriores han sido cuidadosamente desarrolladas, esta fase debería desarrollarse sin problemas.
A partir del diseño final generado en la etapa anterior, en ésta se codificarán y probarán los nuevos programas, usando las herramientas apropiadas. Los resultados de esta etapa son la base de datos afinada y la aplicación debidamente probada. Si las fases anteriores fueron realizadas cuidadosamente, la persona encargada de realizar las pruebas deberá comprobar que el sistema satisface las especificaciones realizadas durante el diseño y que funciona correctamente desde el punto de vista del usuario.

Documentación.
Uno de los productos fundamentales para el uso y el mantenimiento efectivo y eficiente de los sistemas son los manuales. Esta metodología incluye una etapa dedicada a esta actividad tan importante y hace hincapié para que en su elaboración se consideren el estilo de trabajo y las necesidades propias de los usuarios que utilizarán y mantendrán el sistema.
La documentación debería ser un proceso continuo a lo largo de todo el proceso de desarrollo del sistema.

Transición.
El desarrollo de un sistema no se termina con su programación, antes de su liberación, se debe asignar un período de transición que deberá incluir la alimentación de la nueva base de datos, la capacitación de los usuarios y el desarrollo de pruebas.
Durante ésta etapa, se debe realizar una prueba de aceptación del usuario. Proporcionar a los usuarios la nueva aplicación para que trabajen con ella y realicen transacciones reales utilizando el nuevo sistema.
En esta etapa se realizan todas las tareas necesarias para la implementación y se proporciona un periodo inicial de soporte al sistema. La transición debe llevarse a cabo con una interrupción mínima de la organización, debe dejar a los usuarios confiados y listos para explotar el nuevo sistema.

Producción.
Finalmente, en la etapa de producción se asegura que el sistema funciona correctamente, se realizan ajustes y mejoras al sistema, los cambios necesarios deberán ser introducidos sin afectar a los usuarios. El resultado de esta etapa es un sistema listo para su operación.

1.2. HERRAMIENTAS CASE

Durante los últimos veinte años han aparecido diferentes herramientas que han dado soporte a las técnicas de análisis y diseño estructurado más comunes y utilizadas:

- Diagrama de flujo de datos.
- Diagrama de descomposición funcional.
- Diagrama de jerarquía de funciones (pantallas).
- Diagrama entidad relación

Las herramientas CASE de análisis y diseño soportan distintas metodologías como: Yourdon o CASE*method de Richard Barker, por mencionar algunas. Tienen los elementos básicos necesarios para realizar adecuadamente todos los elementos más importantes de un desarrollo en las fases de análisis y diseño:

- Ambiente gráfico.
- Posibilidad de trabajar bajo diferentes metodologías.
- Diccionario de datos (repositorio) que permite reutilizar la información generada reduciendo la ambigüedad y redundancia al máximo nivel posible.
- Facilidad para documentar y generar informes.
- Integración gráfica entre diferentes fases (análisis y diseño) y modelos (procesos y datos).
- Soporte para generación de código, tanto de aplicaciones como de base de datos.
- Posibilidad de realizar procesos de *ingeniería inversa*.

Desde un punto de vista tradicional, una herramienta CASE debe tener, entre otras, las siguientes características:

- Cubrir adecuadamente alguna o todas las etapas fundamentales que se conciben en la actividad de desarrollo siguiendo el ciclo de vida más tradicional: análisis, diseño, construcción e implementación.
- Disponer de una serie de herramientas que permitan realizar de manera sencilla gráficos para cubrir algunos de los aspectos individuales de estas etapas.
- Poder almacenar la información en un diccionario de datos suficientemente compleja y completa como para aprovechar la información al máximo posible en todas las etapas del ciclo de desarrollo y entre los diferentes elementos gráficos empleados dentro de estas fases.

Estas son algunas de las herramientas CASE de análisis y diseño[1.3].

- Excelator.
- Visible Analyst.
- EasyCASE.
- System Arquitech.
- Erwin.
- Oracle Designer/2000.

[1.3] Cabrera, Gregorio y Guillermo Montoya. *Análisis y diseño detallado de aplicaciones informáticas de gestión*, pp. 418 - 422.

1.3. DESIGNER/2000

Oracle Designer/2000 es una herramienta que permite crear aplicaciones para el proceso de información, desde muy sencillas hasta con un alto grado de complejidad, con la habilidad de generar aplicaciones completas a partir de los modelos desarrollados con esta herramienta; o bien, puede usarse para crear definiciones a partir de sistemas en producción. Permite crear aplicaciones *cliente/servidor* funcionando óptimamente bajo diferentes tipos de configuraciones de red y diferentes protocolos de comunicación.
Debido a que esta herramienta soporta la metodología *CASE*method*, abarca todo el ciclo de vida en la creación de sistema de información.

1.3.1. CARACTERISTICAS PRINCIPALES

Las características principales de Designer/2000 son:

- Manejar múltiples versiones de una aplicación.
- Flexibilidad en cuanto a cambios en las necesidades del usuario y la demanda de crecimiento del sistema (escalabilidad).
- Cuenta con herramientas para soportar la *reingeniería de procesos del negocio* (BPR Business Process Reengineering).
- Permite compartir, manejar, accesar y procesar la información una vez que se ha identificado la información global del sistema antes de que la implementación del proyecto haya iniciado.
- Un repositorio compartido que asegura la consistencia de la información y permite a los desarrolladores compartir las definiciones comunes.
- Permite el trabajo en equipo donde múltiples desarrolladores y equipos comparten su repositorio.
- Generación de reportes sobre el estado actual de cualquier elemento de la aplicación.
- Un desarrollo rápido de aplicaciones mediante prototipos (RAD Rapid Application Development).
- Desarrollo de aplicaciones *cliente/servidor*.
- Soporta la *ingeniería inversa*.
- Esta basada en la metodología CASE (CASE*Method).
- Debido a que la herramienta soporta el ciclo de vida del desarrollo de sistemas tradicionales, es de gran utilidad tanto para lideres de proyecto, como analistas y desarrolladores.

Designer/2000 proporciona la facilidad de generar estructuras para los siguientes manejadores de base de datos relacionales (RDBMS)[1.4]:

- Oracle 7 y Oracle 8
- ANSI92
- Rdb 7.0 y 6.1
- DB2/2, DB2/MVS 4.X
- SQL Server 6.5, 6.0
- Sybase System 11

[1.4] Lounsberry, Joni y Otros. *Designer/2000 R2: Forms Design and Generation*, p. 1-12.

Esquema Cliente/Servidor.
El esquema Cliente/Servidor se compone de dos elementos:

1. ***Servidor.*** Quien se encarga de mantener almacenada la información en la base de datos, ya sea para consultas y modificaciones por parte del elemento cliente o para almacenar la nueva información enviada por éste. Además se encarga de atender todos los requerimientos de información o procesamiento de esta información solicitada por cada uno de los clientes mientras estos lo requieran, manejar los esquemas de seguridad, mantener la integridad de la información y administrar la concurrencia sobre la información almacenada en el servidor.
2. ***Cliente.*** Cuenta además con el elemento cliente quien toma los requerimientos solicitados por el usuario por medio de las aplicaciones y los envía al servidor. Una vez que el cliente ha recibido una respuesta del servidor, ya sean los datos solicitados o la negación de estos, le entrega el resultado al usuario final. Típicamente en el esquema Cliente/Servidor el elemento cliente se encuentra remoto o como parte de una red de área local.

Entre estos dos elementos (servidor y cliente) debe existir un medio de comunicación, el cual esta dirigido por el *protocolo de comunicación* quien se encarga de tomar los requerimientos de información solicitados por el cliente y empaquetarla, es decir, convertirlos a un formato entendible por parte del protocolo de comunicación, con el fin de que éste lo lleve hacia el servidor y el mismo elemento, pero en el lado del servidor, lo convierta a un formato que pueda interpretar el servidor y viceversa. Para esto, Designer/2000 cuenta con un producto adicional llamado *SQL*NET* y puede trabajar con protocolos tales como: TCP/IP, IP/SPX, DECNET, NAMEDPIPES y otros.
Con el *SQL*NET* se crea el medio de comunicación, entre el cliente y el servidor, conocido como "*cadena de conexión*". Puede existir más de una *cadena de conexión* cuando existe más de una base de datos y cada *cadena de conexión* estará apuntando a una base de datos en especial.

Designer/2000 es una herramienta CASE Cliente/Servidor basada en la metodología *CASE*Method*, está compuesta por *diagramadores* en el lado del cliente y un *repositorio multiusuario* en el lado del servidor. Esta arquitectura permite modelar aplicaciones en equipos de trabajo dado que la seguridad, integridad, consistencia y concurrencia de la información almacenada está completamente controlada por el manejador de base de datos en el servidor y administrada por el usuario desde el cliente.

Repositorio.
El repositorio es un depósito centralizado de información o diccionario de datos. En el nivel más bajo, el repositorio consiste en un conjunto de procedimientos almacenados en la base de datos y un conjunto de tablas que contienen la información sobre cada sistema que se esté analizando, diseñando y producido con el apoyo de la herramienta CASE.
El repositorio mantiene metadatos, es decir, datos sobre otros datos. Por ejemplo, lo que almacena es la información de las entidades y sus atributos, o de las tablas y sus columnas; los cuales tienen aspectos individuales (propiedades) que los definen. En Designer/2000 existe una jerarquía de elementos, siendo el nivel más alto el *sistema de aplicaciones*, que contiene todos los objetos de un determinado proyecto, o parte de un proyecto, de desarrollo de sistemas.
Hay que tomar en cuenta que los metadatos pueden ser estructuras de datos no reales. Por ejemplo, las definiciones de tablas en el repositorio no son tablas reales de la base de datos; sólo representan tablas, que pueden existir o no. El repositorio almacena el elemento tabla y los valores de sus propiedades, pero no existirá esa tabla en la base de datos hasta que se ejecute la sentencia que creara la tabla.

El repositorio está dentro del esquema o área de dominio de un usuario único en la base de datos de Oracle, conocido como el propietario del repositorio. Por tanto, antes de instalar Designer/2000, o bien se da de alta una nueva cuenta de usuario, o bien se elige una existente; en cualquier caso será el propietario de todas las tablas y códigos del repositorio, por lo regular es el líder del proyecto quien asume esta función. El propietario podrá otorgar privilegios a otros usuarios que

tengan la necesidad de trabajar en el ambiente de Designer/2000, con lo cual podrán manipular los objetos del repositorio.

En cuanto a la seguridad del repositorio, Designer/2000 cuenta con la *Utilidad de Administración del Repositorio (Repositori Admin Utility)*, que se encuentra en el panel de *Utilidades (Utilities)*, para llevar acabo el proceso de *Respaldar (back up)*, actualizar o restaurar un repositorio existente. Esta utilidad cuenta con la opción para respaldar los objetos del repositorio en un archivo de extensión "DMP" y se podrá realizar de tres formas:

- Sobre todos los objetos que pertenecen al dueño del repositorio, es decir, todos los objetos del repositorio, más los objetos de la base de datos que pertenecen al dueño del repositorio.
- Sólo las tablas del repositorio.
- Todas las tablas del repositorio (pero no otros objetos, como vistas o procedimientos), más otras tablas de la base de datos que pertenecen al dueño del repositorio.

Para actualizar o restaurar un repositorio existente se podrá realizar usando un archivo "DMP" creado mediante el proceso de respaldo (back up).

Uno de los componentes principales de una herramienta CASE es el *Repositorio*; los otros componentes son las herramientas que permiten a los usuarios introducir y consultar datos de este repositorio. Mediante el repositorio completamente compartido de Designer/2000, diferentes usuarios podrán trabajar conjuntamente en proyectos. Esto garantiza que no se producirán confusiones sobre requisitos del sistema y que mediante la reutilización, se puede aumentar al máximo la productividad del equipo de trabajo. El repositorio compartido proporciona también la infraestructura necesaria para gestionar la evolución de los sistemas a lo largo del tiempo.

Reingeniería de procesos del negocio.
La reingeniería de procesos del negocio (BPR Business Process Reengineering) es un rediseño radical del modo subyacente en el que una organización realiza sus tareas. Fue pregonada en primer lugar por los tecnólogos de la información. En los informes anuales de Computerworld de 1991 y 1992 los tecnólogos informaban de que la reingeniería era el asunto que más les concernía a ellos. A partir de entonces surgieron una plétora de seminarios, casas consultoras y altos directivos para satisfacer las demandas de conocimientos acerca de la reingeniería.
La reingeniería tiene los siguientes temas clave [1.5]:

- Implica un rediseño y reestructuración radical del trabajo.
- Utiliza la tecnología de información como una parte integral de ese diseño.
- Pretende conseguir mejoras espectaculares del rendimiento.
- Debe poner énfasis en el incremento de la calidad del producto.

Normalmente, cuando se rediseña un sistema es una buena oportunidad para no sólo reemplazar el sistema actual o automatizar un proceso manual, sino para pensar sobre qué cambios pueden hacerse a los procesos subyacentes del negocio para hacerlos más rápidos, más eficientes y que satisfagan mejor las necesidades de la organización. Un esfuerzo con éxito de reingeniería de procesos del negocio significa un espectacular incremento de la productividad, mediante cambios radicales en los procesos del negocio y la infraestructura de tecnología de información que los soportan.
La reingeniería de procesos del negocio no es una de las fases del proceso de desarrollo de sistemas tradicional, pero se puede añadir para extender el alcance del sistema propuesto. Designer/2000 cuenta con la herramienta *Modelador de procesos* (Process Modeler) para soportar la reingeniería de procesos del negocio.

[1.5] Dorsey, Paul y Peter Koletzke. *Manual de Oracle Designer/2000*. pp. 12, 217 – 223.

Ingeniería inversa.
Cuando es necesario desarrollar un sistema para actualizar, remplazar o integrarlo con uno ya existente, es probable que se tenga muy poca o ninguna documentación formal. Tampoco se conseguirá una información precisa o completa del sistema actual simplemente hablando con los usuarios, que seguramente no sabrán cómo se diseñó el sistema. En consecuencia, se complicará el desarrollo o mantenimiento del sistema. Para rescatar esta información Designer/2000 cuenta con las herramientas para poder aplicar una *ingeniería inversa*.

La ingeniería inversa es el proceso automático para crear definiciones en el repositorio a partir de sistemas en producción o para mantener al día las definiciones ya existentes. Este proceso evita tener que añadir manualmente las definiciones, actividad que sería consumidora de tiempo y propensa a errores. Hay dos tipos de utilidades de ingeniería inversa:

- Para objetos de base de datos. Permite la generación de objetos en el *repositorio* al nivel de diseño, con la información de los objetos de la base de datos (funciones, índices, paquetes, procedimientos, secuencias, tablas, trigger, Vistas, etc.).
- Para archivos de programas frontales (específicamente, Forms y Reports). Permite la generación de objetos en el repositorio al nivel de diseño, con la información del programa frontal. Para poder llevar acabo este proceso, es necesario tener en el repositorio todos los objetos de la base de datos, al nivel de diseño, involucrados en el programa frontal.

Realizar el proceso de ingeniería inversa es bastante sencillo sólo basta con iniciar la utilidad, especificar los objetos de la base de datos o el archivo de la aplicación que se desea traer al repositorio y pulsar el botón OK. El menú principal de Designer/2000 no cuenta con una opción para realizar esta tarea, pero puede realizarse mediante el menú *'Generate/Capture Design of'* del *Editor de diseño* (Design Editor), como se muestra en la Figura 1.2.

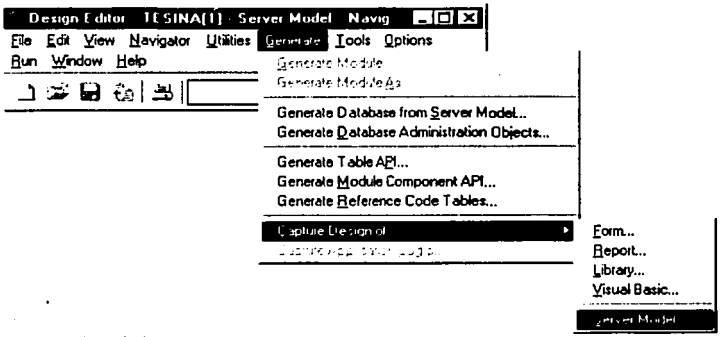

Figura 1.2. Menú de ingeniería inversa. Cuenta con las opciones: "*Form...* "."*Report...* " y "*Library...* ". Estas opciones podrán ser utilizadas para recuperar el diseño de módulos de formas (pantallas), reportes y librerías de Developer/2000 respectivamente. Con la opción "*Server Model...* " se podrá recupera el diseño de objetos de base de datos. Además cuenta con la opción "*Visual Basic...* " para recuperar el diseño de programas frontales de Visual Basic (archivos *.vbp).

La ingeniería inversa permiten reconstruir de manera rápida y precisa información al día de una base de datos en producción hasta obtener el diagrama entidad relación. Para obtener el DER de análisis de una base de datos, Designer/2000 cuenta 3 herramientas diferentes:

1. *La utilidad de ingeniería inversa de la base de datos* (*Server Model*) inserta automáticamente la información de las tablas de la base de datos en el repositorio, por lo que no hay que añadir manualmente las definiciones (actividad normalmente consumidora de tiempo y propensa a errores).
2. *La utilidad de retroencaje de tabla a entidad* crea automáticamente definiciones de entidades a partir de las definiciones de tablas creadas en el paso anterior. Esta utilidad esta disponible en el menú '*Utilities/Table to Entity Retrofit...*' del diagramador entidad relación.
3. *El diagramador entidad relación*, con esta herramienta se podrá crea uno o más diagramas para representar las entidades que creó la utilidad anterior. Sólo basta con crear un nuevo diagrama y elegir el menú '*Edit/Include*' para todas las entidades producto del retroencaje de tabla a entidad.

1.3.2. COMPONENTES DE DESIGNER/2000

Entrar a Designer/2000.

Este icono corresponde al acceso directo para entrar a *Oracle Designer/2000*. Al ser ejecutado, la herramienta mostrara la ventana de conexión solicitando: el usuario (*Username*), la clave de usuario (*password*) y la cadena de conexión (*Connect String*), como se muestra en la Figura 1.3.

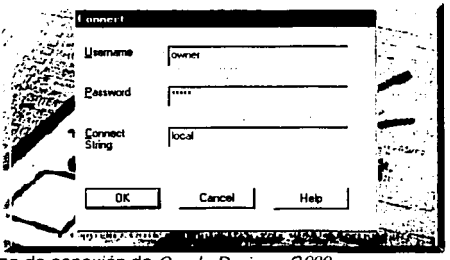

Figura 1.3. Ventana de conexión de *Oracle Designer/2000*.

Sistema de aplicaciones.

Como se mencionó anteriormente, el repositorio almacena todas las definiciones de objetos y existe una jerarquía de elementos, siendo el nivel más alto el *sistema de aplicaciones*, que contiene todos los objetos de un determinado proyecto, o parte de un proyecto, de desarrollo de sistemas. Es decir, bajo el *sistema de aplicaciones* existirán todos los elementos creados con las herramientas de Designer/2000, durante el desarrollo de un proyecto. Asimismo, pueden existir varias

aplicaciones, cada una, representando diferentes proyectos o diferentes versiones de un mismo proyecto. Debido a que las aplicaciones son totalmente independientes, el repositorio puede almacenar diferentes versiones de un mismo proyecto. La Figura 1.4. corresponde a la venta del *sistema de aplicaciones* (application Syste), la cual muestra todas las aplicaciones existentes ya sea para seleccionar alguna o para crear una nueva.

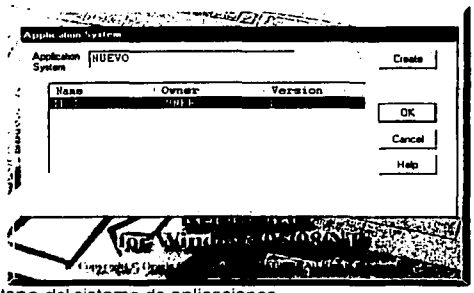

Figura 1.4. Ventana del sistema de aplicaciones.

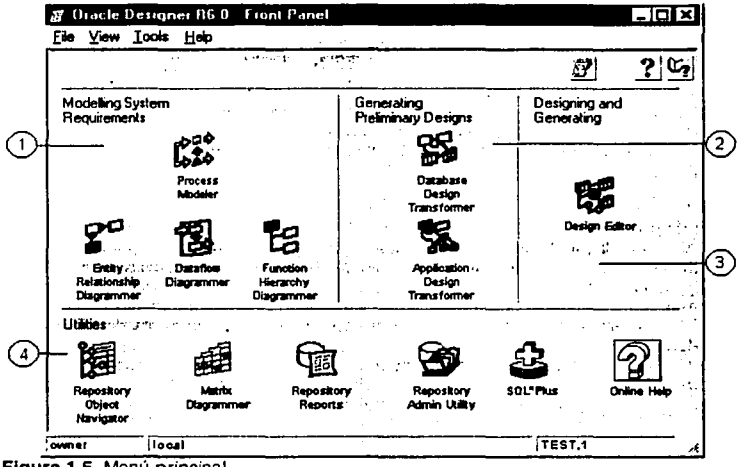

Figura 1.5. Menú principal.

Para crear una aplicación sólo basta con insertar un nombre en el campo *application System* y presionar el botón *Create*, la aplicación será creada y estará disponible en la ventana de aplicaciones para ser seleccionada. Las aplicaciones podrán ser creadas por el propietario o por los usuarios del repositorio que cuenten con los privilegios necesarios.
Para eliminar una aplicación, se podrá hacer con el menú "*Application/Delete...*" de la herramienta "*Repository Object Navigator*" (RON).

Menú principal.
El menú principal se encuentra dividido en 4 partes, como se muestra en la Figura 1.5:

1. Modelando los Requerimientos del Sistema (Modelling System Requirements).
2. Generando el Diseño Preliminar (Generating Preliminary Designs).
3. Diseñando y Generando (Designing and Generating).
4. Utilidades (Utilities).

1.3.2.1. MODELANDO LOS REQUERIMIENTOS DEL SISTEMA

La sección para modelar los requerimientos del sistema cuenta con las herramientas para proporcionar soporte a las actividades relacionadas con las fases de estrategia y análisis, donde es importante tener una buena comunicación con el usuario para el desarrollo del sistema. En ésta parte se aclaran los conceptos y términos, se describe tanto el modelo de datos como el funcional, se ven y se definen los requerimientos. Consta de 4 diagramadores:

1. Modelador de Procesos (Process Modeler).
2. Diagramador Entidad Relación (Entity Relationship Diagrammer).
3. Diagramador de Flujo de Datos (Dataflow Diagrammer).
4. Diagramador Jerárquico Funcional (Function hierarchy Diagrammer).

Modelador de procesos.
El objetivo del modelador de procesos es obtener un cuadro completo de alto nivel del sistema para la comunicación con los directivos o usuarios. Este modelador corresponde y proporciona soporte a la etapa de estrategia. En él se esquematizan los procesos en detalle, se identifican los que son clave, los que pueden mejorarse y también se modelan nuevos procesos antes de ser implementados. Además, esta herramienta se puede utilizar como asistente de la reingeniería de procesos del negocio.
Esta herramienta permite describir los procesos de la organización y almacenar sus detalles en tiempo y costo. Muestra los procesos actuales, como trabajarán en el futuro y permite identificar oportunidades para mejorar los procesos actuales. Cuenta con un diagramador para crear y mantener las definiciones de los procesos del negocio en el repositorio multiusuario, información que estará disponible para ser usada en la etapa subsecuente de análisis. Características principales:

- Puede ser utilizado para crear un diagrama que represente los departamentos o grupos de una empresa.
- Se puede descomponer un proceso de alto nivel en otros de nivel más bajo.

- A cada paso del proceso y/o flujo se le puede definir tiempo, costo y calidad.
- Acceso al repositorio para crear y actualizar las definiciones de los procesos.
- Características de animación y multimedia que muestran los flujos de procesos de manera dinámica.
- Los objetos del diagrama están disponibles para su uso en etapas posteriores.

Los objetos del diagramador son los siguientes (ver Figura 1.6):

1. Unidad organizacional (Organization unit). Todos los procesos pertenecen a una unidad organizacional. Algunas veces se refiere al rol de una persona o grupo, más que a un departamento. El significado de su representación es mostrar al responsable o la fuente de las actividades.
2. Evento de entrada (Trigger). Es el evento de entrada que da comienzo a los procesos.
3. Procesos (Process step). Representa cada actividad dentro del proceso base. Un *Proceso* que solicita información es un *data entry* y uno que la proporciona es un *report*. El proceso que se utiliza para tomar decisiones o para evaluar cierta condición se llama *decision point*.
4. Almacén (Store). Simboliza una colección de datos que no forman parte de un flujo sino que está almacenada en algún sitio.
5. Flujo (Flow). Muestra el flujo de datos de un proceso a otro o a un almacén.
6. Evento de salida (Outcome). Evento que termina la secuencia de los procesos, es decir, es el evento de salida.

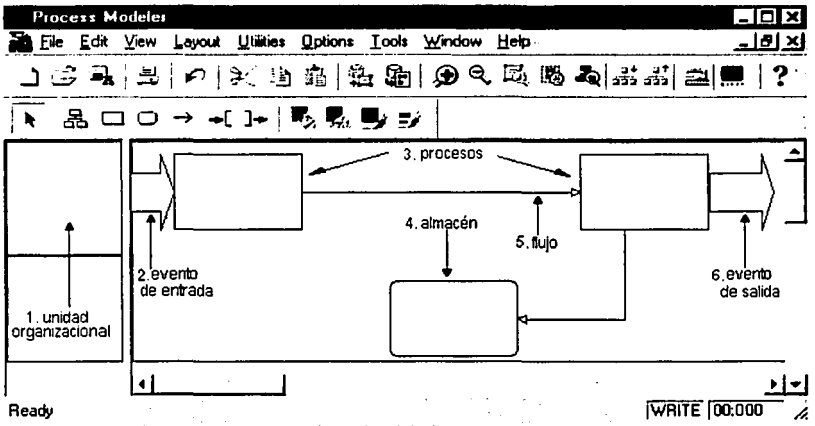

Figura 1.6. Ventana de la herramienta Modelador de procesos (Process Modeler).

Diagramador de flujo de datos.
Por medio del Diagrama de Flujo de Datos (DFD) se representa la información, de entrada y de salida, del modelo para poder determinar los procesos involucrados dentro del sistema; además, muestra como fluye la información a través de toda la organización. El diagramador permite la creación de funciones, almacenamientos, flujos de datos y entidades externas. Es útil durante la

etapa de análisis para complementar totalmente el modelado de los procesos y flujos que estructuran el modelo del negocio.
Este diagramador muestra la misma información que el Modelador de procesos, pero con un tipo diferente de diagrama que resulta más familiar para los analistas. Es importante mencionar que Designer/2000 no hace distinción entre un *proceso* y una *función*.
Una función es aquella actividad que representa qué es lo que un negocio hace o necesita hacer. Cada diagrama de flujo de datos representa una función. Características principales del diagramador:

- Facilidad de crear diagramas de flujo de datos donde sus objetos son almacenados en un repositorio multiusuario.
- Muestra la descomposición de funciones por medio de sus niveles.
- Permite ilustrar el alcance de la operación del sistema.
- Facilidad de autolayout. El autolayout es utilizado para mejorar la presentación del diagrama.
- Ayuda a la comunicación con el usuario y diseñadores.

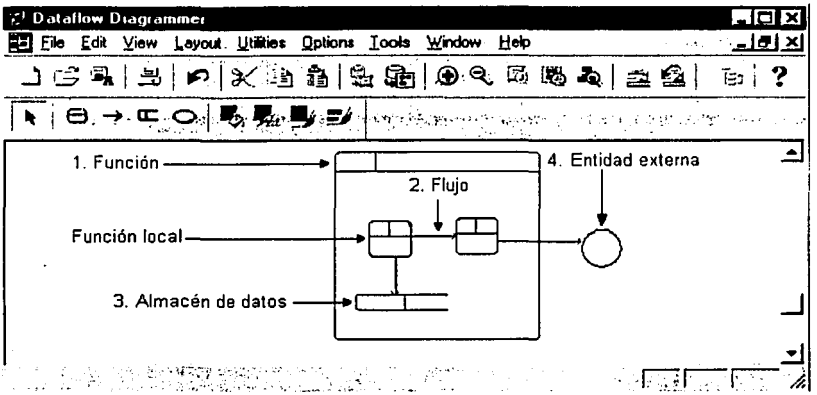

Figura 1.7. Ventana del Diagramador de flujo de datos

En la Figura 1.7 se muestran los objetos del diagramador:

1. Función (Function). Es la función que se va a detallar, dentro de la función se puede tener funciones locales o fuera de ella las globales. Se puede bajar de nivel para crear funciones hijas con la opción: FILE/OPEN UP y subir con FILE/OPEN DOWN.
2. Flujo (Dataflow). El Flujo de datos une las funciones, almacenamientos y entidades externas con otras funciones.
3. Almacén de datos (Datastore). Indica el almacenamiento de datos, es usado por funciones o procesos. El *datastore* generalmente hace referencia a las entidades y usualmente contiene atributos de más de una entidad, cuando esto sucede es recomendable escoger un nombre que lo refleje.

4. Entidad externa (External). La entidad externa se refiere a un objeto que se encuentra fuera del alcance del sistema, por ejemplo, otro sistema o una unidad de negocio, con el que se necesita interactuar.

En cuanto al tema del **Diagramador Entidad Relación (DER)** y del **Diagramador Jerárquico Funcional (DJF)** se abordará en el capitulo 2 y el capitulo 3 respectivamente.

1.3.2.2. GENERANDO EL DISEÑO PRELIMINAR

Para generar el diseño preliminar, Designer/2000 cuenta con dos tranformadores:

1. Transformador para el diseño de la base de datos (Figura 1.8). Transforma el modelo de entidades en un diseño de base de datos. La descripción de éste transformador se encuentra en el *Capitulo 4*.

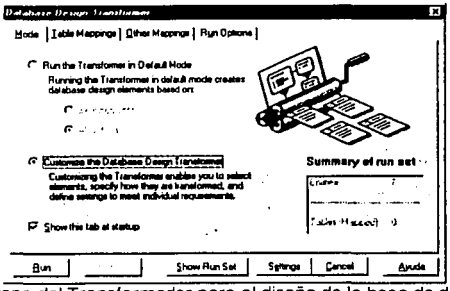

Figura 1.8. Ventana del Transformador para el diseño de la base de datos.

2. Transformador para el diseño de aplicaciones (Figura 1.9). Utilizado para transformar las funciones del negocio en diseño de aplicaciones. La descripción de éste transformador se encuentra en el *Capitulo 5*.

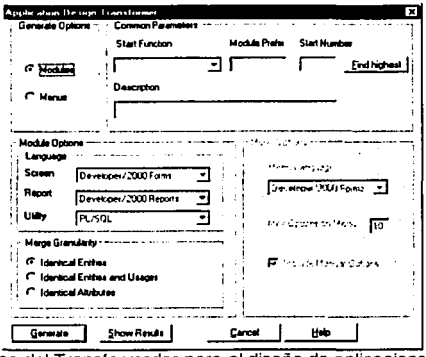

Figura 1.9. Ventana del Transformador para el diseño de aplicaciones.

Estas herramientas son útiles para trasladar la información desde los elementos de diseño lógico (entidades, atributos o funciones) a los elementos de diseño "físico" (tablas, columnas o módulos). El diseño preliminar es una parte intermedia entre el análisis y el diseño. Cuando se generan los elementos de diseño "físico", éstos estarán disponibles para ser editados y modificados con las herramientas del *Editor de diseño* (Design Editor).

1.3.2.3. DISEÑANDO Y GENERANDO

El *Editor de diseño* (Design editor) es una herramienta indispensable durante las etapas de diseño y generación. Esta herramienta permite crear uno o más diagramas para diseñar el modelo de la base de datos. Cada diagrama es una representación gráfica de los objetos (tablas, vistas, constraints, etc.) de la base de datos.
También facilita la creación de diagramas para el diseño de aplicaciones. Cada diagrama es una representación gráfica de un módulo (forma o reporte) con sus componentes. En cada diagrama es posible definir la lógica del módulo y su funcionalidad en general, por ejemplo, que datos podrán ser accesados y cuales serán desplegados.

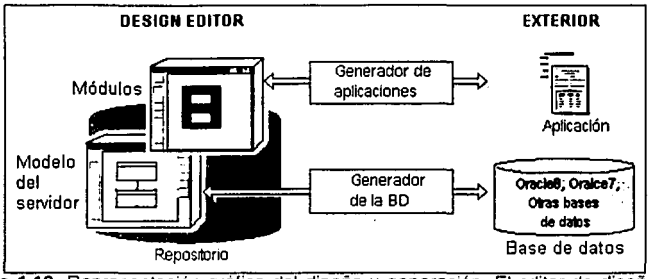

Figura 1.10. Representación gráfica del diseño y generación. El editor de diseño es utilizando para realizar el diseño final de los objetos de la base de datos y de los módulos. La información de cada diagrama es utilizada por el generador de la base de datos y generador de aplicaciones respectivamente.

Una vez que se ha completado el diseño, es posible generar los objetos físicos en la base de datos y los programas frontales desde el editor de diseño, ver Figura 1.10. Para generar las aplicaciones, es posible seleccionar el generador que corresponde a cada tipo de programa frontal que se desea generar.

Características del *Editor de diseño* (*Design Editor*).

- Cuenta con una interfaz gráfica para el diseño de los objetos de la BD y una para el diseño de módulos (programas frontales).
- Permite generar archivos (DLL) de SQL ANSI para crear los objetos de la base de datos como Oracle7, Oracle8 y otros tipos de BD.
- Generar el servidor API (Application Programming Interface) para ser usado por las aplicaciones del cliente para controlar las operaciones que se realicen sobre las tablas: insertar, borrar y actualizar.
  El servidor API es una poderosa interfaz PL/SQL que las aplicaciones del cliente podrán llamar para ejecutar consultas (querys) y operaciones sobre las tablas de la base de datos de Oracle; o bien, para ser usado por las aplicaciones de *Developer/2000 - Form*. Se podrá generar directamente en la base de datos de Oracle o como archivos DDL (scripts).

- Soporta la generación de aplicaciones para el cliente, por ejemplo: formas y reportes de *Developer/2000*; aplicaciones en Visual Basic; páginas dinámicas de Web para realizar consultas sobre servidores de Web tomando los datos de una base de datos Oracle; y aplicaciones en otros tipos de lenguajes de programación.
- Permite el proceso de *ingeniería inversa*. Genera información a partir de aplicaciones existentes para ser almacenada en el repositorio.

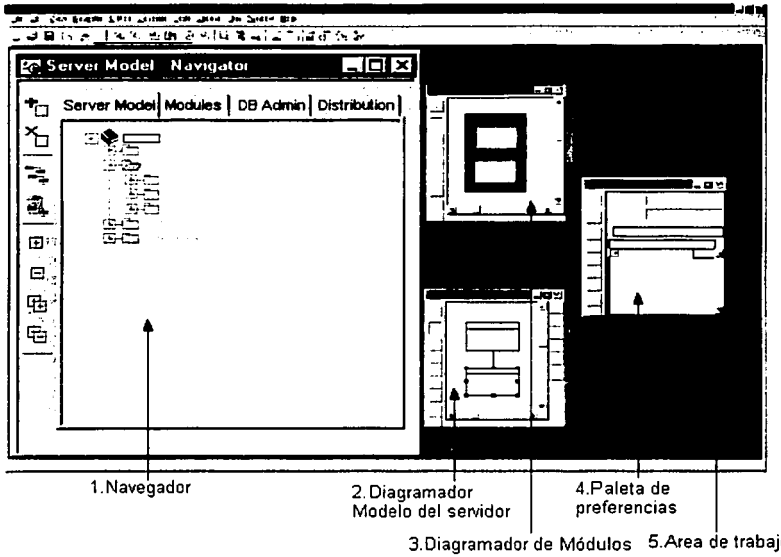

Figura 1.11. Componentes principales del Editor de diseño.

Componentes principales del *Editor de diseño*, ver Figura 1.11:

1. Navegador. Es una estructura de árbol donde se pueden ver y manejar las definiciones del repositorio que participaran en el diseño de la aplicación o de la base de datos. Cuenta con 4 carpetas o 'Tabs': *Modelo del servidor (Server Model)*, utilizado para crear el diseño final de la base de datos; *Módulos (Modules)*, que nos presenta las definiciones del repositorio necesarias para el diseño de aplicaciones; *DB Admin*, en el que se registran los detalles para la administración de la BD; y *Distribución (Distribution)*, para registrar los detalles de la distribución de la base de datos.
   Algunas definiciones del repositorio son incluidas en mas de una carpeta del navegador, por ejemplo, definiciones de tablas que son mostradas en la carpeta *Modelo del servidor* y en la carpeta *Módulos*. En la primer carpeta se encontrarán los elementos asociados a la tabla para el diseño de la BD como índices, triggers y sinónimos. Y para la segunda carpeta sólo

encontraremos los elementos de la tabla que son útiles para el diseño de módulos, como columnas y llaves.

2. Diagramador Modelo del servidor (Server Model Diagram). Muestra el diseño de la base de datos en un modelo gráfico.
3. Diagramador de Módulos (Module Diagram). Representación gráfica del diseño de módulos.
4. Paleta de preferencias (Preference Palette). Es una ventana de propiedades para afinar la apariencia de los módulos.
5. Area de trabajo. Es el espacio en donde se despliegan las ventanas de los diagramas. Para crear un diagrama (Modelo del servidor o módulo), primero se deben seleccionar los elementos apropiados en el navegador y arrastrar hasta el área de trabajo, al soltar la selección de elementos automáticamente se creará el diagrama.

Cuando se usa un diagrama para crear, editar o borrar una definición del repositorio, el navegador es actualizado automáticamente para reflejar los cambios. Si los cambios son hechos por otro usuario que comparte el repositorio, se dispara inmediatamente un mecanismo que marcara con una bandera al navegador, lo cual indica que las definiciones del repositorio han cambiado.

1.3.2.4. UTILIDADES

Para que una herramienta CASE sea altamente productiva debe soportar el uso de una metodología. A lo largo de todo el ciclo de vida del desarrollo de sistemas es importante contar con utilidades de distinto tipo para comprobar, cambiar o complementar la información del repositorio. Designer/2000 cuenta con un panel de utilidades para administrar el repositorio:

1. Navegador de Objetos del Repositorio (Repository Object Navigator 'RON').
2. Diagramador de Matrices (Matrix Diagrammer).
3. Reportes del Repositorio (Repository Reports).
4. Utilidad de Administración del Repositorio (Repository Admin Utility).
5. SQL*Plus.
6. Ayuda (Online Help).

Navegador de objetos del repositorio (RON).
El RON muestra el contenido del repositorio y cuenta con dos ventanas:

- Ventana de navegación, para ver todos los objetos del repositorio: entidades, funciones, módulos, etc.
- Ventana de propiedades, con las características de algún objeto seleccionado en la ventana de navegación.

Permite manipular todos los objetos existentes en el repositorio, independientemente de donde fueron creados, incluyendo la creación y mantenimiento de las aplicaciones. Además se puede usar durante la etapa de diseño y modelado de la base de datos, ver Figura 1.12.

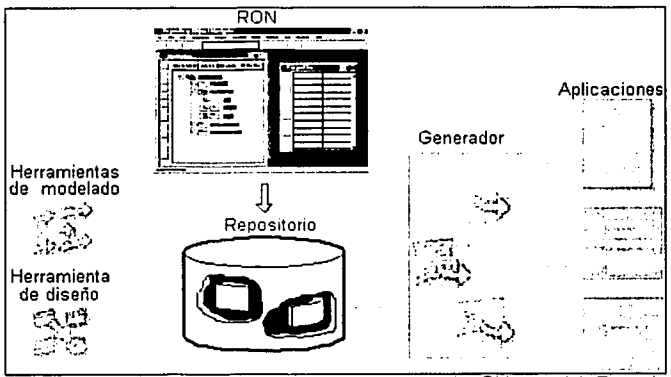

Figura 1.12. Representación gráfica del Navegador de Objetos del Repositorio "RON". Toda la información de los modelos y diseños es almacenada en el repositorio y se podrán ver, modificar o borrar con el RON. También es posible llamar a los generadores desde el RON para generar la aplicación final.

Diagramador de Matrices.
El diagramador de matrices presenta y hace una referencia cruzada, renglones por columnas, de la información contenida en el repositorio. Usualmente requerida para la creación de la matriz CRUD de "funciones vs. Entidades". Se conoce como CRUD por sus siglas en ingles:

Create (crear)
Retrieve (recuperar)
Update (actualizar)
Delete (borrar)

Estas siglas son empleadas para denotar las operaciones permitidas de una función sobre las entidades.
Un diagrama de matriz consiste en dos y en algunos casos de tres ejes, cada eje muestra un diferente tipo de elemento, la intersección de las celdas indica que existe una relación o asociación entre los dos elementos. Las matrices son útiles en las etapas de análisis y diseño, por ejemplo: en la etapa de análisis son útiles para determinar que entidades están asociadas a las funciones; y en la etapa de diseño, para establecer que tablas se incluirán en los diferentes módulos.

Reportes del Repositorio.
Esta herramienta proporciona alrededor de 100 reportes predefinidos con los cuales se podrá examinar el contenido del repositorio con la información más relevante de módulos, funciones, tablas, columnas y demás objetos creados durante el desarrollo de la aplicación.
Además existen reportes útiles para el control de calidad, por ejemplo, el de entidades que no tiene atributos. Y para usuarios avanzados, estos podrán definir sus propios reportes.
La ventana de reportes contiene una lista desde donde se podrá escoger alguno para ser ejecutado. La lista de reportes podrá ser consultada de 3 maneras diferentes:

- Por grupos, por ejemplo, el modelo de datos (Data Model) que contiene reportes de entidades, atributos y relaciones.
- Por tipo de objeto, como el reporte de entidades o funciones.
- Alfabéticamente por el nombre del reporte.

Utilidad para la administración del repositorio.
Desde donde se instala y configura el ambiente del repositorio de Designer/2000. Esta herramienta podrá ser usada para:

- Instalar un nuevo repositorio.
- Remover total o parcialmente el repositorio.
- Desplegar información acerca del contenido del repositorio.
- Control sobre el acceso de usuarios.
- Editar o ampliar la estructura del repositorio.
- Respaldar (back up). Existe la opción para realizar un respaldo del repositorio en un archivo extensión "*.DMP*".
- Actualizar o restaurar un repositorio existente. Restaurar el contenido del repositorio usando un archivo "*.DMP*" creado mediante el proceso de respaldo (backup).

La herramienta está disponible sólo para el administrador, conocido como el dueño (owner) del repositorio.

SQL*Plus.
Para invocar una sesión de SQL*Plus, tomando automáticamente para la conexión el usuario y la clave de usuario (password) proporcionado al iniciar la sesión con Designer/2000.

Ayuda (Online Help).
Permite el acceso a la ayuda en línea de Designer/2000, incluyendo introducciones e índices por tema, así como un extenso glosario.

1.3.2.5. BARRA DE HERRAMIENTAS

La barra de herramientas (toolbar) de los diagramadores, cuenta con una serie de botones con las instrucciones más comunes, a continuación se mencionan los botones que se encuentran en todos los diagramadores.

 Crear un diagrama nuevo.

 Abrir un diagrama.

 Guardar un diagrama.

 Imprimir el diagrama que actualmente está abierto.

 Cortar el objeto seleccionado.

 Copiar el objeto seleccionado.

 Pegar un objeto.

 Hacer una actualización en todo el diagrama.

 Acercar la imagen del diagrama.

 Alejar la imagen del diagrama.

 Para acercar el área seleccionada: primero seleccionar un objeto y luego presionar el botón.


Para acercar el área: primero presionar el botón y luego seleccionar el área.


Ajustar el tamaño para ver todo el diagrama.


Ayuda.

Además de los anteriores, cada diagramador cuenta con botones propios. Los siguientes botones también son de uso común pero sólo están en algunos diagramadores.


Deshacer el último cambio.


Actualizar el objeto seleccionado.


Autolayout, mejorar la apariencia del diagrama, de un objeto o de varios objetos seleccionados.


Layout anterior, es decir, para regresar a la apariencia anterior del diagrama cuando se utiliza el *Autolayout*.

CAPITULO 2.
MODELO ENTIDAD RELACION

El modelo entidad relación, conocido como Diagrama Entidad Relación (DER), es una técnica para definir las necesidades de información de una organización. Este provee un firme fundamento para entregas de alta calidad, con sistemas apropiados que satisfacen las necesidades del negocio. Este modelo es de gran utilidad en las fases de estrategia, análisis, diseño y construcción de la metodología Case. Para poder construir éste modelo, Designer/2000 cuenta con una herramienta llamada *Diagramador Entidad Relación* (*Entity Relationship Diagrammer*). En este capitulo se hace un análisis del modelo (diagrama) entidad relación y la forma en que Designer/2000 soporta la construcción del diagrama entidad relación.

2.1. MODELO ENTIDAD RELACION

En su forma más simple, el modelo entidad relación implica identificar las cosas de importancia dentro de una organización (entidades), las propiedades de esas cosas (atributos) y como se relacionan entre sí (relación).

Objetivos del diagrama entidad relación.
Proporcionar un modelo exacto de las necesidades de información de la organización, que actuará como marco de trabajo para el desarrollo de sistemas nuevos o para modificar el existente. Proporcionar un modelo independiente de cualquier almacenamiento de datos y cualquier método de acceso, que permita tomar decisiones objetivas de las técnicas de implementación y la coexistencia con sistemas antiguos.

El Diagrama Entidad Relación (DER) en las etapas de la metodología CASE:

- Estrategia. El propósito del DER en la etapa de estrategia es identificar las entidades principales del negocio, pero puede no ser necesario identificar todos los atributos de esas entidades. Lo importante es identificar las entidades clave y sus relaciones, para proporcionar una perspectiva de la información que se maneja en el negocio.
  Es útil para demostrar una comprensión preliminar del negocio. El único objetivo es definir las necesidades del usuario al más alto nivel, no es necesario detallar todos los atributos de las entidades, es suficiente con las relaciones y unos cuantos atributos clave con nombres adecuados. También se podría representar el DER sin atributos e incluir definiciones textuales de cada entidad.
- Análisis. El DER de análisis es una representación detallada de los requisitos del negocio relacionados con los datos. El objetivo es representar tantas reglas de negocio, relativas a los datos, como sea posible.
  En ésta fase es importante crear todas las entidades significativas para el sistema, así como definir todos los atributos y las relaciones para cada entidad. Durante la creación del diagrama, es necesario dedicar tiempo a nombrar y describir correctamente las entidades y atributos.
  Después de la presentación del DER, debe quedar claro a todo el mundo lo que representa: un diagrama que será utilizado como modelo lógico de los requisitos relacionados a los datos del área de negocio que se está analizando.

Es importante mencionar que el DER representa sólo los elementos relacionados a los datos, elementos que serán utilizados cuando se asignan entidades y atributos a las funciones. Por lo tanto, es recomendable tener todas las entidades y atributos antes de comenzar a definir la asociación entre funciones y entidades.

- Diseño. En esta etapa se tiene el DER final antes de la construcción y corresponde al diseño de la base de datos. Se toma la información del DER de análisis para realizar el diseño de las tablas y columnas, junto con la especificación detallada de los dominios, validaciones (constraints) de las columnas y los disparadores (trigger) correspondientes. Es importante revisar el diseño con la asistencia del administrador de la base de datos (DBA, DataBase Administrator) para afinar todos los detalles antes de iniciar la construcción.
- Construcción. Después de realizar el análisis del DER y una vez que se afinaron todos los detalles del diseño de la base de datos, se toma toda la información de éste último para construir la *base de datos* que será utilizada por los programas frontales.

2.1.1. COMPONENTES DEL MODELO ENTIDAD RELACION

El diagrama entidad relación tiene los siguientes componentes, como se muestra en la Figura 2.1:

1. Entidad
2. Relación
3. Atributo (opcional y obligatorio)
4. Identificador único.

Existen algunas convenciones que son utilizadas para simplificar los diagramas y poder mostrar información adicional:

5. Arco de exclusividad.
6. Entidad Supertipo / Subtipo.

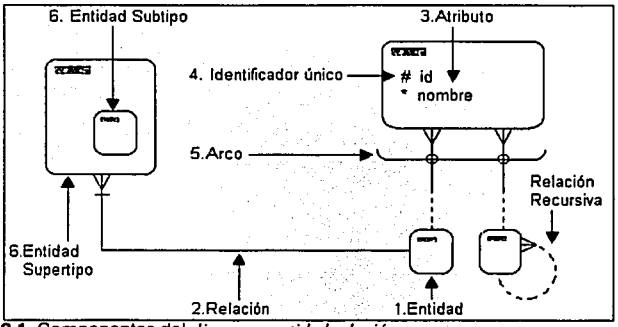

Figura 2.1. Componentes del *diagrama entidad relación.*

2.1.1.1. ENTIDAD

Una entidad es un ente con significado real o conceptual, acerca del cual el negocio o sistema que se está modelando necesita guardar información. La entidad tiene atributos, que agrupan los datos más representativos para identificar las diferentes ocurrencias.

Representación.
Una entidad se representa por medio de una caja con las esquinas redondeadas con un nombre, mostrando, o no, sus detalles o atributos, como se muestra en la Figura 2.2.

Figura 2.2. Representación de una *entidad*.

La caja puede ser de cualquier tamaño, suficiente como para contener un nombre, que permita tener las líneas de "*relación*" para conectar las entidades y evitar cruces de líneas innecesarias o el diagrama se vera como una "telaraña".

Reglas y convenciones para las entidades[2.1].

- Una cosa u objeto sólo puede ser representada por una entidad. Esto es, entidades mutuamente exclusivas en todos los casos.
- El nombre de la entidad se muestra en singular y con letras en mayúsculas. El nombre debe ser uno que represente un tipo o clase de objeto, no una instancia.
- Cada entidad debe tener identificadores. Esto es, cada ocurrencia (instancia) de una entidad puede ser separada y distintamente identificable de todas las otras instancias de ese tipo de entidad.

Ocurrencias (Instancias).
Las ocurrencias son los registros que se guardarán dentro de las entidades. Por ejemplo, "GUANAJUATO" o "DURANGO" no son nombres convenientes para una entidad, el nombre de la entidad podría ser "ESTADO" y los anteriores dos instancias de ésta entidad.

[2.1] Barker, Richard. *CASE*METHOD, Entity Relationship Modelling*, p. 3-2.

2.1.1.2. RELACION

Una relación es la asociación que existe entre las entidades y es binaria, en el sentido de que es siempre una asociación entre exactamente dos entidades; con una excepción, entre una entidad y ella misma (relación recursiva). Cada relación tiene dos extremos, para cada uno de los cuales tiene:

- Un nombre.
- Cardinalidad/grado (cuantos).
- Opcionalidad (opcional u obligatorio).

Estas propiedades se utilizan para describir la asociación desde un extremo, pero se deben definir ambos extremos de la relación.

Representación.
Una relación se representa mediante una línea que une dos cajas de entidades, o recursivamente (recurrentemente) une una caja de entidad consigo mismo. La relación más común es la que tiene un grado de "uno a muchos": en el extremo de "muchos" es obligatoria y opcional en el extremo "uno", como se muestra en la Figura 2.3.

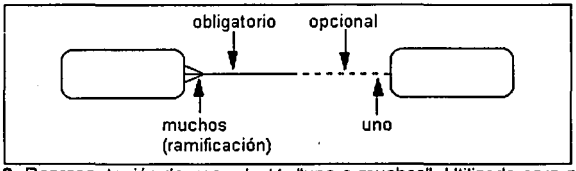

Figura 2.3. Representación de una *relación* "uno a muchos". Utilizada para pensar en la existencia de una relación "padre-hijo", con la condición de que un hijo existe sólo si existe el padre.

Nombre de la relación.
El nombre de cada extremo de una relación se sitúa cerca del extremo apropiado en minúsculas, es decir, cerca de la entidad correspondiente, como se muestra en la Figura 2.4.

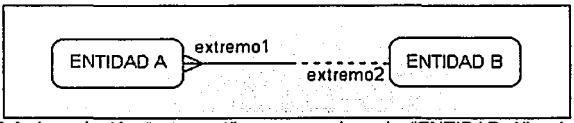

Figura 2.4. La relación "extremo1" corresponde a la "ENTIDAD A" y la relación "extremo2" corresponde a la "ENTIDAD B".

Cardinalidad.
La cardinalidad representa la asociación entre una (1) o más (M) ocurrencias de una entidad y una (1) o más (M) ocurrencias de otra entidad, es decir, el número de ocurrencias que participan en cada extremo de la relación. La cardinalidad se muestra como una línea con terminación de tres puntos, conocido como "ramificación" o "pata de gallo", para representar el lado "*M*" (*Muchos*) de una relación. Y una línea sencilla para el lado "*1*" (*Uno*) de una relación, es decir, la línea se une solamente en un punto de la entidad.

Opcionalidad (opcional u obligatorio).
La opcionalidad representa la condición de existencia, para las instancias, de las entidades que participan en una relación. La condición puede ser: opcional u obligatoria.
Para representar el extremo de una relación "*obligatoria*" se dibuja una línea continua, para esa mitad de la relación. Y para representar el extremo de una relación "*opcional*" se dibuja una línea discontinua, para esa mitad de la relación.

Lectura del diagrama entidad relación.
Las relaciones pueden ser leídas como sentencias completas que describen las reglas del negocio. Para la lectura se usa la siguiente sintaxis:

1. Se inicia la sentencia con la palabra "Cada", más
2. Nombre de la entidad inicial, más
3. Opcionalidad de la entidad inicial (debe ser o estar / puede ser o estar), más
4. Relación de la entidad inicial a la entidad final, más
5. Cardinalidad que llega a la entidad final (uno y sólo uno / uno o más), más
6. Nombre de la entidad final o su plural cuando la cardinalidad implique más de un elemento.

Ejemplo:
La lectura de la relación entre la entidad PAIS y la entidad ESTADO, seria la siguiente (Figura 2.5):

Relación	***Sintaxis***					
	1	***2***	***3***	***4***	***5***	***6***
Uno a uno:	Cada	ESTADO	debe ser	de	un y solo un	PAIS
Uno a muchos:	Cada	PAIS	puede	tener	uno o más	ESTADOS

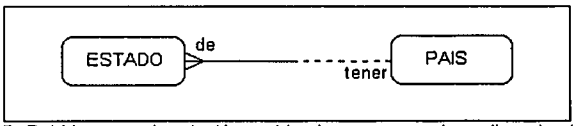

Figura 2.5. Debido a que la relación es binaria, es necesario realizar dos "lecturas" para describir completamente la relación:
Lectura 1: Cada ESTADO debe ser de un y solo un PAIS.
Lectura 2: Cada PAIS puede tener uno o más ESTADOS.

Relaciones recursivas.
Las relaciones recursivas son aquellas que se dan entre ocurrencias de la misma entidad y sirven para modelar jerarquías. Siempre son opcionales ya que de otra manera se estaría creando una jerarquía infinita, como ejemplo se muestra la Figura 2.6.

Figura 2.6. Relación recursiva. Por ejemplo, en una entidad llamada "EMPLEADO":
- Un EMPLEADO puede ser manejado por un y sólo un EMPLEADO.
- Un EMPLEADO puede estar a cargo de uno o más EMPLEADOS.

Algunas Relaciones son más comunes que otras. La Figura 2.7 muestra algunas relaciones validas e invalidas: "Uno a muchos", "uno a uno" y "muchos a muchos".

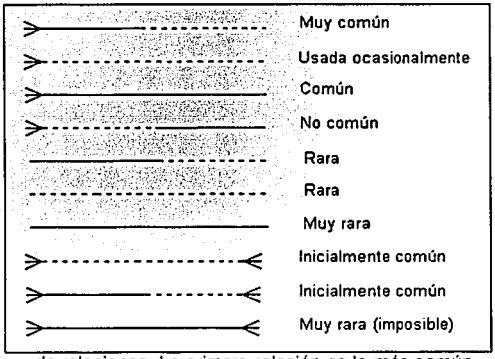

Figura 2.7. Tipos de relaciones. La primera relación es la más común.
La tercera relación (uno a muchos, obligatoria en ambos lados) tiene un significado especial, es la relación "maestro – detalle" vista a menudo en transacciones.
Las relaciones "muchos a muchos" son más comunes en versiones iniciales de un modelo, pero estas relaciones deberán ser resueltas con entidades de intersección al final del análisis.

La Figura 2.8 muestra algunas relaciones recursivas validas e invalidas: "Uno a muchos", "uno a uno" y "muchos a muchos".

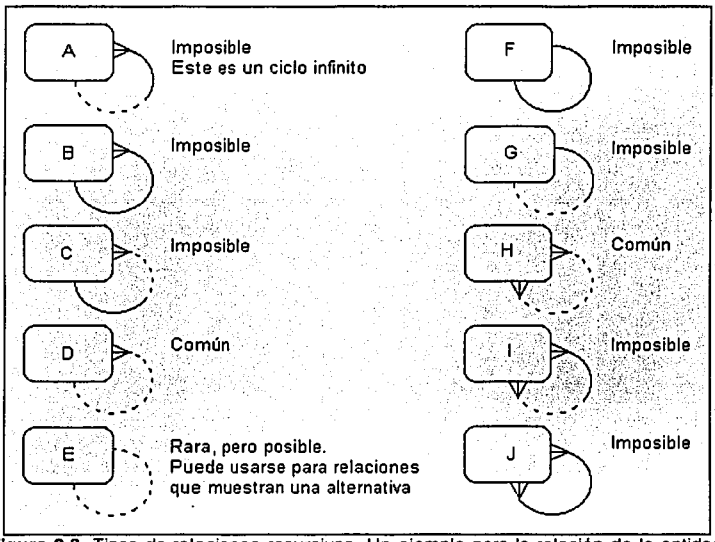

Figura 2.8. Tipos de relaciones recursivas. Un ejemplo para la relación de la entidad "H", muchos a muchos y opcional en ambos lados, es el siguiente:
"Cada COMPONENTE puede componerse de uno o más (otro) COMPONENTES y cada COMPONENTE puede ser usado en uno o más (otro) COMPONENTES".

Entidad de intersección.
Las relaciones de muchos a muchos son comunes durante la fase de estrategia y al principio del análisis. Al final de la fase de análisis deberían estar resueltas; no pueden implementarse en tablas de un sistema de base de datos relacional, así que cada una de estas relaciones debe resolverse descomponiéndose en dos relaciones "**1**" a "**M**" asociadas a una entidad denominada *entidad de intersección* o asociativa, creada con este propósito[2.2].

Como se menciono en el párrafo anterior, la resolución de una relación "*muchos a muchos*" se consigue insertando una nueva *entidad de intersección* entre las dos entidades conocidas como *entidades de referencia* y estableciendo una relación entre cada una de las entidades de referencia y la entidad de intersección.
Las relaciones entre la entidad de intersección y la entidad de referencia deben tener las siguientes características, Figura 2.9:

[2.2] Dorsey, Paul y Peter Koletzke. *Manual de Oracle Designer/2000*. p. 234.

- La opcionalidad de la relación de la *entidad de intersección* a la *entidad de referencia* debe ser "*debe*".
- La opcionalidad de la relación de la *entidad de referencia* a la *entidad de intersección* debe ser la misma que tenía la relación de esta *entidad de referencia* en cuestión a la otra.
- La cardinalidad de cada *entidad de referencia* a la *entidad de intersección* debe ser "*muchos*".
- La cardinalidad de la *entidad de intersección* a cada *entidad de referencia* debe ser "*uno*".

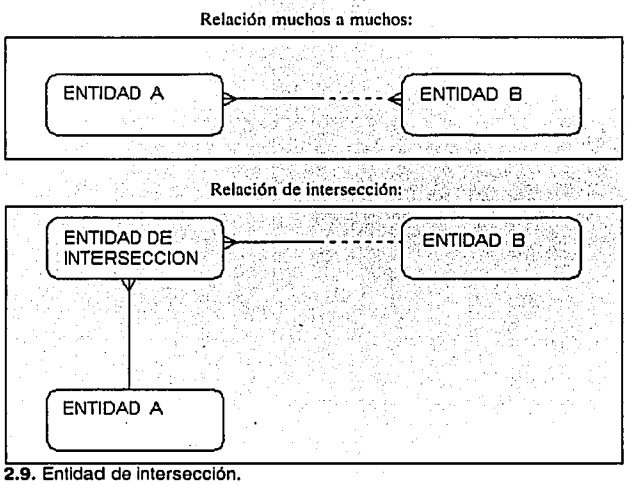

Figura 2.9. Entidad de intersección.

Redundancia.
Un diagrama de entidad relación no debería tener ninguna relación que pudiera, en todas las circunstancias, derivarse de otras relaciones. En una base de datos, archivo o implementación manual de relaciones, la redundancia es común para asegurar la ejecución.

2.1.1.3. ATRIBUTO

Un atributo es una propiedad que sirve para caracterizar, identificar, clasificar, cuantificar o expresar el estado de la entidad. Las entidades normalmente contienen dos o más atributos.
Las relaciones son el resultado de la interacción entre dos entidades, de las cuales dependen para su existencia. Los atributos son cada una de las porciones de información de un elemento de entidad. Los atributos definen las propiedades de las entidades. Hay tres tipos de atributos[2.3]:

- Los que identifican a cada ocurrencia de identidad, de los que uno de ellos es el identificador principal o clave primaria, que identifica inequívocamente a cada elemento de entidad.
- Los que describen la ocurrencia de entidad. Son cada una de las propiedades características de esa ocurrencia de entidad.
- Los que se refieren a ocurrencias de otra tabla. A estos atributos se les llama clave ajena. El número total de atributos que se emplean para definir una entidad en un sistema depende del uso que vayan a recibir.

Definiendo un atributo.
Un atributo debe tener un nombre significativo en singular y en minúsculas. El nombre debe ser único dentro de la entidad. Debe existir un valor por atributo para cada ocurrencia de un tipo de entidad. Para cada atributo es necesario definir:

- Nombre.
- Tipo de dato que almacenará.
- Si es un identificador único "#".
- Opcionalidad (opcional "°" u obligatorio "*").
- Formato.
- Si tendrá valor inicial.
- Rango y/o valores permitidos (dominio).

Representación.
El nombre del atributo se precede con un símbolo:

Símbolo	Descripción
#	Para indicar que un atributo es parte del *identificador único* y siempre debe contener un valor. Consiste en una combinación de atributos y/o relaciones que identifican cada ocurrencia de una entidad de manera única.
*	Símbolo utilizado para indicar que el atributo es *obligatorio*, es decir, el atributo debe contener siempre un valor.
°	Este símbolo se utiliza para indicar que el atributo es *opcional*, es decir, puede o no tener valor.

[2.3] Cabrera, Gregorio y Guillermo Montoya. *Análisis y diseño detallado de aplicaciones informáticas de gestión*, p. 112.

Ejemplo:
Para la entidad AFILIADO, se definen los siguientes atributos, como se muestra en la Figura 2.10:

Nombre	Dominio	Opcional / Identificador único	Formato	Valor inicial	Tipo	
id folio		#			Char(12)	
nombre		*			Char (60)	
a paterno		*			Char(30)	
a materno		*			Char(30)	
calle		*			Char(60)	
rfc		°	AAAA999999AAA		Char(13)	
cve municipio		*		D.F	Char(6)	Arteaga Calvillo Ecatepec

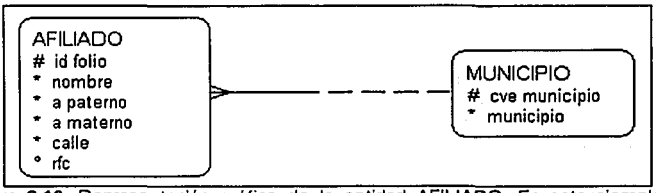

Figura 2.10. Representación gráfica de la entidad AFILIADO. En este ejemplo la entidad MUNICIPIO será utilizada como "lista de valores".

Dominio.
Es un conjunto de reglas de validación, restricciones de formato y otras propiedades que se aplican a un grupo de atributos. Los atributos que se encuentran en el dominio están sujetos al mismo conjunto de validaciones. Por ejemplo: Una lista de valores o un rango.

Las relaciones como atributos.
Las relaciones son tan importantes como las entidades, se deberán tomar como atributos que estarán en la entidad "*hijo*" y serán una copia de los atributos de la entidad "*padre*", dichos atributos serán los que formen parte del identificador único de la entidad "*padre*".

2.1.1.4. IDENTIFICADOR UNICO

El identificador único corresponde a cualquier combinación de atributos y/o relaciones que sirve, en todos los casos, para identificar únicamente una ocurrencia de una entidad. El identificador único siempre será obligatorio. Un identificador único puede ser:

- Uno o más atributos.
- Una o más relaciones.
- Una combinación de atributo(s) y relación(es).

Representación.

Símbolo	Descripción
#	Símbolo utilizado para indicar que un atributo es (o es parte) de un identificador único.
	Para indicar que una relación es parte del identificador único, se representa por medio de una barra vertical sobre la relación y cerca de la entidad a la cual se le transferirá el identificador único.

Si ya existe un atributo candidato obvio único en el negocio que está en uso, se recomienda utilizar éste como identificador único. De otra manera podrá utilizar una combinación de atributos y/o relaciones.

Ejemplo:
De la Figura 2.11, se tienen los siguientes identificadores únicos:

Entidad	Identificador único
PAIS	El atributo "cve pais".
ESTADO	El atributo "cve estado" y la relación con la entidad PAIS (cve pais).

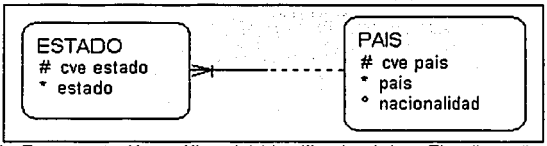

Figura 2.11. Representación gráfica del identificador único. El atributo "cve pais" se deberá tomar como un atributo más de la entidad ESTADO y será parte del identificador único.

Clave foránea.
La *clave foránea* es el conjunto de atributos de una entidad que forman el *identificador único* en otra entidad y se presenta cuando la relación no es parte del *identificador único*. La "opcionalidad" de la *clave foránea* dependerá de la "opcionalidad" de la relación.
Por ejemplo, en la Figura 2.12. se muestra una relación que no es parte del *identificador único*. Por lo tanto, los atributos para la entidad ESTADO serían los siguientes: "cve estado" como identificador único; "estado" como atributo obligatorio; "cve pais" como clave foránea y es obligatorio, ya que la opcionalidad de la relación para el extremo de la entidad ESTADO es obligatorio.

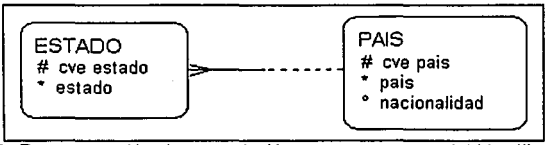

Figura 2.12. Representación de una relación que no es parte del identificador único. En este ejemplo el atributo "cve pais" se deberá tomar como un atributo más de la entidad ESTADO y será una clave foránea obligatoria.

Relaciones transferibles y no transferibles.
Cuando una relación puede ser transferida de una ocurrencia de una entidad a una ocurrencia diferente de la misma entidad, se conoce como relación transferible. Inicialmente todas las relaciones son transferibles. Una relación que forma parte de un identificador único, normalmente no será transferible.
Una relación no transferible se representa, al final de la relación correspondiente, con el siguiente símbolo: <>

Respecto a la lectura de las relaciones no transferibles se puede agregar la siguiente cláusula:

"y éste nunca podrá ser transferido".

Tomando como ejemplo la relación de la Figura 2.13, la lectura sería la siguiente:

Cada ESTADO debe ser de un y sólo un PAIS, y este nunca podrá ser transferido a otro PAIS.

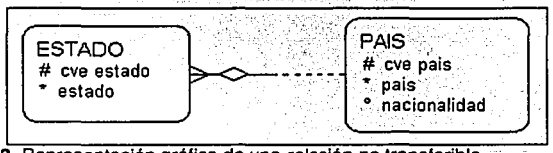

Figura 2.13. Representación gráfica de una relación no transferible.

2.1.1.5. RELACIONES EXCLUSIVAS (ARCOS)

Permite identificar cuando dos o más relaciones de la misma entidad pueden ser mutuamente exclusivas.

Representación.
Se representa dibujando un arco cruzando cada una de las relaciones involucradas y la línea del arco con los finales encorvados. Debido a que los arcos pueden cruzar otras relaciones no involucradas, se deben mostrar las relaciones incluidas con un círculo, como se muestra en la Figura 2.14.

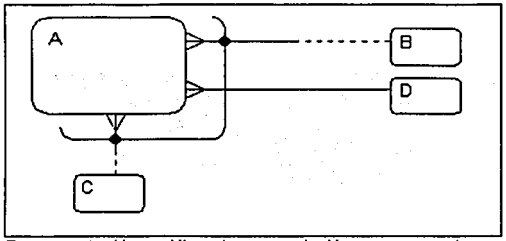

Figura 2.14. Representación gráfica de una relación con arco de exclusividad. En este ejemplo las relaciones de las entidades "B" y "C" están involucradas en el arco y la relación de la entidad "D" no está involucrada en el arco.

Lectura.
Respecto a la lectura de las relaciones exclusivas, se utiliza como ejemplo la Figura 2.15:

Cada PEDIDO debe ser entregado por uno y sólo un REPARTIDOR, o entregado en una y sólo una CAJA.

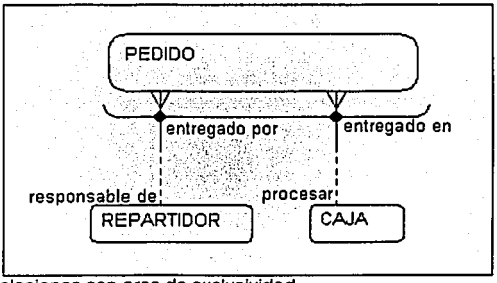

Figura 2.15. Relaciones con arco de exclusividad.

Nota:

- Los arcos sólo pueden cruzar terminaciones de relaciones en donde todas sean obligatorias o todas sean opcionales, deben tener la misma opcionalidad.
- Una terminación de relación sólo puede estar en un arco.
- Un arco debe cruzar al menos dos terminaciones de relaciones, y normalmente no más de tres o cuatro.
- Las terminaciones de las relaciones incluidas en el arco deben estar asociadas a la misma entidad y no se pueden incluir desde entidades de subtipo ni supertipo.
- Un arco de un grupo de relaciones es equivalente a un grupo de subtipos de la entidad.

2.1.1.6. SUPERTIPOS Y SUBTIPOS

Subtipos.
Un subtipo es una entidad. Una entidad se puede dividir en uno o más subtipos mutuamente exclusivos, cada uno de los cuales tienen atributos y/o relaciones. Los atributos comunes y/o relaciones se definen explícitamente sólo una vez en el nivel más alto. Los subtipos pueden tener tanto atributos como relaciones en sí mismas, se puede dividir en otros subtipos de niveles más bajos, y así continuamente, pero empíricamente se ha mostrado que dos o tres niveles son suficientes salvo en circunstancias más inusuales.
Una entidad subtipo hereda implícitamente todos los atributos y relaciones de la entidad al nivel más alto, conocido como *supertipo*.
Cuando una entidad tiene subtipos, todas las instancias de la entidad deben ser clasificadas en alguno de ellos.

Supertipos.
Es un medio de clasificar una entidad que tiene subtipos. Una entidad puede ser tanto un subtipo de una entidad "X" como un supertipo de otra entidad "Y".
Tanto las entidades subtipo como supertipo deben ajustarse a todas las normas de una entidad, ya que eso es lo que son.

Representación.
Los subtipos se dibujan dentro del supertipo, como se muestra en la Figura 2.16.

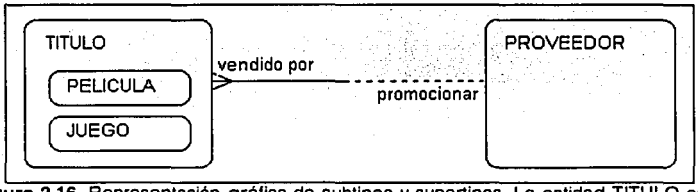

Figura 2.16. Representación gráfica de subtipos y supertipos. La entidad TITULO es un supertipo. Los subtipos son las entidades PELICULA y JUEGO.

Lectura.

- Cuando se lee el nombre del supertipo se añade { que puede ser un *subtipo1* o *subtipo2* }.
- Cuando se lee el nombre del subtipo hay que añadir { que es un tipo de *supertipo* }.

Ejemplo.
La lectura para la relación de la Figura 2.16, sería la siguiente:

1. *Cada TITULO, que puede ser una PELICULA ó un JUEGO, debe ser vendido por uno y un solo proveedor.*
2. *Cada PROVEEDOR puede promocionar uno o varios TITULOS, que pueden ser PELICULAS o JUEGOS.*

2.2. DIAGRAMADOR ENTIDAD RELACION

Para construir el *modelo entidad relación*, Designer/2000 cuenta con el *diagramador entidad relación*. Esta herramienta se concentra en la parte de los datos y en los detalles de los datos, ofreciendo un medio efectivo para especificar y controlar dicha información por medio de entidades, atributos y relaciones. Sus características:

- Los datos aquí representados son esencialmente lógicos.
- Facilidad para crear diagramas entidad relación, donde sus objetos son almacenados en un repositorio multiusuarios.
- La información está disponible para ser tomada por otras herramientas, por ejemplo, para realizar la asociación entre funciones y entidades desde el *diagramador jerárquico funcional*.
- Acceso al *Transformador para el diseño de la base de datos (Database Design Transformer)*, herramienta que automatiza el desarrollo del sistema derivando un diseño de base de datos inicial a partir del diagrama entidad relación.
- Facilidad para modificar automáticamente la presentación (autolayout) del diagrama.
- Soporte e implementación de entidades *supertipo* y *subtipo*. Así como, de *relaciones recursivas* y *relaciones con arco de exclusividad*.

Sus objetos.
El diagramador cuenta con los objetos necesarios para dibujar todos los componentes del *modelo entidad relación*, como se muestra en la Figura 2.17:

1. Entidad
2. Relación
3. Atributo. Es importante mencionar que Designer/2000 muestra el nombre de un atributo en mayúsculas, mientras que la metodología CASE propone usar letras minúsculas.
4. Identificador único.
5. Arco de exclusividad.
6. Entidad Supertipo / Subtipo.

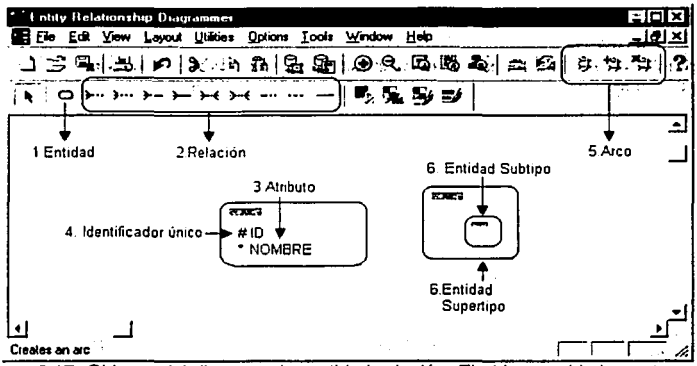

Figura 2.17. Objetos del diagramador entidad relación. El objeto entidad cuenta con una ventana de propiedades para definir los identificadores únicos y los atributos.

2.2.1. DIBUJANDO LOS OBJETOS DEL DIAGRAMADOR ENTIDAD RELACION

Este diagramador facilita la construcción del Diagrama Entidad Relación (DER). Es tan sencillo como pulsar el botón de la barra de herramientas del elemento (entidad o relación) que se desea crear y llevarlo al área de trabajo.
Para crear un nuevo DER se podrá realizar con en el menú "*File\New*" o presionar el botón con el icono de la hoja blanca que se encuentra en la barra de herramientas. En caso de que el diagrama exista podrá utilizar el menú "*File\Open*" o presionar el botón con el icono del folder. Con cualquiera de las opciones anteriores, aparecerá el *Tool Palette* y el área de trabajo lista para crear, modificar, incluir o borrar cualquier componente del DER.

Crear Entidades.

Basta con seleccionar el botón de entidad, pulsar una vez con el ratón en el área de trabajo y aparecerá una ventana en donde podrá insertar los siguientes datos: Nombre, Nombre Corto y el Nombre en Plural.

Ejemplo:
Para crear las entidades *PAIS* y *ESTADO*, es necesario dar por lo menos el nombre de la entidad Para el nombre corto, el diagramador tomara los 3 primeros caracteres del nombre y para "Plural" le agregará una "S" al nombre, como se muestra en la Figura 2.18. Por último, presionar el botón "OK" para crear la entidad o en su defecto "Cancel".

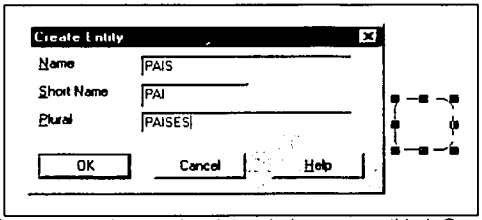

Figura 2.18. Ventana para insertar los datos de la nueva entidad. Cuando se utilice el *Transformador para el diseño de la base de datos*, el "nombre corto" será utilizado, opcionalmente, como prefijo para construir el nombre de los campos y el "plural" será utilizado para construir el nombre de la tabla.

Para crear entidades *subtipo/supertipo*, se crea primero la entidad que será *supertipo* y posteriormente se crea sobre ésta, la entidad *subtipo*. Un cuadro de alerta se mostrará indicando el nombre de la entidad que será *subtipo* de la otra entidad.

Definir los atributos de la entidad.
Para crear los atributos de una entidad, se podrá hacer seleccionando la entidad y pulsando dos veces sobre ésta (o utilizando el menú *Edit\Properties*). Realizando lo anterior, se mostrará la ventana *Edit Entity*. La carpeta *Attributes*, de la ventana *Edit Entity*, cuenta con las siguientes opciones para definir los atributos y sus características, como se muestra en la Figura 2.19:

- Name. Nombre del atributo.
- Seq. Para indicar la secuencia (orden) de los atributos.
- Domain. En caso de existir, se mostrará una lista para seleccionar el dominio que el atributo podría tomar.
- Opt. Está opción se utiliza para indicar si el atributo es opcional, cuando no está marcado indica que es obligatorio.
- Format. Se muestra una lista para seleccionar el tipo de dato que almacenará el atributo: numérico (NUMBER), fecha (DATE), carácter (VARCHAR2), etc.
- MaxLen. Para indicar la longitud del campo, cuando es requerido, por ejemplo en el caso de los atributos tipo *DATE* no es necesario indicar la longitud.
- Dec. El número de decimales que tendrá un atributo tipo numérico (NUMBER), es opcional.
- Primary. Marcar esta opción para indicar que el atributo será parte de la llave primaria.
- Comment. Para hacer comentarios o notas sobre el atributo, es opcional con una longitud de 80 caracteres. La información capturada en este campo es sumamente importante, pues será parte del diccionario de datos y cuando se generan las formas desde Designer/2000, se utilizará como ayuda para el campo relacionado con éste atributo.

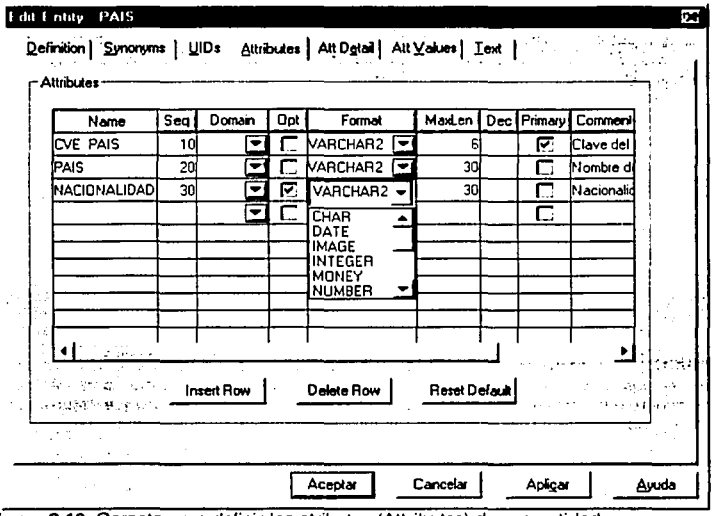

Figura 2.19. Carpeta para definir los atributos (Attributes) de una entidad.

Además, la carpeta *Attributes* cuenta con botones para insertar o borrar atributos, para aceptar, aplicar o cancelar los cambios y el botón de ayuda con la información relacionada a la carpeta seleccionada.

Nota.
Toda la información capturada en la ventana *Edit Entity* será almacenada en el repositorio y servirá como parte de la documentación del sistema que se está desarrollando. Al mismo tiempo, se estará construyendo el diccionario de datos.

Ejemplo:
Atributos para las entidades *PAIS* y *ESTADO*:

PAIS.

Name	Seq	Domain	Opt	Format (MaxLen/Dec)	Primary	Comment
CVE PAIS	10	S/D	*	VARCHAR(6)	SI	Clave del País, máximo de 6 caracteres
PAIS	20	S/D	*	VARCHAR(30)	NO	Nombre del País, máximo de 30 caracteres
NACIONALIDAD	30	S/D	°	VARCHAR(30)	NO	Nacionalidad, máximo de 30 caracteres y es opcional.

ESTADO

Name	Seq	Domain	Opt	Format (MaxLen/Dec)	Primary	Comment
CVE ESTADO	10	S/D	*	VARCHAR(6)	SI	Clave del Estado, máximo de 6 caracteres
ESTADO	20	S/D	*	VARCHAR(30)	NO	Nombre del Estado, máximo de 30 caracteres

Donde:

- S/D = Sin Dominio.
- * = Obligatorio.
- ° = Opcional.

Crear la relación entre las entidades.
Para crear la relación, el diagramador cuenta con varios botones ubicados en la barra de herramienta, a la derecha del botón *Entidad*. Solo basta con seleccionar el botón correspondiente a la relación deseada, pulsar una vez sobre la entidad final (hijo) y finalmente pulsar una vez más sobre la entidad inicial (padre). Realizando lo anterior, se mostrará la ventana "*crear relación*" (Create Relationship) para insertar el nombre de la relación para la tabla final "*From Name*" y el nombre de la relación para la tabla inicial "*To Name*".

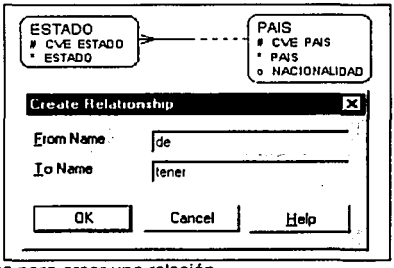

Figura 2.20. Ventana para crear una relación.

Ejemplo:
La relación entre la entidad *PAIS* y la entidad *ESTADO* sería la siguiente, ver Figura 2.20:

Entidades	From Name	To Name	Tipo Relación
PAIS ESTADO	de	tener	DE UNO (opcional) A MUCHOS (obligatorio).

La relación cuenta con una ventana de propiedades: *Edit Relationship*. Esta ventana se puede editar pulsando dos veces sobre la línea de la relación. La carpeta *Edit Relationship - Definition* cuenta con las siguientes opciones, ver Figura 2.21:

- Cambiar el nombre de la relación: From Name, To Name.
- Cambiar la opcionalidad (Optionality) y la cardinalidad (Degree).
- Opción para indicar si la relación será parte del identificador único (Primary UID) en la entidad final, conocida como "detalle" o "hijo".
- Opción para indicar si la relación será transferible (Transferable). Se define como no transferible desactivando ésta casilla de verificación.

En la parte inferior de la ventana están los botones para aceptar, aplicar o cancelar los cambios y el botón de ayuda.

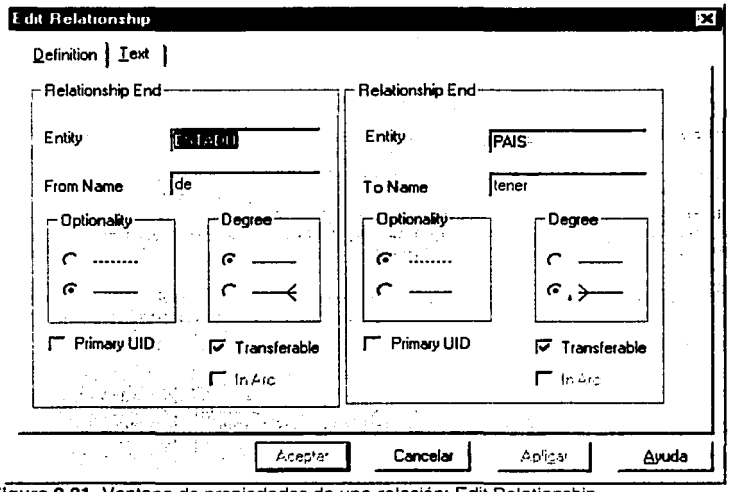

Figura 2.21. Ventana de propiedades de una relación: Edit Relationship.

Para crear una *relación recursiva*, el procedimiento es similar: seleccionar el botón de la relación correspondiente, pulsar una vez sobre un lado de la entidad y posteriormente pulsa una vez más sobre otro lado de la misma entidad.

Respecto al *arco de exclusividad*:

- Si se necesita crear una relación de arco: seleccionar las relaciones involucradas y con el menú ***Utilities\Create Arc***.
- Para agregar una relación: seleccionar la relación, el arco y con el menú ***Utilities\Add to Arc***.
- Para eliminar una relación de arco: seleccionar la relación, el arco y con la opción "***Utilities\Remove from Arc***".

Nota.
Es importante recordar que una vez que se han creado los elementos de cualquier diagramador, estos podrán ser incluidos en cualquier cantidad de diagramas, del mismo tipo, sin necesidad de crearlos nuevamente, sólo se tendrán que incluir. Por ejemplo, para el DER en caso de existir una entidad o relación en el repositorio se podrá incluir con la opción "***Edit\Include***", o presionando el botón derecho del ratón en el área de trabajo y se mostrará un menú del que podrá seleccionar la opción "***Include\...***". Ya sea para una entidad o para una relación, se mostrará una lista de la cual se podrá seleccionar el objeto a incluir.

También es importante tener presente que un objeto puede ser eliminado del diagrama, pero **no del repositorio !**. Para eliminar un objeto del diagrama, se utiliza el menú "***Edit\Cut***". Para eliminar el objeto del diagrama y del repositorio, se utiliza el menú "***Edit\Delete***" (o presionando la tecla DEL). Cualquier cambio que se realice sobre un objeto, de cualquier diagrama, se verán afectados todos los diagramas que contengan dicho objeto.

Beneficios del diagramador.

- Facilidad para construir el DER.
- Provee oraciones definitivas de las reglas requeridas para definir una estructura de información de la organización.
- Es fácil de visualizar y recordar, ya que está basado en diagramas.
- Es útil en la comunicación entre analistas, analistas y usuario, y entre analistas y diseñadores.
- Soporta la construcción del DER durante la etapa de análisis y toda la información estará disponible para ser usada por otras herramientas.

Las entidades que serán utilizadas para ejemplificar el presente trabajo y que estarán relacionadas con el *Diagrama Jerárquico Funcional*, se podrán consultar en el **Anexo A**. En éste apartado se indican los detalles de los atributos y las relaciones de cada entidad.

CAPITULO 3.
MODELO DE FUNCIONES

El Diagrama Jerárquico de Funciones (DJF) brinda un modelo exacto de las necesidades funcionales del negocio y puede servir como base para documentar las funciones de negocio. Este modelo es de gran utilidad en las fases de análisis, diseño y construcción de la metodología Case. Para poder construir éste modelo, Designer/2000 cuenta con una herramienta llamada *Diagramador de Jerarquías de Funciones* (*Function Hierarchy Diagrammer*). En este capítulo se abordará el tema del Modelo de funciones y la forma en que Designer/2000 soporta la construcción del Diagrama Jerárquico de Funciones (DJF).

3.1. MODELO JERARQUICO DE FUNCIONES

El objetivo del *modelo* (o *diagrama*) *jerárquico de funciones* es representar lo que se hace, o necesita hacerse en el negocio, para cubrir sus objetivos. Permite mostrar los diferentes niveles de funciones del sistema en un único diagrama. Con este tipo de diagrama se puede conocer la descomposición de funciones y, por lo tanto, cuál es el nivel más bajo y cual el más alto de una serie de funciones. Debido a que está en una forma estructurada, ayuda a la comunicación y verificación del entendimiento, esencial para el efectivo desarrollo de aplicaciones.

Una vez que se han representado los datos, creando entidades y definiciones de atributos con el DER, se puede utilizar el *diagrama jerárquico de funciones* para declarar qué elementos de los datos se desean asociar con cada función A este proceso se le conoce como ***asociación entre funciones y entidades***, que no sólo incluye los elementos de los datos, sino también las acciones que se efectuarán sobre ellos (inserción, actualización, eliminación y/o consulta).
Características principales del diagrama jerárquico de funciones:

- Modelar los requerimientos funcionales del negocio (lo que hace el negocio o lo que requiere hacer en un futuro).
- Mostrar los requerimientos del negocio antes del diseño y la construcción del sistema.
- Minimiza los cambios en el proceso de desarrollo.
- Es ideal para involucrar a los usuarios antes del proceso de desarrollo.
- Está fuertemente relacionado con el modelo de datos (DER).

El *Diagrama Jerárquico de Funciones* (DJF) en las etapas de la metodología CASE:

- Análisis. Después de haber recabado toda la información posible, el analista se encuentra con una gran cantidad de notas con los requisitos de usuario. Esta información deberá ser ordenada y organizada para construir la jerarquía de funciones. Todos los requisitos de usuario tienen que estar lo suficientemente detallados para que el analista pueda construir una función que satisfaga las necesidades del usuario.
  La jerarquía de funciones se utiliza para organizar los requerimientos de usuario, esto es, una lista estructurada de todos los requisitos obtenidos de los usuarios. Un modelo funcional (DJF) está fuertemente unido con el modelo de datos (DER). El modelo entidad relación debe

contemplar la información requerida para poder realizar las funciones del negocio. Las funciones del negocio deben usar las entidades incluidas en el modelo entidad relación.
Una vez completada la jerarquía de funciones, debe resumirse y organizarse en un diagrama (DJF) más coherente, eliminar la redundancia e identificar las funciones que serán utilizadas en la construcción de la aplicación.

- Diseño. Sólo después del análisis, la persona encargada del desarrollo, sabrá exactamente lo que el sistema tiene que hacer. Durante esta etapa se toman los modelos lógicos para construir los diseños físicos, es decir, el resultado de la consulta a los usuarios sobre qué debe formar parte del nuevo sistema y qué debe permanecer como un proceso manual será reflejado en el DJF. Del cual se tomarán las funciones elementales como preparación para el diseño de los programas frontales. El resultado será el diseño de cada pantalla, o reporte, con una lista detallada de la funcionalidad que debe tener cada programa que será construido.
- Construcción. Los diseños junto con las especificaciones de cada programa son utilizados para la construcción de la aplicación. Antes de finalizar la construcción, se debe someter al sistema a una serie de pruebas y asegurarse de haber construido todas las funciones elementales del diagrama jerárquico funcional.

3.1.1. FUNCION

Una función es una actividad que es o será ejecutada, que permite cumplir la misión del negocio y lograr sus objetivos. La función representa lo que un negocio hace o necesita hacer y no el cómo lo hace.

Ejemplo:
Las funciones del negocio para una compañía de telecomunicaciones pueden ser:

- Proveer servicio telefónico de larga distancia a los clientes
- Monitorear mensualmente los cargos a clientes por servicios telefónicos

Representación.
Una función se representa dentro de una caja que contiene la definición de la función y una referencia (etiqueta) en la parte superior izquierda, como se muestra en la Figura 3.1.

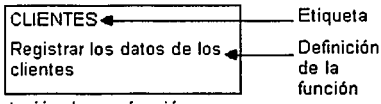

Figura 3.1. Representación de una función.

Para definir una función se deben tomar en cuenta los siguientes puntos:

- La etiqueta es utilizada para identificar la función y puede ser un nombre corto o un conjunto de números indicando la posición de la función en la jerarquía.
- La definición debe iniciar con un verbo en infinitivo y después una frase descriptiva, debe ser breve pero comprensible. Se deben utilizar verbos comprensibles para el usuario y evitar verbos ambiguos como "procesar".
- Hacer referencia a las entidades y atributos definidos en el modelo entidad relación.
- Documentar las condiciones bajo las cuales la función operará. Si las condiciones son complejas se recomienda documentar por separado.
- No hacer referencia a mecanismos, es decir, a una técnica en particular para ejecutar la función.

Mecanismo.
Es una técnica particular para implementar una función. Los mecanismos se refieren a la gente, papeles, estructuras de organización, máquinas, etc.
Los mecanismos cambian constantemente porque reflejan el uso actual de la tecnología. Debido a lo anterior es importante identificar QUÉ debe hacerse (la función), de otro modo existe el peligro de modelar "CÓMO" se hace (mecanismo). Durante la etapa de análisis se debe acordar "QUÉ" se necesita hacer para alcanzar los objetivos del negocio, durante la etapa de diseño se decidirá "CÓMO" ejecutar la función.

Durante las entrevistas, la gente tiende a decir "CÓMO" hace sus funciones, no se debe desalentar al entrevistado a hablar de esta manera, es responsabilidad del analista el descubrir las funciones que son implementadas por los mecanismos que describe.

Ejemplo:

Función:

- "Informar a los clientes las promociones vigentes".

Mecanismos:

- "Informar por correo a los clientes de las promociones vigentes".
- "Por medio de una conversación telefónica, informar a los clientes de las promociones vigentes".
- "Enviar un fax a cada cliente con las promociones vigentes".

3.1.2. DESCOMPOSICION DE FUNCIONES

Al principio de la descomposición de funciones se define una función general (el nivel mas alto) que muestre una visión global del área del negocio que se está modelando. Para el siguiente nivel de la jerarquía, la función de alto nivel es descompuesta en subfunciones estratégicas, que son realizadas para alcanzar las metas del negocio. Por ejemplo, para una tienda de videos la función de primer nivel "*DIRECCION*" podría ser descompuesta en 6 subfunciones, como se muestra en la Figura 3.2.

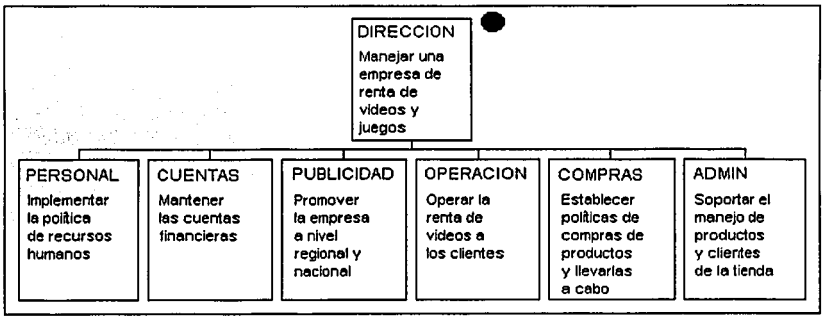

Figura 3.2. Descomposición de funciones.

La descomposición de funciones es un proceso iterativo por el cual cada función es dividida en una serie de subfunciones, que a su vez son sometidas al mismo proceso para obtener las actividades desempeñadas por cada función padre:

- La función de primer nivel de una jerarquía puede abarcar todo el negocio o una parte del negocio dependiendo del alcance del estudio que se realiza.
- Una función del negocio puede ser descompuesta en dos o más subfunciones.
- Todas las subfunciones deben ayudar en las tareas de la función.
- Cualquier función de la jerarquía debe ser descrita completamente por sus subfunciones.
- Cuando termina la descomposición de funciones, la jerarquía contiene funciones elementales o funciones de más bajo nivel, las cuales son utilizadas para realizar la asociación entre funciones y entidades.

Jerarquía de funciones.
La jerarquía de funciones, producto de la descomposición, no muestra la secuencia en que las funciones son ejecutadas, ni cuales actividades se llevan en paralelo, ni posibles iteraciones. Otras técnicas, como el modelado de procesos, pueden ser utilizadas para mostrar esta información. Una jerarquía de funciones modela las funciones del negocio. Estas funciones pueden ser implementadas por medio de procedimientos manuales y/o sistemas automatizados.

En la jerarquía de funciones existen los siguientes tipos de funciones, como se muestra en la Figura 3.3:

- Función raíz.
- Función padre.
- Función hija.
- Función elemental.
- Función atómica.
- Función común y función maestra.

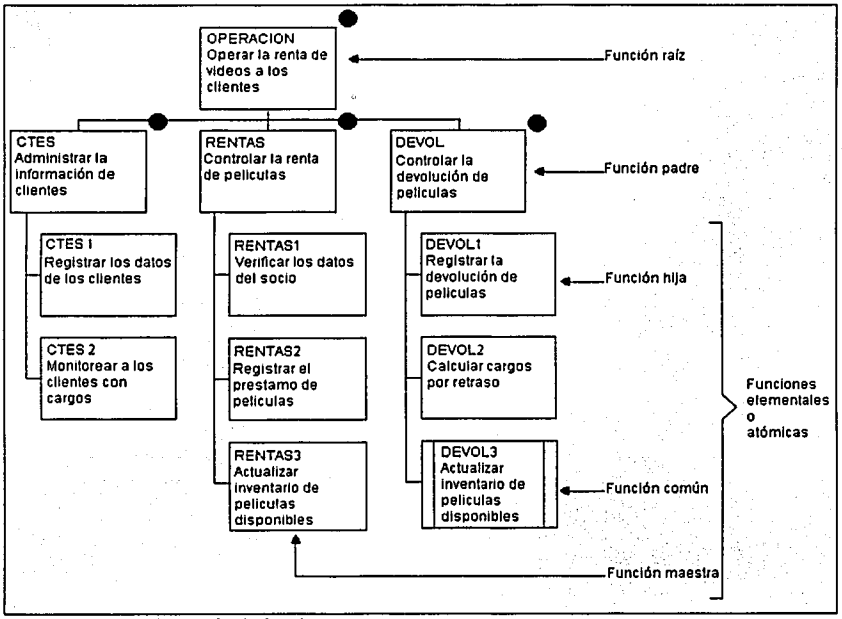

Figura 3.3. Jerarquía de funciones.

Función raíz , función padre y función hija.
La función principal de cada nivel de descomposición es mencionada como la función padre y sus componentes como funciones hijas. Una función padre podrá tener varias funciones hijas, algunas veces menos en los niveles más bajos del diagrama, pero siempre más de una. La función de nivel más alto es llamada función raíz y es una función que no tiene padre.

Funciones elementales.
Es una función que si comienza, debe ser completada exitosamente. De lo contrario, debe regresarse a su estado inicial de tal forma que los cambios no se realicen, como si nada hubiera ocurrido. No contiene pasos intermedios, por lo tanto, las funciones elementales nos indican donde detener el proceso de descomposición.

Funciones atómicas.
Es una función que no puede ser descompuesta, por lo tanto se encuentra en el último nivel del diagrama. Al terminar el análisis, el diagrama de funciones tiene en el último nivel funciones elementales o funciones atómicas.

Función común y función maestra.
Una función común es una copia de otra función conocida como maestra, que se usa cuando la misma función aparece en más de un lugar en el diagrama. Las funciones comunes se representan con una barra vertical a cada lado del rectángulo.
Cuando una función es común a otra, una de las dos es la *función maestra*, mientas que la función común que se refiere a ésta se le llama *función copia*. Cualquier cambio a la función maestra es reflejado automáticamente a todas sus copias. Una función común no puede:

- Ser cambiada (excepto borrarse).
- Tener funciones comunes que se refieran a ella.
- Ser descompuesta.

Las funciones comunes en las jerarquías no deben ser frecuentes. Aun cuando existan cientos de funciones en un diagrama jerárquico funcional, las funciones comunes deben ser raras. Un gran número de funciones comunes indica que el análisis está incompleto.

3.2. DIAGRAMADOR JERARQUICO FUNCIONAL

Este es el tercer diagramador que ofrece Designer/2000 para modelar funciones o procesos (términos que Designer/2000 considera sinónimos), el cual permite crear jerarquías de todas aquellas funciones definidas por el negocio e identificar las que pueden ser automatizadas.
El objetivo del Diagramador de jerarquías de funciones es mostrar las funciones del sistema de un modo jerárquico o multinivel. Las funciones son las mismas que los pasos de los procesos del Modelador de procesos o las funciones del Diagramador de flujo de datos, pero el diagrama (en una de sus visualizaciones) tienen más el aspecto de un típico cuadro organizativo[3.1].
Sus características:

- Permite crear y mantener la definición de funciones almacenadas en el repositorio multiusuario.
- Modificar la presentación (layout) de las jerarquías permitiendo trabajar a cualquier nivel de detalle, es decir, colapsar y expandir cualquier nivel de alguna jerarquía.
- Elegir la orientación vertical, horizontal o híbrida para los diagramas.
- Obtener una derivación automática de la matriz de funciones contra entidades.
- Permite crear tantas jerarquías de funciones como se necesiten en el modelado.
- Facilidad para descomponer las funciones del negocio y definir las funciones elementales.
- Permite definir el tipo de información que será usada, asociando entidades y atributos a cada función que será automatizada.
- Facilita el acceso al ***Transformador para el diseño de aplicaciones*** (*Application Design Transforme*) para obtener el diseño de módulos a partir de un modelo de funciones.

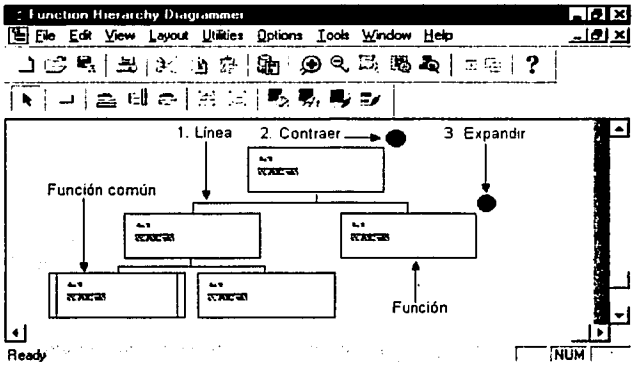

Figura 3.4. Objetos del diagramador jerárquico funcional.

Sus objetos.
Con este diagramador sólo puede dibujarse un objeto: ***Funciones***. Pero cuenta con otros elementos como se muestra en la Figura 3.4:

[3.1] Dorsey, Paul y Peter Koletzke. *Manual de Oracle Designer/2000*, p. 246.

1. Las líneas entre las funciones no representan flujos, sino relaciones padre - hijo.
2. Icono para contraer (ocultar) las funciones hijas.
3. Icono para expandir (mostrar) las funciones hijas.

Hay tres formas de visualizar el diagrama jerárquico funcional: horizontal (es el formato por defecto), vertical e híbrido. El formato híbrido es una mezcla de los dos anteriores, muestra las funciones "padre" con formato horizontal y las funciones "hijo" con formato vertical.

Barra de herramientas (Tool Palette).
La barra de herramientas cuenta con los siguientes botones, Figura 3.5:

1. Function. Botón para crear nuevas funciones.
2. Horizontal Layout. Presentación del diagrama en forma horizontal.
3. Vertical Layout. Presentación del diagrama en forma vertical.
4. Hybrid Layout. Presentación del diagrama en forma horizontal y vertical.

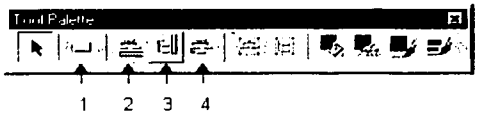

Figura 3.5. Barra de herramientas (Tool Palette) del diagramador jerárquico funcional.

3.2.1. DIBUJANDO LOS OBJETOS DEL DIAGRAMADOR JERARQUICO FUNCIONAL

Este diagramador facilita la construcción del modelo jerárquico funcional y permite asociar las funciones con las entidades y atributos del DER. La información es almacenada en el repositorio y estará disponible para ser usada por el *Transformador para el diseño de aplicaciones.*

Crear un Diagrama.
La creación de un DJF se podrá hacer con el menú "*File\New*" o presionando el botón con el icono de la hoja blanca que se encuentra en la barra de herramientas. La Figura 3.6 corresponde a la ventana (New Diagram) que se mostrará al momento de crear un nuevo diagrama. Para crear un DJF, éste debe estar basado, por lo menos, en una función raíz y se podrá seleccionar alguna de la lista de funciones o crear una nueva con el botón *New Function*. Para seleccionar una función raíz existente, la ventana "New Diagram" cuenta con:

1. Un campo para insertar un criterio de búsqueda de alguna función raíz, en Designer/2000 se conoce como *función root.*
2. Un botón para realizar la búsqueda.
3. Una lista con todas las funciones disponibles para crear un nuevo DJF o una lista con el resultado de alguna búsqueda.

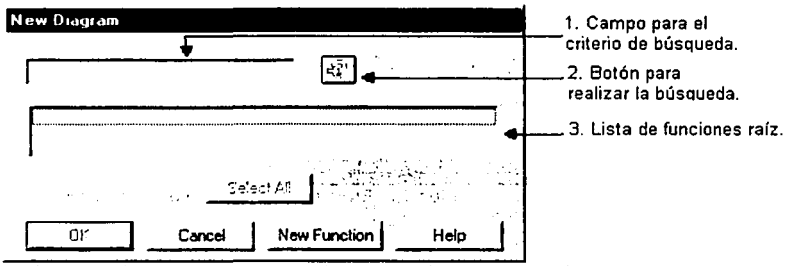

Figura 3.6. Ventana para crear un diagrama jerárquico funcional.

Ejemplo:
Para crear un nuevo diagrama llamado "DFJ_AFILIACION", basado en una nueva función raíz (o función root) con etiqueta "AFILIACION", se podrá hacer siguiendo estos pasos:

1. Seleccionar la opción para crear un nuevo diagrama.
2. Debido a que un DJF debe tener una función raíz, presionar el botón *New Function* para crear una primera función raíz y se mostrará la ventana de la Figura 3.7
3. Insertar la etiqueta (label) de la función: "AFILIACION". Y una breve descripción de la misma (Short Definition): "Gestionar las solicitudes de afiliación".
4. Presionar el botón "*OK*" para crear la función raíz.

5. Hasta ahora sólo se ha creado la función raíz, pero es necesario guardar el diagrama con el botón *"Save"* o con el menú *"File\Save Diagram"*, insertar el nombre del diagrama "DJF_AFILIACION" y presionar el botón *"OK"*.

En caso de existir funciones raíz, se podrá seleccionar alguna de la lista que aparece en la ventana *New Diagram*, como base para un nuevo diagrama. Pero es importante recordar que aún cuando se trate de un diagrama diferente y se utilicen funciones de otro diagrama, cualquier cambio que se realice sobre las funciones se verán afectados los demás diagramas.
Para abrir un diagrama podrá utilizar el menú *"File\Open"* o presionar el botón con el icono del folder. Después de crear o abrir un diagrama, aparecerá la barra de herramientas (Tool Palette) y el área de trabajo lista para crear, modificar, incluir o borrar cualquier componente del DJF.

Crear una Función.
Para crear una función, pulsar el botón ***Function*** (del tool palette), después pulsar sobre la función que vaya a ser "padre" de la nueva función o sobre el área de trabajo, y se mostrará la ventana de la Figura 3.7 para insertar los datos de la etiqueta (label) y la definición de la función (Short Definition).

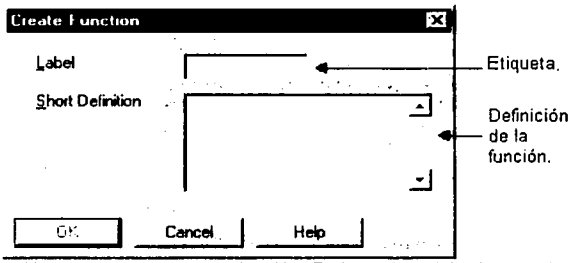

Figura 3.7. Ventana para crear una función. En la cual se debe insertar la etiqueta y la definición de la nueva función.

Al crear una función sobre otra, se estará indicando directamente la jerarquía de la nueva función, que estará un nivel abajo de la función sobre la cual se está creando (función padre). En caso de crear la función sobre el área de trabajo, ésta no tendrá padre (Parent). Posteriormente se podrá mover sobre cualquier función, sin importar el nivel, quedará exactamente un nivel abajo de la función seleccionada.
Las funciones podrán quedar en cualquier nivel deseado, sin importar el orden en que hayan sido creadas, incluso podrán quedar en el nivel más alto o el más bajo. Sólo basta con seleccionar la función con el ratón, arrastrarla y soltarla sobre otra función.

Ejemplo:
Para el diagrama jerárquico funcional "DJF_AFILIACION", se crearon otras 2 funciones con las siguientes características, como se muestra en la Figura 3.8:

1. La primera con la etiqueta "CATALOGOS" y con la siguiente descripción: "Mantener los catálogos". La cual se encuentra bajo la función raíz "AFILIACION".
2. La segunda con la etiqueta "AFI_C001" y con la siguiente descripción: "Mantener los catálogos de países, estados y municipios". Esta se encuentra bajo la función "CATALOGOS".

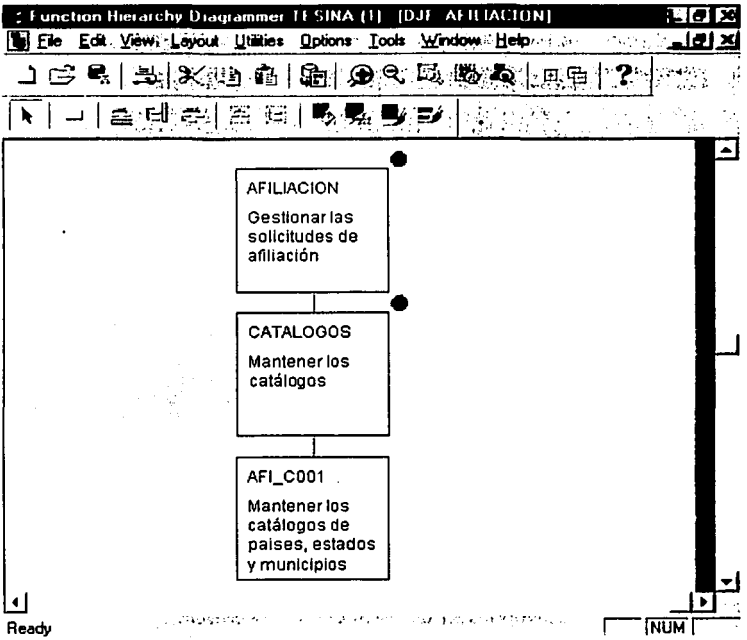

Figura 3.8. Sesión del diagrama jerárquico funcional "DJF_AFILIACION". En este diagrama la función raíz o función root es "AFILIACION".

3.2.2. PROPIEDADES DE LA FUNCION

Para ver las propiedades de una función se podrá utilizar el menú "*Edit\Properties*" o dando dos click sobre la función y se mostrará la ventana ***Edit Function***, la cual cuenta con 6 carpetas que podrán ser utilizadas para completar el análisis y documentar las "*funciones elementales*" o "*atómicas*", ver la Figura 3.9.

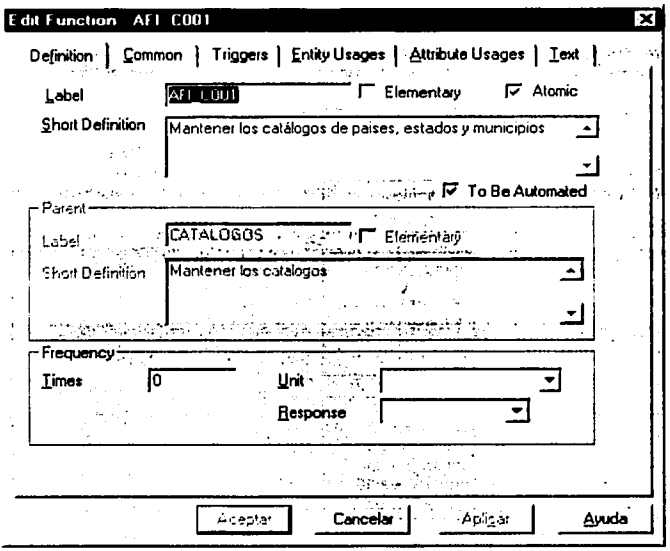

Figura 3.9. Ventana de propiedades de una función (Edit Function).

Carpeta Definition.
Contiene los siguientes campos:

- Label. Es la referencia o etiqueta de la función, puede estar compuesta por números, letras o ambos. Debe ser significativo y fácil de recordar, por ejemplo, la etiqueta "ADM1.3" sugiere que la función padre es "ADM1" y que a su vez ésta está bajo "ADM" cuya definición podría comenzar con "Administrar". Este campo es obligatorio y con un máximo de 30 caracteres.
- Elementary. Se utiliza para indicar si es, o no, una función elemental para el negocio.
- Atomic. Esta opción es sólo de consulta e indica si es, o no, una función atómica. Su estado cambiará cuando: 1) Si la función está en el último nivel, la opción estará marcada; 2) Si tiene funciones hijas, la opción no estará marcada.

- Short Definition. Corresponde a la definición de la función y es recomendable utilizar nombres con referencia al modelo entidad relación. Es obligatoria, con un máximo de 240 caracteres. Se debe insertar cuando la función es creada y se podrá modificar en la ventana de propiedades "*Edit Function*", al igual que la etiqueta.
- To Be Automated. Es utilizado para indicar que la función será automatizada, cuando la opción está marcada. Esta propiedad es sólo con propósito informativo.
- Parent. En caso de ser una función hija, éste grupo desplegará los datos de la función que está un nivel mas arriba, es decir, la etiqueta y la definición de la función padre.
- Frequency. Los datos para este bloque son opcionales.
  La opción "*Times*" es para especificar que tan a menudo se ejecuta, o se ejecutará, la función dentro de un periodo de tiempo, que se especifica en la opción *Unit*.
  La opción "*Unit*" cuenta con una lista para seleccionar el periodo de tiempo: segundo, minuto, hora, día, semana, mensual, trimestre, semestral o por año.
  La opción "*Response*" muestra una lista con dos opciones: *Overnight* (nocturno), para indicar que las funciones serán convertidas durante el proceso de desarrollo en módulos de "utilidades" (Utility), por ejemplo, en procedimientos almacenados; *Inmediate*, para indicar que las funciones serán convertidas durante el proceso de desarrollo en módulos de programas frontales (pantallas) o reportes.

Carpeta Common.
Los datos en esta carpeta son opcionales y están habilitados para ser usados cuando se crea una función común (common), es decir, para seleccionar a la función maestra. Cuenta con dos opciones:

- Application System drop-down list. Cuando existe más de un *sistema de aplicaciones* (application system), podrá seleccionar el nombre y versión del sistema de aplicaciones que es dueño de la función maestra.
- Master Function drop-down list. Muestra una lista de todas las funciones disponibles que podrán ser utilizadas como función maestra. La función común (common) asumirá todos los detalles de la función maestra y se verán reflejados al momento de generar el módulo, los cambios que se realicen sobre la función común no serán tomados en cuenta.

Carpeta Triggers.
Esta carpeta está habilitada para especificar los eventos de la función y es opcional. Un evento, es un suceso que actúa como disparador (trigger) de una o más funciones, un evento puede ser el alcanzar un punto significativo en el tiempo, un cambio de estado, o la ocurrencia de algo externo que causa que el negocio reaccione.

Carpeta Entity Usages.
En esta carpeta, se podrá realizar la asociación entre la función y las entidades, ver Figura 3.10:

- Entity. Muestra una lista de las entidades disponibles para seleccionar la que será usada por la función. Una vez seleccionada una entidad, ésta no será mostrada en la siguiente lista al ser llamada para insertar otra entidad en la misma función.
- Create, Retrieve, Update, Delete, Archive y Other. Estas opciones son utilizadas para indicar las operaciones que podrán ser realizadas sobre la entidad, por ejemplo, cuando la opción "Retrieve" está marcada indica que podrá recuperar información de la entidad seleccionada. La opción de "otros", está disponible para indicar usos definidos por el usuario.
- Comments. Para insertar comentarios acerca de la entidad, es opcional con un máximo de 240 caracteres.

- Botón "Insert". Insertar un renglón para agregar otra entidad que será usada por la función.
- Botón "Delete". Borrar el renglón seleccionado.
- Botón "Select All". Para seleccionar todas las entidades disponibles e incluirlas en la función.
- Botón "Reset Default". Regresa la apariencia original de la carpeta, por ejemplo, si se modificó el tamaño de las columnas para visualizar mejor las descripciones, al presionar este botón, regresarán a su tamaño original.

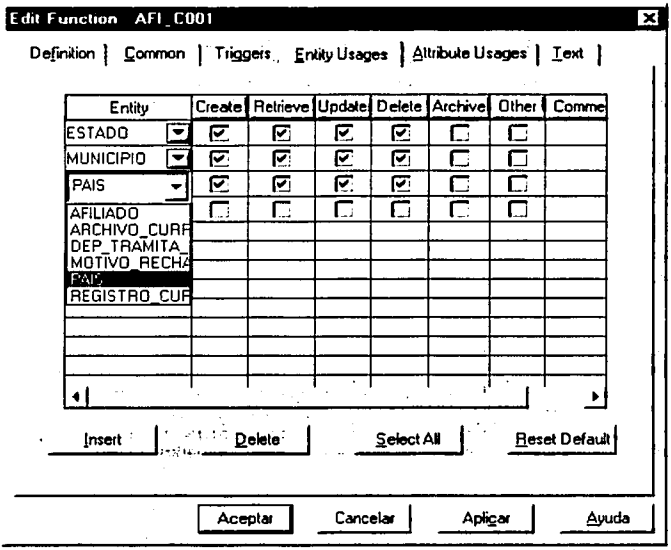

Figura 3.10. Carpeta Entity Usages. Para realizar la asociación entre la función y las entidades.

Carpeta Attribute Usages.
En esta carpeta se podrá realizar la asociación entre la función y los atributos de cada una de las entidades seleccionadas en la carpeta *Entity Usages*, ver Figura 3.11:

- Entity. Muestra el nombre de la entidad, de la cual se escogerán los atributos que serán usados por la función. En ésta lista, también podrá seleccionar las entidades que usará la función, una a la vez, o cambiar a otra entidad previamente seleccionada sin necesidad de ir a la carpeta "*Entity Usages*".
- Attribute. Lista todos los atributos de la entidad para seleccionar los que serán usados por la función.
- Insert, Retrieve, Update, Nullify, Archive y Other. Estas opciones se utilizan para indicar las operaciones permitidas para cada uno de los atributos, por ejemplo, "*Nullify*" es para indicar si aceptará nulos.

- Comments. Es opcional, con un máximo de 240 caracteres.
- Botón Insert. Inserta un renglón para seleccionar un atributo mas.
- Botón Delete. Borrar un renglón.
- Botón Select All. Seleccionar todos los atributos de la lista.
- Botón Reset Default. Regresa la apariencia original de la carpeta.

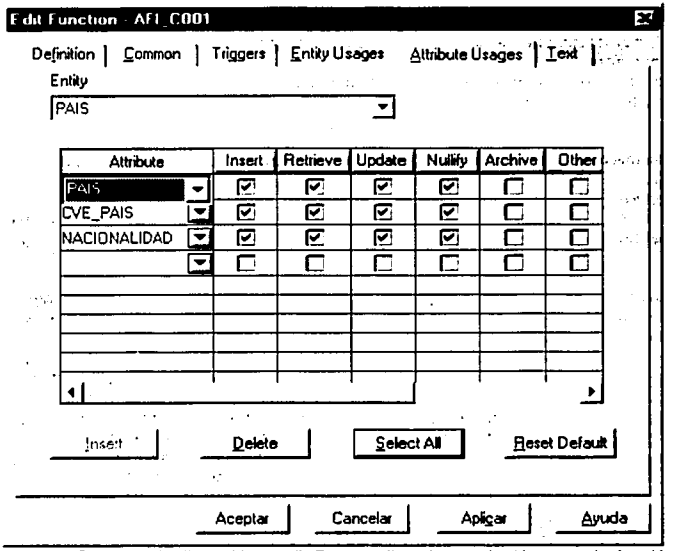

Figura 3.11. Carpeta "Attribute Usages". Para realizar la asociación entre la función y los atributos de cada entidad.

Carpeta Text.
Cuando es necesario agregar una descripción detallada para documentar las reglas y condiciones en que deberá operar la función, en ésta carpeta se podrá realizar y es opcional.

Adicionalmente, en cada una de las carpetas se encuentran los siguientes botones:

- OK button. Salvar los cambios realizados y cerrar la ventana.
- Cancel button. Cancelar, y cerrar la ventana.
- Apply button. Salvar los cambios realizados sin cerrar la ventana.
- Help button. Despliega la ayuda relacionada a la carpeta actual.

Ejemplo:

1. La Figura 3.10 muestra la carpeta ***Entity Usages***, en la cual se realiza la asociación entre la función "AFI_C001" y las entidades "PAIS", "ESTADO" y "MUNICIPIO". Las operaciones permitidas para estas entidades son: crear (create), recuperar (retrieve), actualizar (update) y borrar (delete).
2. La Figura 3.11 muestra la carpeta ***Attribute Usages***, en la cual se realiza la asignación de todos los atributos de la entidad "PAIS" a la función "AFI_C001", con opción a realizar las operaciones de insertar (Insert), recuperar (Retrieve) y actualizar (Update).
   En el punto anterior se mencionó que las operaciones permitidas para la entidad "PAIS" son: crear, recuperar, actualizar y borrar. Al presionar el botón "*Select All*" de la carpeta "*Attribute Usages*", automáticamente se muestran todos los atributos de la entidad con las operaciones antes mencionadas.
   La asignación de los atributos para las entidades de "ESTADO" y "MUNICIPIO", se realiza de la misma forma.

Toda la información capturada en la ventana de propiedades de la función (Edit Function) será almacenada en el repositorio y estará disponible para ser tomada por el ***Transformador para el diseño de aplicaciones*** (*Application Design Transformer*). Esta herramienta toma los detalles de la asociación entre la función y las entidades para generar el diseño de módulos, este tema es tratado en el capitulo 5.

El *diagrama jerárquico funcional* (DJF) y las funciones que serán utilizadas en el "*Transformador para el diseño de aplicaciones*" para generar el diseño de módulos y que servirán como ejemplo en el presente trabajo, se podrán consultar en el **Anexo B**.

CAPITULO 4.
DISEÑO DE TABLAS

Como parte del ciclo de vida del desarrollo de sistemas, se encuentra el diseño y construcción de la base de datos. En la etapa de diseño se toma toda la información del DER lógico para realizar el diseño de la base de datos y finalmente construir la base de datos física. En este capítulo se muestran las herramientas de Designer/2000 que soportan el diseño y construcción de la base de datos: Transformador para el diseño de la base de datos (*Database Design Transformer*), Modelo del servidor (*Server Model*) y el generador de la BD (*Generate Database from Server Model*).

4.1. TRANSFORMADOR PARA EL DISEÑO DE LA BASE DE DATOS

El *Transformador para el diseño de la base de datos* (*Database Design Transformer*) podrá ser utilizado durante la etapa de diseño para crear y actualizar el diseño preliminar de la BD, tomando la información del diagrama entidad relación. Esta herramienta transforma las entidades en tablas, los atributos en columnas y las relaciones entre las entidades en validaciones (constraints) de llaves primarias y foráneas. Así como el modelo de entidades, el diseño de la BD que es generado por el *Transformador para el diseño de la base de datos* es almacenado en el repositorio. Esta herramienta permite:

- Controlar el alcance del proceso para generar el diseño preliminar de la BD, es decir, controlar los tipos de elementos que el transformador deberá crear o modificar.
- Generar todo el diseño de la BD de un solo paso o progresivamente refinando el modelo y agregándole detalles.
- Visualizar los detalles del *mapeo* y los tipos de elementos que el transformador deberá crear o modificar.

Para generar el diseño preliminar de la base de datos a partir del DER, también llamado *mapeo* de entidades, se podrá realizar de dos formas:

1. Desde el *Entity Relationship Diagrammer*. Primero seleccionando las entidades a transformar (mapear) y con el menú "*Utilities\Database Design Transformer...*". Se mostrará la ventana del transformador lista para ser ejecutada con las entidades seleccionadas.
2. Presionar el botón "*Database Design Transformer*" que se encuentra en el menú principal de Designer/2000. Se mostrará la ventana del transformador, lista para hacer la selección de las entidades a transformar.

La ventana del transformador cuenta con 4 carpetas, como se muestra en la Figura 4.1:

1. Mode.
2. Table Mappings.
3. Other Mappings.
4. Run Options.

Carpeta Mode.
La primer carpeta cuenta con 2 opciones para realizar el mapeo de las entidades:

1. Con la opción "*Run the Transformer in Default Mode*", se podrá realizar el mapeo de todas las entidades o de las entidades previamente seleccionadas,
2. Al seleccionar la opción "*Customize the Database Design Transformer*", automáticamente se habilitaran las demás carpetas para realizar el mapeo manualmente de acuerdo a las preferencias que se le indiquen.

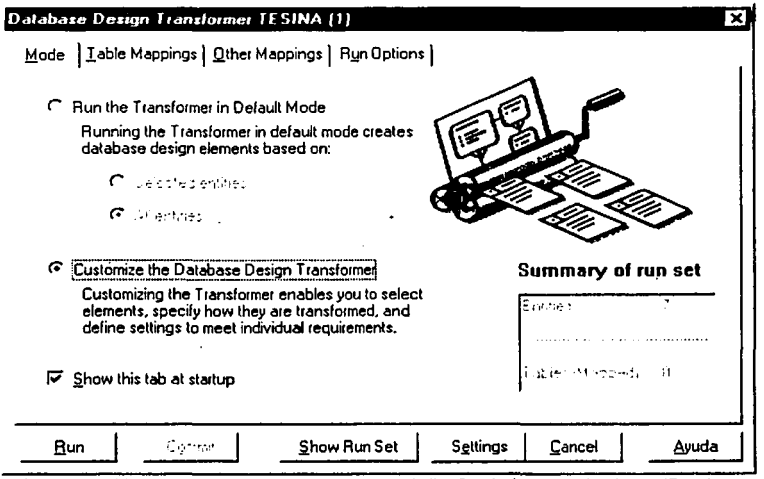

Figura 4.1. Ventana del Transformador para el diseño de la base de datos (Database Design Transformer).

Carpeta Table Mappings.
En esta carpeta se realiza la selección de las entidades que serán mapeadas. La selección se podrá realizar de la siguiente forma: marcando la opción de *In Set*; eligiendo cada entidad de manera individual; o bien, incluyendo todas las entidades. Para seleccionar todas las entidades, se utiliza el botón derecho del ratón para mostrar un menú del cual se deberá seleccionar la opción *Include All*.
Si una entidad no ha sido mapeada a tabla (*No Mapping*), entonces se creará una tabla nueva. Cuando una entidad ya fue mapeada (*Mapped*), entonces se podrá modificar la tabla correspondiente o dejar como está. Las opciones que se encuentran dentro de ésta carpeta son:

- In Set. Indicador del conjunto de entidades a mapear. Cuando se elige una entidad que no ha sido mapeada, automáticamente se incluyen todos sus atributos.
- Entity. Se listan cada una de las entidades de la aplicación. En el caso de los subtipos, se muestran con una identación debajo de la entidad supertipo.
- Map Type. Indica el estado de la entidad, es decir, si ya fue mapeada o no.

- Arc. Sólo se usa para las entidades subtipos que han sido mapeadas. El transformador crea un mapeo de estilo "*arc*" para las entidades subtipo, es decir, se crea una tabla separada de su supertipo heredando los atributos del supertipo y ligando las tablas por medio de una llave foránea.
- Table. Muestra el nombre de la tabla correspondiente, cuando una entidad ha sido mapeada.

Carpeta Other Mappings.
En ésta carpeta podrá modificar el mapeo para una entidad que ya ha sido mapeada. Se podrán crear/eliminar columnas, llaves primarias, llaves foráneas, índices, o simplemente ver los componentes de la entidad mapeada.

Carpeta Run Options.
En ésta carpeta se indica el tipo de elemento que se creará y/o modificará durante el mapeo, el cual puede realizarse de diferentes maneras:

- Generar un diseño de BD completo incluyendo tablas, columnas, llaves e índices en una sola ejecución del transformador. Esto se logra eligiendo la opción *create* en todas las opciones. Se recomienda su uso cuando todos los objetos se generarán por primera vez.
- Generar el diseño por partes, corriendo varias veces el transformador. Primero se generarían las tablas, luego las columnas y por último las llaves e índices. Es importante verificar las dependencias durante la generación.
- Modificar las propiedades de los objetos creados. Por ejemplo, adicionar nuevas columnas a las tablas, una vez definidos los nuevos atributos en las entidades correspondientes. Cuando se desea realizar alguna modificación, se debe especificar qué tipos de elementos se desean modificar: Tables (tablas), Columns (columnas), Keys (llaves primarias y foráneas) e Indexes (índices). Cuando se marca alguna opción para modificar, se habilitarán otras propiedades para especificar las que serán modificadas, tales como: *Name*, el nombre de las propiedades; *text*, el texto de las propiedades; *Sequencing*, la secuencia de las propiedades; *Volumetrics*, las propiedades de volumen; *Datatype*, el tipo de dato; y *Display*, las propiedades de despliegue de los elementos.

Respecto a los botones que se encuentran en la parte de abajo:

- Run. Botón para ejecutar el mapeo.
- Commit. Para elegir la frecuencia con que se salvarán los elementos en el repositorio durante la generación.
- Show Run Set. Muestra una lista de las entidades y todos sus componentes que serán mapeados.
- Settings. Este botón mostrará otra ventana con tres carpetas: Carpeta "*Database*", cuenta con la opción para especificar sobre que base de datos y usuario se podría crear una copia de las tablas seleccionadas; Carpeta "*Settings-Keys*", para especificar las reglas de borrado o de actualización para las llaves foráneas; así como, decidir el nivel en el que se realizarán las validaciones (constraints): validar únicamente en el servidor, validar únicamente en el cliente o en ambos. En la carpeta "*Settings-Other Settings*", existe la opción para especificar el prefijo que será usado cuando se construyan los nombres para las tablas; así como, la opción de poner como prefijo el nombre corto de la tabla en el nombre de las columnas.
- Cancel. Cancelar y cerrar la ventana.
- Help. Mostrar la ayuda.

Ejemplo:
Para realizar el mapeo de las entidades del *Anexo A* (capitulo 2) se siguieron estos pasos:

1. Presionar el botón "*Database Design Transformer*" (menu principal de Designer/2000).
2. De la carpeta *Mode* se leccionar la opción "*Customize the Database Design Transformer*".
3. En la carpeta "*Table Mappings*" seleccionar todas las entidades. Cuando es un gran número de entidades es recomendable utilizar el botón derecho del ratón y se mostrará un menú, para seleccionar la opción "*Include All*", como se muestra en la Figura 4.2.

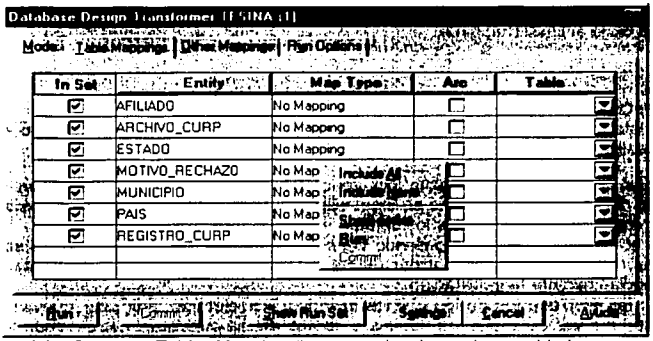

Figura 4.2. Carpeta "Table Mappings" para seleccionar las entidades que serán "mapeadas". Además se muestra el menú con la opción para seleccionar todas las entidades.

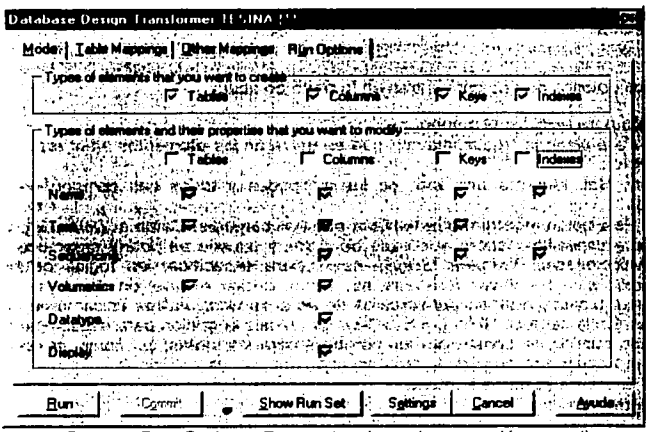

Figura 4.3. Carpeta Run Options. Para seleccionar la operación a realizar: crear o modificar.

4. Carpeta *Run Options*. Verificar que todas las opciones de "*Types of elements that you want to create*" estén marcadas, como se muestra en la Figura 4.3.
5. Presionar el botón *Settings*. Dejar por defecto las opciones de la carpeta *Settings-Keys*: reglas de borrado, de actualización y el nivel de validación para los constraints (validar en el servidor).
6. En la carpeta "*Settings-Other Settings*" marcar la opción "*Foreign Key columns*" para indicar que el nombre de las llaves foráneas tendrán prefijo; y en "*Table prefix*" poner como prefijo "**AFI_**" para el nombre de las tablas, como se muestra en la Figura 4.4. Presionar el botón para aceptar, se mostrará una ventana de información y nuevamente dar aceptar.

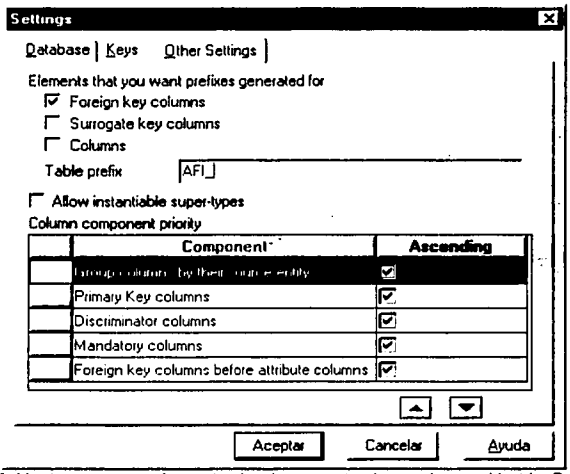

Figura 4.4. Ventana que será mostrada al momento de presionar el botón Settings.

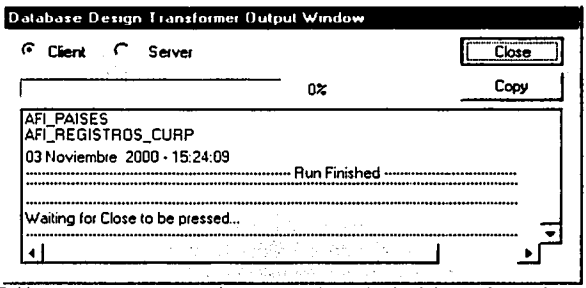

Figura 4.5. Ventana que muestra el avance y el resultado del transformador.

7. Para iniciar el mapeo, presionar el botón "*Run*" y se visualizará una ventana mostrando la salida del proceso. Al terminar presionar el botón "*Close*", ver Figura 4.5. La columna "*Map Type*", de la carpeta "*Table Mappings*", mostrará la palabra *Mapped* y en la columna *Table* se mostrará el nombre de cada tabla.
8. Por último presionar el botón *Close* para cerrar la ventana del transformador.

Después de realizar el mapeo de entidades, se podrá ver el diseño preliminar de la base de datos con la herramienta *Modelo del servidor* (*Server Model - Navigator*) del *Editor de diseño*.

4.2. MODELO DEL SERVIDOR

La herramienta *Modelo del Servidor* (*Server Model - Navigator*) se encuentra en el *Editor de diseño* (*Desig Editor*). Con esta herramienta podrá visualizar el diseño preliminar de la BD generado por el *Transformador para el diseño de la base de datos* y realizar las modificaciones necesarias para construir el diseño final que será utilizado para construir los objetos de la BD. El diseño de las tablas se podrá visualizar de dos formas, como se muestra en la Figura 4.6:

- Con el navegador (*Server Model - Navigator*), que tiene una estructura en forma de árbol.
- Con el diagramador (*Server Model Diagram*), que corresponde al DER de diseño.

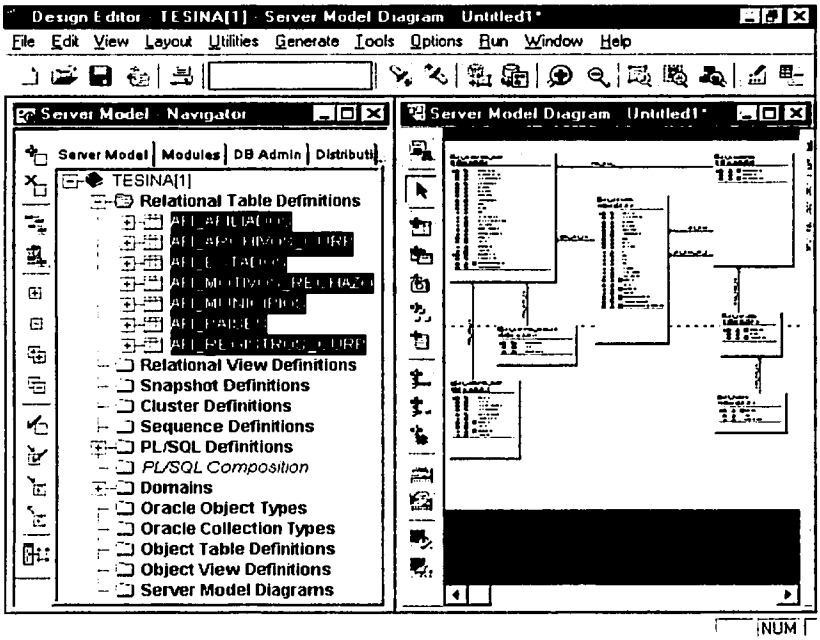

Figura 4.6. Sesión del *Modelo del Servidor* (*Server Model - Navigator*). En la ventana izquierda se encuentra el navegador con la estructura de las tablas en forma de árbol. Y del lado derecho, se encuentra el diagrama correspondiente.

Para ver el diagrama de diseño (*Server Model Diagram*) sólo basta con seleccionar las tablas, con el puntero del ratón arrastrarlas al área de trabajo y finalmente soltar la selección; o bien,

seleccionar las tablas y utilizar la opción del menú "*File\New\Server Model Diagram*". Ya sea con el navegador o en la ventana del diagrama, es posible crear, borrar o modificar el diseño de la BD.

Barra de Herramientas (Toolbar).
La barra de herramientas disponible para el *Modelo del Servidor*, tiene algunos botones adicionales a los mencionados en el *Capitulo 1* y son los siguientes:

Muestra la ventana de propiedades para el objeto seleccionado.

Cambiar el formato de la ventana de propiedades:

1. Editar todas las propiedades del objeto.
2. Editar las propiedades más comunes del objeto.

Botón *Generate DLL*. Llamar la ventana para generar el código (SQL DLL) de los objetos que serán creados en la base de datos y que están definidos en el *Modelo del Servidor*.

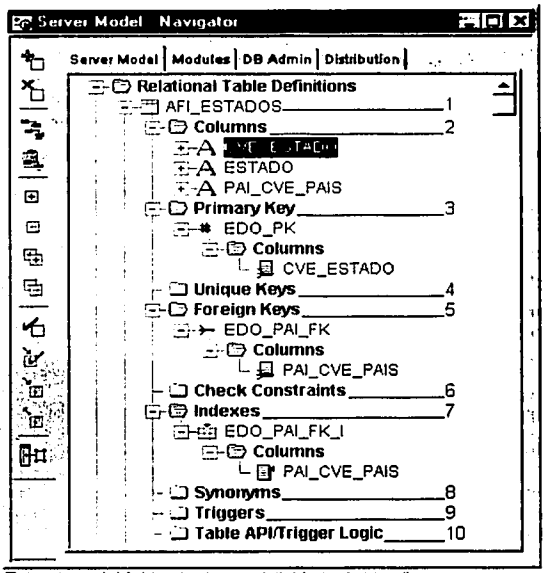

Figura 4.7. Estructura del folder "*Relational Table Definitions*".

Navegador – Modelo del Servidor.
El *Navegador - Modelo del Servidor* (*Server Model - Navigator*) muestra los objetos disponibles para el diseño de la base de datos y tiene una estructura en forma de árbol. En el primer nivel del árbol se encuentra el nombre de la aplicación (*Application System*) y en el segundo nivel del árbol se encuentra, entre otros, la definición de las tablas (*Relational Table Definitions*).

Estructura del folder "*Relational Table Definitions*".
Esta carpeta tiene la siguiente estructura, como se muestra en la Figura 4.7:

1. Tables. Contiene la definición de todas las tablas almacenadas en el repositorio.
2. Columns. Muestra las columnas de la tabla.
3. Primary Key (PK). Nombre de la llave primaria y las columnas que son parte de ella.
4. Unique Keys. Contiene la definición de las columnas únicas. Cuando se define una columna como única, ésta no podrá contener valores duplicados.
5. Foreign Keys (FK). Agrupa las columnas que son llaves foráneas. Por ejemplo, la tabla "AFI_ESTADOS" tiene como FK la columna "PAI_CVE_PAIS", la cual a su vez, es la llave primaria (PK) de la tabla "AFI_PAISES".
6. Check Constraints. Para crear las validaciones (constraints) de los valores permitidos para alguna columna. Por ejemplo, en una columna utilizada para almacenar los valores posibles de "sexo", el *Check Constraints* podría ser: *validar que la columna sólo acepte como valores la "F" para femenino o "M" para masculino'.*
7. Indexes. Utilizado para definir índices. Del ejemplo anterior, mapeo de entidades, el generador ha creado un índice para la columna "PAI_CVE_PAIS" (FK) de la tabla "AFI_ESTADOS".
8. Synonyms. Esta carpeta muestra los sinónimos existentes. Un sinónimo es un alias que es usado para identificar los objetos de la BD, en una BD. Una ventaja de usar sinónimos es, por ejemplo, si el nombre de un objeto bajo la BD cambia, sólo se necesita redefinir el sinónimo y no todos los objetos que hacen referencia a éste.
9. Triggers. Para definir los disparadores (triggers) adicionales a la tabla.
10. Table API/Trigger Logic. Definir procedimientos para realizar validaciones adicionales sobre la operación de la tabla. Por ejemplo, llevar la auditoria sobre el borrado o actualización de los registros de una tabla.

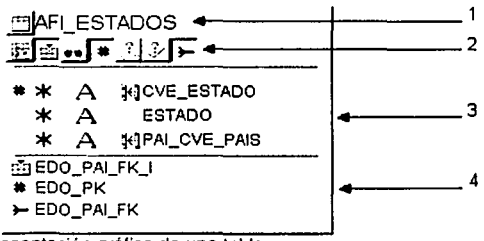

Figura 4.8. Representación gráfica de una tabla.

Tabla.
En el diagrama "*Server Model Diagram*", las tablas están formadas de la siguiente manera, como se muestra en la Figura 4.8:

1. Nombre de la tabla.
2. Botones para mostrar los objetos adicionales a la tabla.
3. En esta sección se encuentran las columnas de la tabla. El símbolo "#" es para identificar los campos que forman parte de la llave primaria (PK), el símbolo "*" es para identificar los campos que son obligatorios y el símbolo "o" es para identificar los campos que son opcionales.
4. Sección para mostrar los objetos adicionales a la tabla, como los índices, constraints, etc.

Propiedades de los objetos.
En la ventana de propiedades se podrá modificar la definición de los objetos. Para editar la ventana de propiedades, se podrá hacer de 3 formas:

1. Seleccionado el objeto y con la opción del menú "*Edit/Properties...*".
2. Pulsar dos veces sobre el objeto.
3. Con el botón de propiedades de la barra de herramientas.

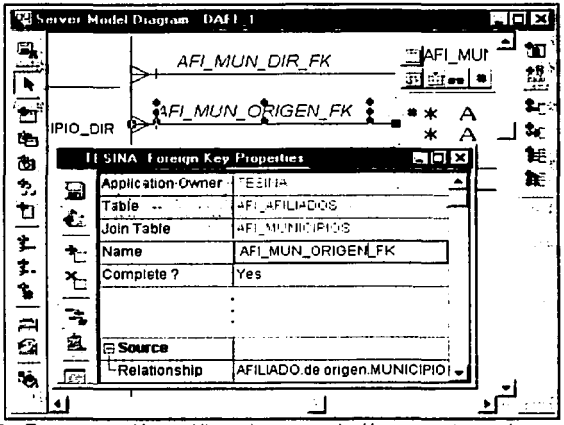

Figura 4.9. Representación gráfica de una relación y ventana de propiedades correspondiente a la relación.

Ejemplo:
Antes de crear los objetos en la BD es importante revisar el diseño final, ya que es la representación gráfica de como se crearán los objetos en la BD. Del diseño preliminar generado en el ejemplo anterior, mapeo de entidades, se podrían realizar las siguientes modificaciones.
Entre la tabla "AFI_AFILIADOS" y la tabla "AFI_MUNICIPIOS" existen dos relaciones: Una para "Origen" (lugar de nacimiento) y otra para "Dirección". Seria conveniente cambiar el nombre, que el generador pone por defecto a las columnas (FK) y las relaciones involucradas entre estas dos tablas, para identificarlas más fácilmente:

1. Para "Origen" (lugar de nacimiento). Editar la ventana de propiedades de la relación, para identificarla, en la propiedad *Relationship* debe aparecer el nombre para éste extremo de la relación, que en este caso es "de origen". En la propiedad *Name* insertar "AFI_MUN_ORIGEN_FK" como se muestra en la Figura 4.9.
   Después, editar las propiedades de la(s) columna(s) involucrada(s) en la relación y en la propiedad *Name* insertar "MUN_CVE_MUNICIPIO_ORI" como se muestra en la Figura 4.10.
2. Para "Dirección". La modificación se podrá realizar de la misma forma en que se hizo para "Origen", sólo que para el nombre de la columna insertar "MUN_CVE_MUNICIPIO_DIR" y para el nombre de la relación insertar "AFI_MUN_DIR_FK".

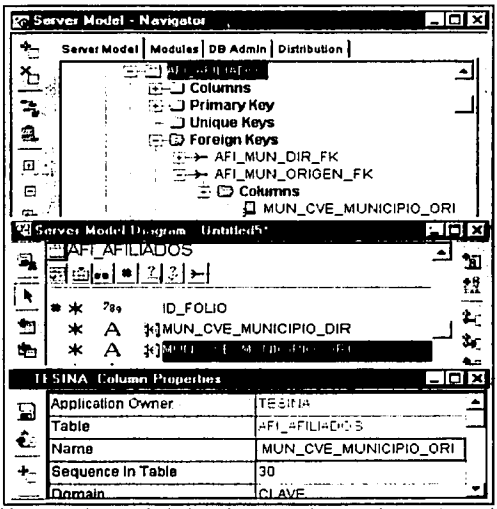

Figura 4.10. Ventana de propiedades de una columna. La ventana de propiedades podrá ser editada desde el navegador o desde el diagrama.

4.3. GENERADOR DE ARCHIVOS DLL

El *Editor de diseño* (Design Editor) cuenta con una herramienta llamada "*Generate Database from Server Model*" para generar los archivos DDL (Data Definition Languaje). Esta herramienta es de gran utilidad durante la etapa de construcción ya que genera los archivos con el código (DDL) para crear los objetos de la BD, totalmente libre de errores y listos para ser ejecutados.
Una vez que se ha depurado el diseño final de la BD, se procede con la generación de los archivos DDL. Para generar los archivos DLL basta con presionar el botón *Generate DLL* y se mostrará la ventana del generador *Generate Database from Server Model*, que cuenta con dos carpetas: *Target* y *Objects*., como se muestra en la Figura 4.11:

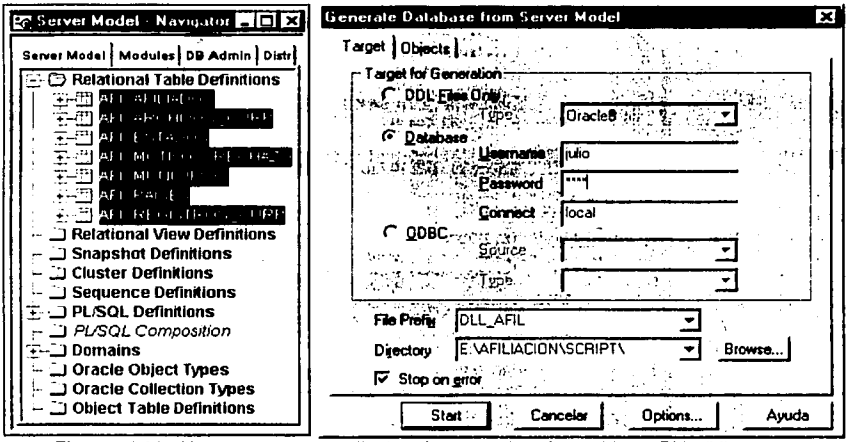

Figura 4.11. Ventana correspondiente al generador de archivos DLL "*Generate Database from Server Model*". En este ejemplo, la cadena de conexión "*local*" apunta a una base de datos "Personal Oracle 8".

En la carpeta de *Target* se debe insertar el nombre que tendrán los archivos, el directorio donde serán creados y seleccionar una de estas tres opciones:

- DDL Files Only. Generar sólo los archivos DLL para ser ejecutados manualmente en la BD seleccionada, como puede ser Oracle, DB2, SQL Server, etc.
- Database. Crear los archivos con opción de ser ejecutados directamente en la BD de Oracle. En esta opción se debe proporcionar el usuario (Username), clave del usuario (Password) y la cadena de conexión (Connect) de la base de datos donde serán creados los objetos. Como se mencionó en el *Capítulo 1*, la cadena de conexión se define utilizando la herramienta SQL*NET.
- ODBC. Generar los archivos utilizando ODBC.

En la carpeta de ***Objects*** se podrán incluir/excluir los objetos que participarán en el proceso o simplemente consultar la selección realizada previamente. Respecto a la función de los botones que se encuentran en la parte inferior de la ventana:

- Start. Ejecutar el proceso.
- Cancel. Cancelar y cerrar la ventana.
- Options... Muestra una lista para seleccionar los tipos de objetos que serán creados.
- Ayuda. Muestra información relacionada al generador.

El generador podrá ser ejecutado tantas veces como se requiera. Cada vez que se ejecute, se mostrará una caja de diálogo con el avance del proceso. Una vez terminado el proceso, el generador creará varios archivos con el mismo nombre pero con diferente extensión, un archivo para cada tipo de objeto y uno más con extensión SQL. Este último será el archivo maestro que contendrá los comandos para ejecutar los otros archivos.

Ejemplo:
Para generar los archivos DLL utilizando el diseño de la BD que se encuentra en el *Server model*, se realizó siguiendo estos pasos:

1. Seleccionar todas las tablas involucradas en el diseño de la BD.
2. Presionar el botón "*Generate DLL*", para visualizar la ventana "*Generate Database from Server Model*".
3. En la carpeta *Target*, seleccionar la opción *Database*. Proporcionar el usuario (Username), clave del usuario (Password) y la cadena de conexión (Connect). En la opción "*File Prefix*" insertar el nombre que tendrán los archivos, por ejemplo, "DLL_AFIL" y en *Directory* insertar la ruta donde serán creados, por ejemplo, "E:\AFILIACION\SCRIPT\", como se muestra en la Figura 4.11.
4. En el botón "*Options...*", verificar que estén marcadas todas las opciones. Debido a que la selección de los objetos se realizó antes de llamar al generador, en la carpeta de "*Objects*" no es necesario hacer cambio alguno.

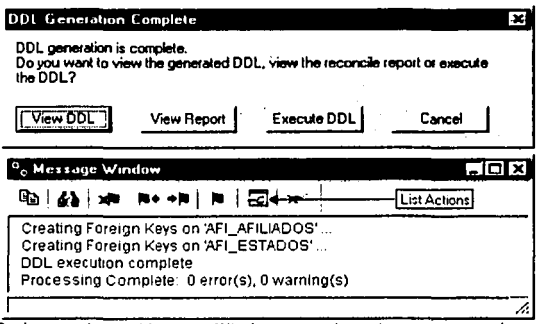

Figura 4.12. La ventana *Message Window* muestra el avance y el resultado del generador de archivos DLL.
La ventana *DDL Generation Complete* se mostrará siempre y cuando no existan errores, indicando que la generación de archivos se ha completado. El botón "*View DLL*" edita los archivos para ver el código generado, el botón "*View Report*" muestra un reporte con el resultado del proceso y con el botón "*Execute DLL*" se ejecutarán los archivos para crear los objetos en la BD.

5. Presionar el botón "*Start*" para iniciar el proceso y se visualizará la ventana "*Message Window*" que mostrará el avance del proceso.
6. Si no se presenta ningún error, se mostrará una segunda ventana (*DLL Generation Complete*) indicando que el proceso ha terminado, como se muestra en la Figura 4.12. Por ultimo, presionar el botón *Execute DLL* para ejecutar los archivos y crear los objetos en la BD.

Para ejecutar los archivos DLL manualmente, hay que tomar en cuenta el siguiente orden de ejecución:

- Primero, ejecutar el archivo con extensión "*.tab*" para crear las tablas en la BD.
- Después, ejecutar los archivos para crear vistas, índices y finalmente los disparadores (triggers), ya que todos los objetos dependen de la existencia de las tablas.

Para ejecutar los archivos DLL directamente desde el prompt de SQL*Plus basta con ejecutar el archivo maestro con extensión "*.sql*", que contiene el código para ejecutar los otros archivos. Después de ejecutar los archivos DLL, la base de datos puede contener: ligas (Database links), secuencias (sequences), constraints, sinónimos (synonyms), tablas (tables), disparadores (triggers), índices (indexes), etc.

El ejemplo anterior corresponde a la primera parte de la etapa de construcción: crear la base de datos. Además de ser la BD que utilizará la aplicación, Designer/2000 utilizará la BD durante la generación de los programas frontales de la aplicación.

El código de los archivos DLL, generados por Designer/2000 en el ejemplo anterior, se podrá consultar en el **Anexo C**.

CAPITULO 5.
DISEÑO DE FORMAS

Además del diseño y construcción de la base de datos se encuentra el diseño y construcción de los programas frontales de la aplicación, como parte del desarrollo de sistemas. Por último, en este capítulo se muestran las herramientas de Designer/2000 que soportan el diseño y construcción de los programas frontales: Transformador para el diseño de aplicaciones (*Application Design Transformer*), Diseñador de módulos (*Design Editor - Modules*) y el Generador de aplicaciones (*Generate*).

5.1. TRANSFORMADOR PARA EL DISEÑO DE APLICACIONES

El *Transformador para el diseño de aplicaciones* (*Application Design Transformer*) podrá ser utilizado durante la etapa de diseño. Esta herramienta toma la información del diagrama jerárquico funcional para crear el diseño preliminar de los programas frontales. Transforma una función, elemental o atómica, en un módulo. Las entidades asociadas a la función serán representadas en el módulo como bloques, los atributos como columnas (items) y las relaciones entre las entidades serán representadas como validaciones (constraints) de llaves primarias. La información de los módulos generados por el *Transformador para el diseño de aplicaciones* es almacenada en el repositorio. Esta herramienta permite:

- Elegir de entre las funciones cuál será transformada en un módulo candidato.
- Mantener los nombres cortos y largos de los módulos como únicos en todo el sistema, tanto para los módulos candidatos y los no candidatos. El transformador no sobrescribe un módulo nuevo sobre uno ya existente y envía información de alerta acerca de los módulos generados.
- Generar aplicaciones a partir de los módulos obtenidos por el transformador.

Un *módulo* es un elemento del repositorio que representa básicamente un programa o parte de un programa de la aplicación final. Puede ser un menú, pantalla, informe, utilidad o programa SQL que realice algunas tareas o proporcione una interfaz de usuario con la base de datos. Hay diferentes lenguajes para los módulos, dependiendo de la herramienta que se utilice para el código final [5.1].

Para generar los módulos candidatos a partir del *Diagrama Jerárquico de Funciones* (DJF), se podrá realizar de dos formas:

1. Desde el *Diagramador Jerárquico Funcional* (*Function hierarchy Diagrammer*). Primero seleccionando la función a transformar y con la opción del menú "*Utilities\ Application Design Transformer*". Se mostrará la ventana del transformador lista para generar el módulo con la información de la función seleccionada.
2. Presionar el botón "*Application Design Transformer*" que se encuentra en el menú principal de Designer/2000. Se mostrará la ventana del transformador para seleccionar la función a transformar.

[5.1] Dorsey, Paul y Peter Koletzke. *Manual de Oracle Designer/2000*, pp. 296.

La ventana del *Transformador para el diseño de aplicaciones* (*Application Design Transformer*) tiene los siguientes componentes, como se muestra en la Figura 5.1:

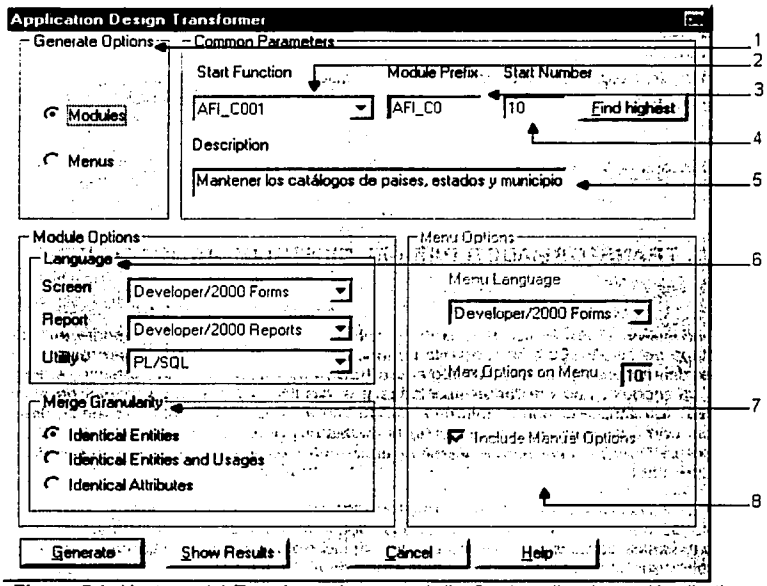

Figura 5.1. Ventana del Transformador para el diseño de aplicaciones (Application Design Transformer).

1. Bloque "Generate Options". Para especificar el tipo de objeto a generar: módulo o menú.
2. Start Function. Muestra una lista para seleccionar la *función* que será utilizada para generar el módulo candidato.
3. Module Prefix. Corresponde al prefijo que será usado para formar el nombre corto del módulo. Inicialmente se mostrarán los primeros seis caracteres del nombre de la aplicación (*application system*), pero podrá ser modificado para insertar otro prefijo.
4. Start Number. Número obligatorio entre 10 y 9999 que formará parte del nombre corto del módulo. Es útil cuando se generan varios módulos con el mismo prefijo (*Module Prefix*) y sólo se agrega éste número para diferenciarlos.
5. Description. Este campo es sólo de consulta y muestra la descripción de la función seleccionada.
6. Language. En este bloque podrá seleccionar el lenguaje de programación que será utilizado para generar los módulos.
7. Merge Granularity. Especificar la opción apropiada que determine el nivel de los datos a comparar. Esto con la finalidad de elegir cuando unir módulos y cuando separarlos.

Puede elegirse una de estas tres opciones para *Merge Granularity* [5.2]:

- *Identical Entities* combina funciones dentro de un módulo si tienen las mismas entidades referenciándolas.
- *Identical Entities and Usages* agrupa funciones en un módulo si las funciones tienen las mismas entidades y el mismo CRUD en esas entidades
- *Identical Attributes* enlaza funciones en un módulo si esas funciones tienen los mismos atributos referenciándolas.

8. Menú Options. Este bloque se habilitará en caso de haber marcado la opción "*Menus*", en el bloque "*Generate Options*", para seleccionar el lenguaje de programación apropiado para generar el módulo de un menú.

Respecto a los botones que se encuentran en la parte inferior del transformador:

- Generate. Crear el módulo candidato.
- Show Results. Muestra el resultado del proceso de generación.
- Cancel. Cancelar y cerrar la ventana.
- Help. Mostrar la ayuda.

Ejemplo:
Como ejemplo, se utilizó la función *AFI_C001* (*Mantener los catálogos de países, estados y municipios*) para generar el módulo candidato correspondiente. Los pasos para generar el módulo candidato son los siguientes (ver Figura 5.1):

1. Presionar el botón *Application Design Transformer*, desde el menú principal de Designer/2000.
2. En la ventana del *Transformador para el diseño de aplicaciones*, marcar la opción "*Modules*" del bloque "*Generate Options*".
3. De la lista "*Start Function*", seleccionar la función *AFI_C001*.
4. En "*Module Prefix*", insertar los primeros 6 caracteres del nombre de la función: *AFI_C0*.
5. En "*Start Number*", insertar el número: 10. En este ejemplo, el nombre del módulo será: *AFI_C00010*. El cual podrá ser modificado con la herramienta *Módulos* del *Editor de diseño*, para dejar el mismo nombre que tiene la función (*AFI_C001*).
6. En "*Language*", seleccionar las herramientas de Oracle (Developer/2000 y PL/SQL).
7. Respecto a "*Merge granularity*", marcar la opción "*Identical Entities*".
8. Presionar el botón "*Generate*" y se visualizará una ventana que mostrará el avance del proceso. Al terminar, se podrá utilizar la herramienta *Módulos* del *Editor de diseño* para ver el módulo candidato.

De la misma forma se realizó la transformación de las funciones involucradas en el DJF del *Anexo B* (*Figura B.1*) para obtener los módulos candidatos de: AFI_C002, AFI_F001, AFI_F002, AFI_R001 y AFI_R002.

[5.2] Dorsey, Paul y Peter Koletzke. *Manual de Oracle Designer/2000*, pp. 298.

5.2. DISEÑO DE MODULOS

Durante la etapa de diseño se podrá utilizar la herramienta *Módulos* (Modules) del *Editor de diseño* para realizar el diseño de los programas frontales, utilizando los módulos candidatos generados con el *Transformador para el diseño de aplicaciones* (*Application Design Transformer*) o creando nuevos módulos directamente desde la herramienta *Módulos*.
En el navegador del *Editor de diseño* se encuentra la carpeta correspondiente a la herramienta *Módulos* (Modules). Con ésta herramienta podrá:

- Crear y mantener el diseño de los módulos.
- Crear y mantener los detalles de cómo se usarán las tablas en los módulos.
- En cada módulo, crear y mantener las validaciones correspondientes a las ligas (relaciones) entre las tablas utilizadas.
- Diseñar la estructura de los programas frontales, que pueden ser: formas (pantallas), reportes o gráficas.
- Crear y mantener la información de campos calculados.
- Depurar el diseño de cada módulo hasta que esté listo para crear el programa frontal desde el *Editor de diseño*.
- Visualizar en forma de diagrama el diseño del módulo. Para ver el diagrama, sólo basta con seleccionar el módulo, arrastrar el puntero al área de trabajo y soltar la selección.

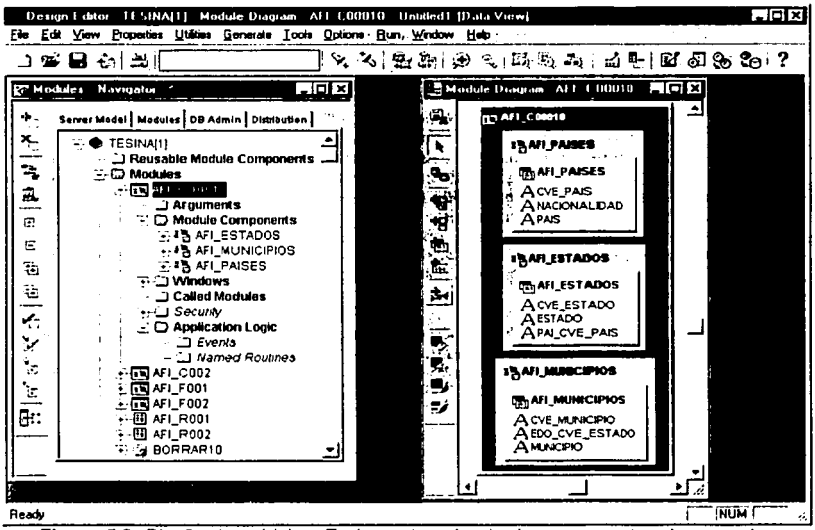

Figura 5.2. Diseño de módulos. En la ventana izquierda se encuentra el navegador de módulos (*Modules - Navigator*). Y del lado derecho, se encuentra el diagrama del módulo *AFI_C00010*.

El diseño de los módulos se podrá visualizar de dos formas, como se muestra en la Figura 5.2:

- Con el navegador (*Modules - Navigator*), que tiene una estructura en forma de árbol.
- Con el diagramador (*Module Diagram*), un diagrama por cada módulo.

Ya sea con el navegador o directamente en el diagrama podrá modificar, crear o borrar objetos para depurar el diseño de los módulos.

Barra de Herramientas (Toolbar).
La barra de herramientas disponible para el diseño de módulos, tiene algunos botones adicionales a los mencionados en el *Capitulo 1*, los cuales a continuación se mencionan:


Botón para visualizar la ventana de preferencias.

Botón "Generate", para visualizar la ventana del generador de aplicaciones.

Cambiar el formato de la ventana de propiedades:

1. Ventana "Property Palettes", para editar todas las propiedades del objeto.
2. Ventana "Property Dialogs " , para editar las propiedades más comunes del objeto.

5.2.1. NAVEGADOR DE MODULOS

El *Navegador de Módulos* (*Modules - Navigator*) muestra los objetos disponibles para el diseño de aplicaciones y tiene una estructura en forma de árbol. En el primer nivel del árbol se encuentra el nombre de la aplicación (*application system*) y en el segundo nivel del árbol se encuentra, entre otros, la definición de los módulos (*Modules*).

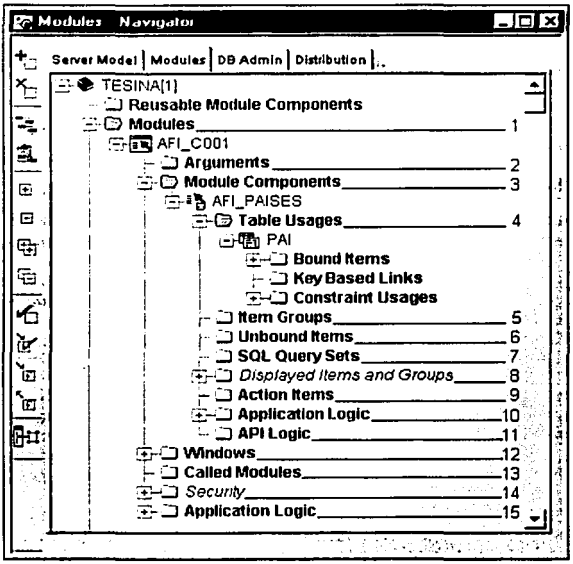

Figura 5.3. Estructura del folder "Modules".

Estructura del folder "*Modules*".
Bajo "*Modules*" se encuentra la siguiente estructura, como se muestra en la Figura 5.3:

1. Modules. Contiene la definición de todos los módulos, ya sean candidatos o no candidatos.
2. Arguments. Contiene los parámetros que utilizará el programa en tiempo de ejecución.
3. Module Components. Contiene la definición de los bloques del módulo. Cada bloque puede estar basado en una tabla o simplemente contener elementos sin basarse en una tabla. Por ejemplo, el bloque "AFI_PAISES" está basado en la tabla del mismo nombre.
4. Table Usages. Muestra la tabla utilizada por el bloque. Así como, los campos (*Bound Items*) de la tabla, la relación (*Key Based Links*) con otros bloques y las validaciones (*Constraint Usages*) de llaves primarias y llaves foráneas.

5. Item Groups. Para definir grupos de elementos (*items*). Por ejemplo, para los datos de una persona se podrían hacer dos grupos: uno con los *items* relacionados con el nombre; y otro, con los *items* relacionados con la dirección.
6. Unbound items. Para definir los campos calculados. Un campo calculado, es un elemento que no está basado en una tabla.
7. SQL Query Sets. Para definir consultas adicionales y ser combinadas con las consultas automáticas de un bloque (*module components*) que esté basado en una tabla.
8. Displayed Items and Groups. Contiene los elementos (*items*) del bloque que serán mostrados al ejecutar el programa. Un *item* puede ser un campo basado en una tabla, un campo calculado o un botón que contenga código para realizar alguna acción.
9. Action Items. Para definir los botones que tendrá el módulo. A cada botón es necesario agregarle las instrucciones, en el lenguaje correspondiente, de la acción a realizar cada vez que sea presionado.
10. Application logic. Agregar las instrucciones relacionadas al bloque, en el lenguaje de programación correspondiente.
11. API logic. Para hacer validaciones adicionales.
12. Windows. Definir las ventanas que tendrá el módulo.
13. Called Modules. Definir el llamado de otros módulos.
14. Security. Para definir la seguridad del módulo, relacionada a los usuarios.
15. Application logic. Agregar las instrucciones relacionadas al módulo, en el lenguaje de programación correspondiente.

En el navegador se muestra un icono para identificar los módulos candidatos o el lenguaje de programación definido para el módulo. Por ejemplo, cuando se utiliza como lenguaje de programación *Oracle Developer/2000*, estos son los posibles iconos:


Para indicar que se trata de un módulo candidato. Un módulo candidato es aquel que ha sido creado desde el *Transformador para el diseño de aplicaciones* (*Application Design Transformer*) y su propiedad "*Candidate ?*" tiene como valor "*Yes*". Una vez seleccionado el lenguaje que será utilizado y al cambiar el valor de la propiedad "*Candidate ? = No*", se mostrará el icono correspondiente al lenguaje.
Cuando un módulo es creado desde el *Editor de diseño*, el valor de la propiedad "*Candidate ?*" será "No". El valor por defecto es "No".


Este icono indica que se utilizará como lenguaje de programación "*Oracle Developer Forms*" para crear un programa frontal (pantalla).


Indica que se utilizará como lenguaje de programación "*Oracle Developer Reports*" para crear el programa de un reporte.


Este icono indica que se realizará el diseño de un menú.


Para indicar que será una librería.

Ejemplo:
Para el módulo candidato "AFI_C00010" (Mantener los catálogos de países, estados y municipios), se modificaron las siguientes propiedades (ver la Figura 5.4):

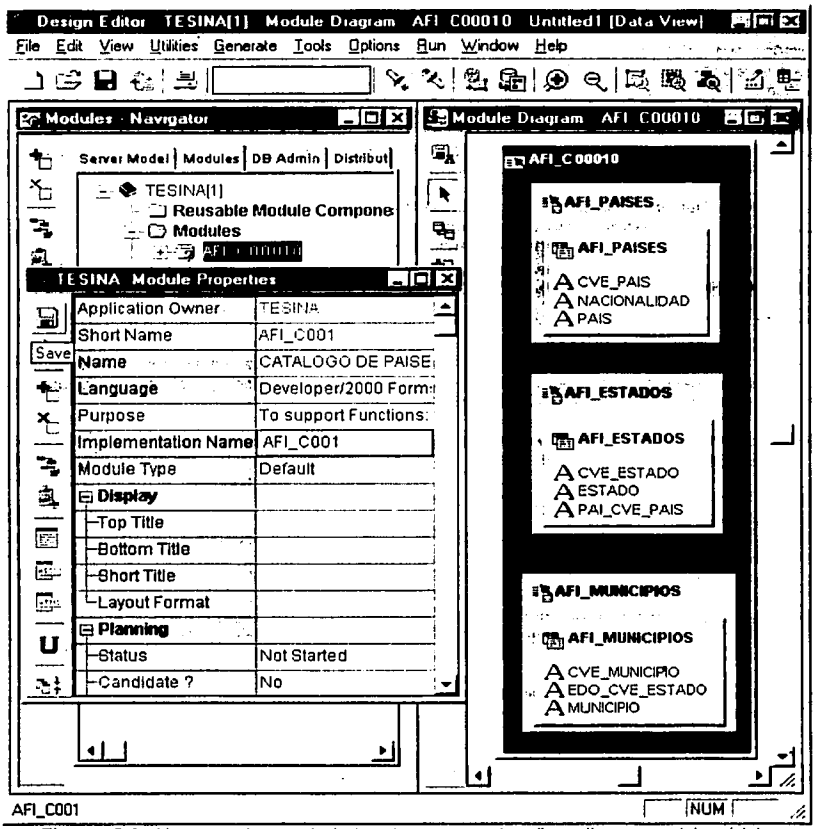

Figura 5.4. Ventana de propiedades "*Property Palettes*" y diagrama del módulo candidato *AFI_C00010*.

- Short Name = AFI_C001. Para que el nombre corto sea igual a la etiqueta de la función utilizada para generar este módulo candidato.

- Name = CATALOGO DE PAISES
- Language = Developer/2000 Forms. Para generar una forma (pantalla). Un programa creado con el lenguaje de "*Developer/2000 Forms*" es conocido como "*forma*".
- Implementation Name = AFI_C001. Nombre que será utilizado para crear el archivo correspondiente al programa. Se recomienda poner el nombre corto (Short Name) del módulo.
- Module Type = Default. Tiene las opciones de: Menu, Library y Default. Esta última es para crear una "*forma*".
- Candidate ? = No.
  Después de modificar las propiedades anteriores, presionar el botón "*SAVE*" para guardar los cambios.
- Además se modificó la secuencia de navegación de los "*module components*", es decir, el orden en que se deberá pasar de un bloque a otro al utilizar la tecla del tabulador. Para esto, se editó la ventana de propiedades de cada "*module components*" y en "*Usage Sequence*" se insertó el número de secuencia "10" para "AFI_PAISES", el número "20" para "AFI_ESTADOS" y el número de secuencia "30" para "AFI_MUNICIPIOS".
- Por último, se modificó el nombre de la ventana (*Windows*), propiedad "*Name=W_CATALOGO*".

5.2.2. DIAGRAMA DEL MODULO

Cada diagrama está basado en la definición de un módulo, los cambios que se realicen en el diagrama serán almacenados en el repositorio afectando la información del módulo, la cual será usada por el generador de aplicaciones. El diagramador cuenta con dos vistas para mostrar el diseño de los módulos: *Data view* y *Display view*. Para cambiar de una vista a otra, basta con presionar el botón que se encuentra en la barra de herramientas del diagrama y que corresponde a los siguientes iconos:


Este icono se muestra cuando se está en *Data view* y al presionarlo se mostrará el diagrama en *Display view*.

Este icono se muestra cuando se está en *Display view* y al presionarlo se mostrará el diagrama en *Data view*.

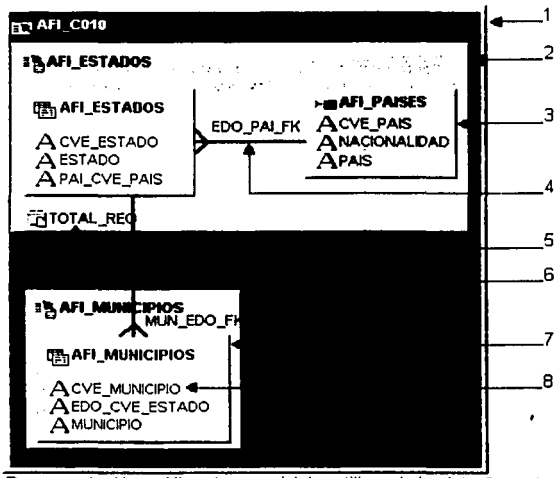

Figura 5.5. Representación gráfica de un módulo, utilizando la vista *Data view*.

Vista *Data view*.
Corresponde a la vista por defecto, la cual muestra el diseño y el nombre de los elementos que serán incluidos por el generador en la aplicación del cliente. En la Figura 5.5 se muestra un diagrama típico con sus componentes:

1. Module. Cada diagrama se basa en la definición de un módulo, mostrando los bloques y las ligas entre las tablas seleccionadas.

2. Module components. Representa un bloque y puede existir uno o más "*Module components*" en un diagrama, unidos por una "liga".
3. Lookup. Representa el diseño de una Lista de Valores (LOV) que sirve para proporcionar información relacionada a una tabla y se utiliza únicamente para consultar información de algún "catálogo". Por ejemplo, en una pantalla para capturar los datos de una persona, una LOV podría ser el catálogo de las "Delegaciones" o "Municipios" y estar relacionado al campo para capturar los datos de "Dirección".
   Para construir un *lookup*, primero seleccionar el componente "*Table Usages*" que será *lookup*, sin soltar se lleva al área del componente "*Table Usages*" que contendrá la LOV y finalmente soltar la selección
4. Liga de una lookup. Es una liga que une a un componente "*Table usages*" con un *lookup*. Representa la relación que existe entre las dos tablas involucradas.
5. Unbound items. Campos que no están basados en alguna tabla.
6. Key based links. Es una liga que representa la relación que existe entre dos tablas, es decir, para crear una relación "*maestro - detalle*" entre dos bloques. Para dibujar una liga, en caso de existir la relación entre las tablas involucradas, basta con presionar el siguiente botón, que se encuentra en la barra de herramientas del diagrama, vista "*Data view*":

   después pulsar con el ratón sobre el área "*Table Usages*" del primer componente y finalmente pulsar con el ratón sobre el área "*Table Usages*" del segundo componente de la relación.
7. Table usages. Representa la tabla usada por el bloque.
8. Bound items. Representan los campos de la tabla y son utilizados para consultar, borrar, insertar o modificar los registros de la tabla.

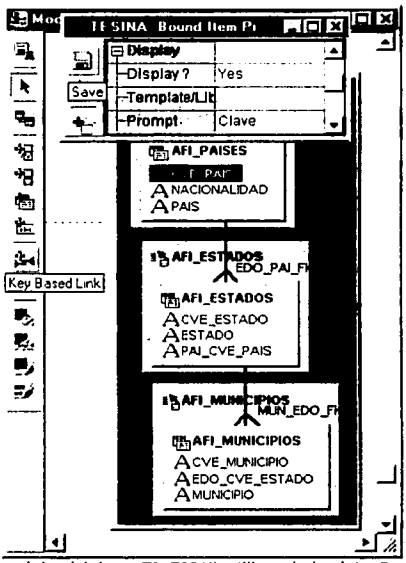

Figura 5.6. Diagrama del módulo "AFI_C001", utilizando la vista *Data view*.

Ejemplo:
Para el módulo "AFI_C001", se utiliza el diagrama "*Data view*" para modificar las siguientes propiedades, como se muestra en la Figura 5.6:

1. Bloque "AFI_PAISES". Para el *item* "CVE_PAIS", modificar la propiedad: Prompt = Clave.
   El "Prompt" corresponde a la etiqueta que se mostrará junto al *item* "CVE_PAIS" al momento de ejecutar el programa.
2. Bloque "AFI_ESTADOS". Para el *item* "CVE_ESTADO", modificar la propiedad: Prompt=Clave.
   Y para el *item* "PAI_CVE_PAIS", que es la FK, modificar la propiedad: Display? = No.
3. Bloque "AFI_MUNICIPIOS". Para el *item* "CVE_MUNICIPIO", modificar la propiedad: Prompt=Clave.
   Para "EDO_CVE_ESTADO", que es la clave foránea, modificar la propiedad: Display? = No.
4. Por último, dibujar las ligas entre los bloques.

Vista *Display view*.
Sólo muestra el diseño de los elementos que serán visualizados en la aplicación del cliente al momento de ser ejecutada. Tomado el diagrama del ejemplo anterior (Figura 5.6), pero en la vista *Display view*, el diagrama mostrará la etiqueta (Prompt) correspondiente en lugar del nombre de cada item, como se muestra en la Figura 5.7. Los componentes del diagrama "*Display view*" son:

1. Module. Representación gráfica del módulo.
2. Window. Representa el "marco" del *canvas*. Un módulo puede tener uno o varios componentes *windows* y un componente *window* puede tener uno o varios *canvas*.
3. Canvas. Es el área donde serán mostrados los elementos, como pueden ser los campos, etiquetas, imágenes, etc.
4. Module Component. Representa un bloque.
5. Bound item. Representan los campos de la tabla.

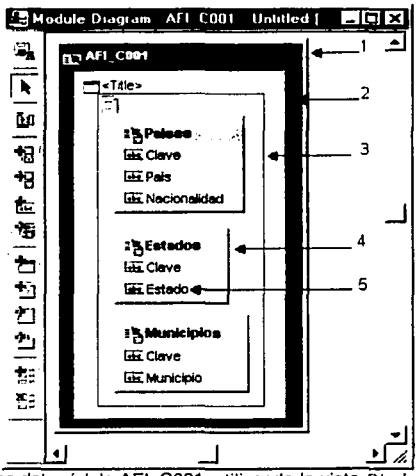

Figura 5.7. Diagrama del módulo AFI_C001, utilizando la vista *Display view*.

Propiedades Generales.
Siguiendo con el ejemplo del módulo "AFI_C001", es necesario definir algunas propiedades generales antes de realizar la generación del programa correspondiente. Las cuales, para este ejemplo, se realizan en la venta de propiedades "*Property Dialogs*" (ver Figura 5.8) y a continuación se mencionan:

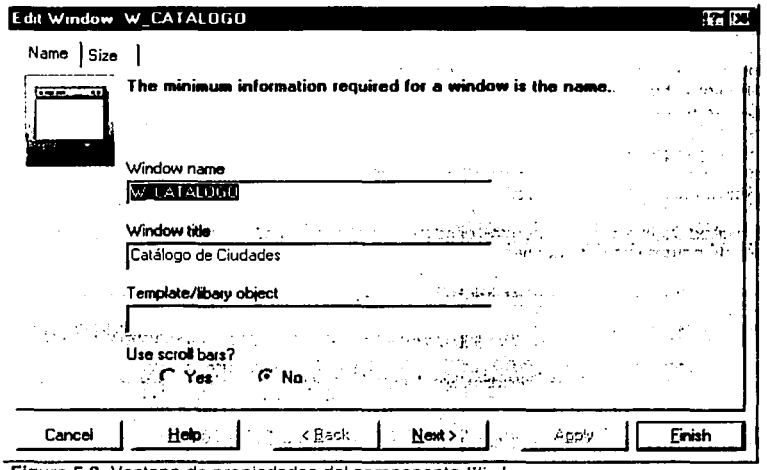

Figura 5.8. Ventana de propiedades del componente *Window*.

- Para el componente *Window* "*W_CATALOGO*":
  En la pestaña *Name*, agregar el titulo de la ventana: *Window title= Catálogo de Ciudades.*

- Para los "*Module Components*" de: AFI_PAISES, AFI_ESTADOS y AFI_MUNICIPIOS.

  1. En la pestaña *Display*, indicar el número de registros que serán mostrados. Con la opción "*Number of rows displayed - rows=10*", es decir, crear el bloque con la posibilidad de ver diez registros al mismo tiempo
  2. Por último, en la pestaña *Placement*, indicar el tipo de canvas a usar con la opción "*Placement of the module component=New Tab canvas page*". Con esta opción se tendrá una pantalla con tres carpetas: una para "Países", otra para "Estados" y una más para "Municipios".
  3. Presionar el botón *Finish* para aceptar los cambios y cerrar la ventana.

5.3. GENERADOR DE MODULOS

Respecto a la etapa de construcción, Designer/2000 cuenta con la herramienta *Generador de módulos* para crear los programas frontales. Después de realizar el diseño de cada módulo de la aplicación es recomendable que se realice un "*checklist*", es decir, asegurarse de que todo esté correcto. Se debe tomar en cuenta que si una parte de la definición del módulo está incompleta, la generación puede realizarse, más no así el producto y/o su funcionalidad. Por ejemplo, se podría realizar el siguiente "*checklist*":

- Por lo menos tener un bloque con una columna,
- el uso de tablas *lookup* si se requiere,
- ligas entre las tablas,
- detalles para las tablas,
- despliegue de las columnas,
- revisar y adecuar las preferencias para los estándares del proyecto,
- la existencia de las tablas físicas en la BD.

De no realizar lo anterior, se podrían presentar problemas de construcción para generar el archivo fuente o de compilación para generar el archivo ejecutable.

La ventana para generar los módulos físicos, utilizando como lenguaje de programación *Oracle Developer/2000*, corresponde a la Figura 5.9 y tiene las siguientes opciones:

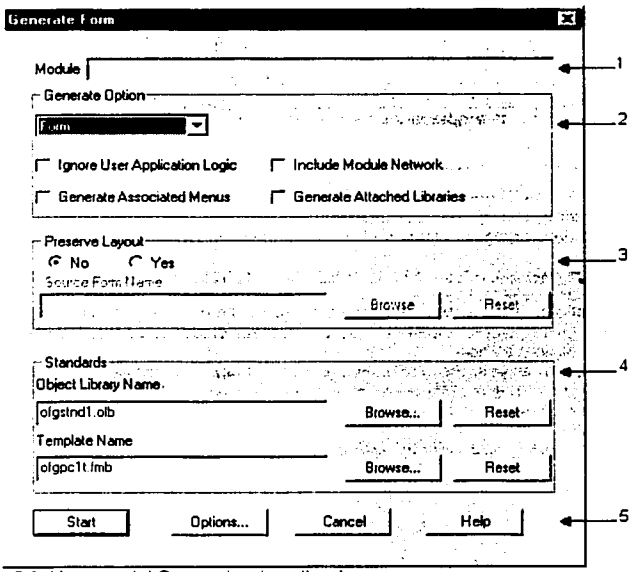

Figura 5.9. Ventana del Generador de aplicaciones.

1. Es sólo de consulta y muestra el nombre del módulo seleccionado.
2. Para indicar el tipo de módulo que será creado, por ejemplo, si será una forma (pantalla) o un reporte.
3. Debido a que la generación de un módulo físico se podrá realizar tantas veces como sea necesario, en ésta opción podrá indicar si desea conservar la apariencia del programa generado anteriormente.
4. En este bloque se muestra el nombre de una librería, con extensión "*.olb*" para *Developer/2000*, que será incluida en el archivo a generar. Además, muestra el archivo que servirá como modelo (template) y del cual se tomaran las características generales para generar el nuevo archivo. Estos archivos podrán ser cambiados por otros personalizados y son opcionales.
5. Botón "*start*", inicia el proceso para generar el módulo físico. El botón "*Options...*" muestra otra ventana para indicar, entre otras opciones, la ruta en la que serán creados los archivos correspondientes al módulo; basta con hacerlo una vez para que todos los archivos generados se encuentren en la ruta indicada. El botón "*Cancel*" cierra la ventana sin generar el módulo. Y con el botón "*Help*" se muestra la ayuda acerca de ésta ventana.

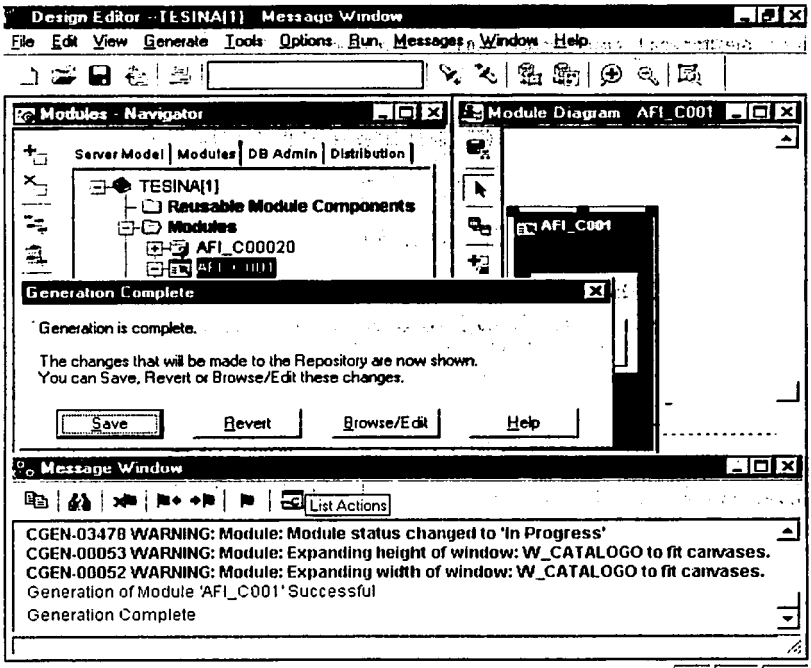

Figura 5.10. Ventana *Message Window* que muestra el resultado del generador de módulos. La ventana *Generation Complete*, indica que el proceso se realizó con éxito.

Ejemplo:
Para generar el módulo físico (archivo fuente y ejecutable) de "AFI_C001":

- Seleccionar el módulo y presionar el botón *Generate* para visualizar la venta "*Generate Form*".
- Presionar el botón "*Options...*". En el campo "*Destination*", de las carpetas "*Form Option, Menu Option*" y "*Compile*", insertar la ruta donde será creado el archivo fuente y el archivo ejecutable. Por ejemplo, "*E:\AFILIACION\FORMAS*". Aceptar los cambios y salir de la ventana.
- Utilizar los archivos que ofrece la herramienta para la librería y el template.
- Presionar el botón *Start* para crear el archivo fuente y el ejecutable. Durante el proceso se mostrará la venta "*Message window*" indicando el avance y al terminar se mostrará el mensaje "*Generation of Module 'AFI_C001' Successful ... Generation Complete*", en caso de no haber errores, como se muestra en la Figura 5.10.
- En la ventana "*Generation complete*", presionar el botón "*Save*" podrá guardar los cambios en el repositorio o de lo contrario presionar el botón "*Revert*".

Con el botón "*List Actions*", de la ventana "*Message window*", se mostrará la ventana de la Figura 5.11. En esta ventana se encuentra en primer lugar la opción para ver el archivo fuente del módulo, por medio de *Developer/2000*. La segunda opción es para ver un archivo extensión "*.LOG*" que contiene el resultado de compilar el archivo fuente para obtener el ejecutable. Y por último, se tiene la opción para ejecutar el nuevo programa.

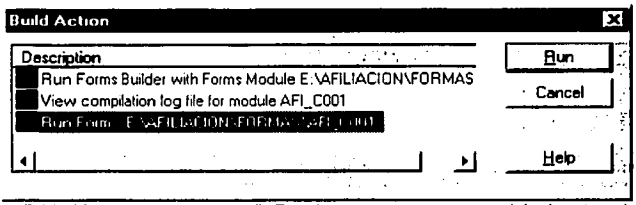

Figura 5.11. Ventana "*Build Action*". Desde esta ventana se podrá ejecutar el nuevo programa para realizar las pruebas correspondientes.

Continuando con el ejemplo del módulo "AFI_C001", seleccionar la opción "*Run Form:...*" y presionar el botón "*Run*" para ver el nuevo programa en ejecución. La Figura 5.12 corresponde al módulo "AFI_C001" en ejecución, el cual tiene los componentes que se mencionaron en el diagrama "*Display view*" de la Figura 5.7:

1. Module.
2. Window.
3. Canvas.
4. Module Component.
5. Bound item.

El resultado del ejemplo anterior son dos archivos: el fuente con extensión "*.FMB*" y el ejecutable con extensión "*.FMX*". Para ver el archivo fuente, es necesario utilizar la herramienta "*Developer/2000 Forms*".

El diseño de módulos para obtener reportes se realiza de manera similar que el diseño de módulos para obtener formas, sólo hay que seleccionar "*Developer/2000 Reports*" en la propiedad "*Language*" del módulo. El archivo fuente de un reporte tiene la extensión "*.RDF* " y los archivos ejecutables de los reportes tienen la extensión "*.REP*". Para ver el archivo fuente de un reporte, es necesario utilizar la herramienta "*Developer/2000 Reports*".
En la ventana del generador de reportes sólo basta con indicar si se utilizará algún *template* y presionar el botón de "*Options...*" para insertar la ruta donde serán creados los archivos (opción "*Directories-Generated Reports*") y para insertar el usuario, la clave de usuario y la cadena de conexión (opción "*Runtime Database User*").

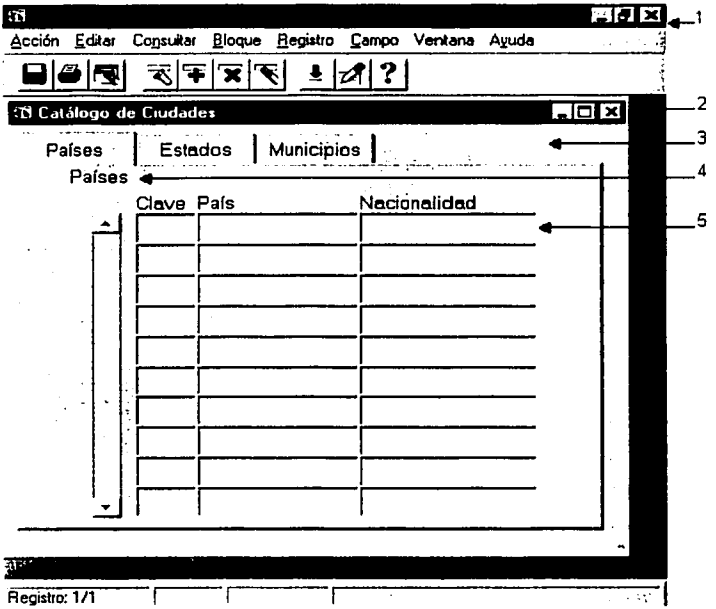

Figura 5.12. Imagen del programa "AFI_C001" en ejecución. Cuenta con 3 carpetas: Países, Estados y Municipios. Para insertar, consultar o modificar la información de las tablas correspondientes.

Las imágenes de los módulos restantes, en ejecución, se muestran en el **Anexo D.** El diseño y la construcción de los módulos se realizó desde Designer/2000. Para construir el menú y la pantalla principal, se utilizó el lenguaje de programación Developer/2000; así como, para afinar algunos detalles de los programas generados desde Designer/2000.

Es fundamental el uso de una metodología en el desarrollo de sistemas, para lograr una real satisfacción de los usuarios y ampliar la vida útil de los sistemas de información. Además, permite contar con productos que serán invaluables para la extensión y mantenimiento de los sistemas informáticos, adaptándolos a la evolución natural de las organizaciones.

Oracle Designer/2000 permite soportar todo el ciclo de vida de los sistemas tradicionales, siendo de gran utilidad para profesionales en el área de Ingeniería de Software. Esta herramienta cuenta con diagramadores que son de gran utilidad para almacenar la información del modelado de sistemas en el repositorio. Además cuenta con generadores, herramientas que facilitan la construcción de sistemas cubriendo gran parte de las etapas de desarrollo; generan aproximadamente un 60% del código y sólo es necesario agregar un 40% de código para completar la aplicación. El grupo de generadores se puede clasificar en dos tipos:

- El generador de código para el servidor. Es una herramienta para crear los programas SQL que se usarán para crear los objetos de la base de datos.
- Los generadores de programas frontales. Son herramientas para generar productos de Developer/2000, Visual Basic, Oracle WebServer y C++, entre otros.
  Los generadores para Developer/2000 (Forms, Reports y Graphics) se diferencian de los otros generadores frontales en que no requieren un compilador externo y de la capacidad de producir programas frontales completamente terminados y ejecutables.
  Los otros generadores requieren una compilación del producto resultante en la herramienta de programación correspondiente al lenguaje. Para crear la aplicación final, es necesario efectuar algo de trabajo fuera de Designer/2000, es decir, es necesario realizar la compilación del código generado por Designer/2000 con el resto del código escrito con cualquier otro lenguaje de programación. La ventaja es que puede confiarse en Designer/2000 para crear el código específico de los datos y concentrarse en el código del resto de la aplicación.

Debido a lo anterior, en el presente trabajo se utilizaron los generadores de Developer/2000 para crear los programas frontales. Además, para mejorar la apariencia y funcionalidad de los módulos construidos con Designer/2000 fue más fácil y rápido utilizando la herramienta de desarrollo Developer/2000.

En los proyectos que involucran herramientas "CASE", es necesario tener en cuenta los siguientes factores:

- Contar con una metodología.
- Contar con una herramienta CASE adecuada para soportar la metodología que se utilizará, que cubra las expectativas y necesidades de los analistas, desarrolladores y de los usuarios.
- La divulgación de una cultura organizacional e informática con el fin de facilitar la asimilación y dominio de este tipo de tecnologías.
- Finalmente, aún cuando la herramienta CASE permite generar aplicaciones, es necesario contar con una herramienta de desarrollo para complementar y depurar las aplicaciones generadas con la herramienta CASE.

Barker, Richard,
CASE*METHOD, Entity Relationship Modelling
Addison-Wesley, impreso en Inglaterra, 1989.

Besser, Brian y Otros,
Oracle7 para desarrolladores, Guía del participante,
Develop Applications with the Procedural Option
Producto de Oracle Corporation, impreso en U.S.A., 1992.

Cabrera, Gregorio y Guillermo Montoya,
Análisis y diseño detallado de aplicaciones informáticas de gestión
McGall-Hill, impreso en España, 1999.

Dorsey, Paul y Peter Koletzke,
Manual de Oracle Designer/2000
Osborne/McGraw-Hill, impreso en España, 1997.

Loney, Kevin,
Oracle manual del administrador
McGraw-Hill, impreso en México, 1995.

Lounsberry, Joni y Otros,
Designer/2000 R2: Forms Design and Generation
Producto de Oracle Corporation, 1998.

Senn, James,
Análisis y diseño de sistemas de información
McGraw-Hill, impreso en México, 1992.

Yourdon, Edward,
Análisis Estructurado Moderno
Prentice-Hall Hispanoamericana, impreso en México, 1993.

ANEXO A
MODELO ENTIDAD RELACION

Para ejemplificar el presente trabajo, se muestra el prototipo de una aplicación que tienen por objetivo llevar el control de las solicitudes de nuevos afiliados, es decir, posibles clientes en una empresa "Administradora de Fondos para el Retiro (AFORE)"; así como, generar archivos con la información de los afiliados que no cuentan con la Clave Unica de Registro de Población "CURP"; y leer los archivos de respuesta de la solicitud, conocidos como archivos de distribución de CURP.

Para el registro de nuevos afiliados, es necesario capturar los datos generales: folio de solicitud, nombre completo, sexo, edad, dirección, número de seguro social, RFC y/o CURP, lugar de nacimiento, nacionalidad y fecha de nacimiento.
De los nuevos afiliados es necesario identificar aquellos que no cuentan con la CURP para generar un archivo de solicitud, con los datos generales de cada afiliado, y que será enviado a la entidad encargada de asignar la CURP.
Es necesario realizar una copia de los registros incluidos en cada archivo de solicitud para llevar un control y un histórico de los registros que han sido enviados a la entidad encargada de asignar la CURP. Estos registros deberán estar en un estado de pendientes hasta recibir el archivo de repuesta. Los datos del archivo deberán ser almacenados en la base de datos en espera de la respuesta, se deberá registrar: el número de lote; la fecha en que se generó el archivo; el tipo de operación, que inicialmente tendrá la clave de solicitud (71) y posteriormente será actualizada con la clave de respuesta (72); el total de registros incluidos en el archivo.
Los registros incluidos en un archivo de solicitud, serán aquellos que no cuenten con la CURP y que no estén en espera de recibir la CURP.

El archivo de distribución de CURP podrá contener tanto registros aceptados como registros rechazados. Los datos del archivo, y de los registros involucrados, deberán ser actualizados de acuerdo a la respuesta de la entidad encargada de asignar la CURP. Cuando un registro es rechazado, se indicará el motivo por el cual se rechaza. Cuando un registro es aceptado, éste contendrá los siguientes datos: la CURP asignada; una clave indicando la aceptación; el folio y la clave del documento que confirman la asignación de la CURP; la clave de alta proporcionada por la entidad encargada de asignar la CURP.

Formato para el nombre de los archivos.
El nombre de los archivos estará formado por 12 dígitos más la extensión, como se muestra a continuación:

- Las primeras cuatro posiciones para indicar el año, por ejemplo: 2001.
- Las siguientes ocho posiciones corresponden a un número consecutivo, el cual iniciara al cambio de año, por ejemplo: 00000001.
- La extensión del archivo será: "001" para indicar que es un archivo de solicitud; "002" para identificar los archivos de respuesta con la distribución de CURP.

Por ejemplo:

- El nombre del primer archivo, del año 2001, para solicitar la asignación de la CURP sería: 200100000001.001
- Y el nombre del archivo de respuesta con la distribución de CURP sería: 200100000001.002

FORMATO PARA LOS ARCHIVOS DE DISTRIBUCION DE CURP

Encabezado de distribución de CURP.

Id	Nombre del campo: Descripción.	Posición
1	Tipo de registro, 2 posiciones: "01" para indicar que es el encabezado.	001 – 002
2	Numero de lote, 12 posiciones: Corresponde al nombre del archivo.	003 – 014
3	Entidad que solicita, 3 posiciones: Clave de la entidad que solicita el servicio, en este caso siempre será "055".	015 – 017
4	Espacios: 2 posiciones.	018 – 019
5	Tipo de operación, 2 posiciones: "71" para solicitud. "72" para distribución (respuesta de la solicitud).	020 – 021
6	Tipo de servicio, 2 posiciones: "07" indica el servicio de CURP.	022 – 023
7	Clave de la entidad encargada de asignar la CURP, 3 posiciones: Siempre será "001".	024 – 026
8	Espacios: 30 posiciones.	027 – 056
9	Fecha del lote, 8 posiciones: Con formato "YYYYMMDD".	057 – 064
10	Espacios: 336 posiciones.	065 – 400

Detalle de distribución de CURP.

Id	Nombre del campo: Descripción.	Posición
1	Tipo de registro, 2 posiciones: "04" para indicar que es el detalle.	001 – 002
2	Espacios, 4 posiciones: Rellenar con ceros "0".	003 – 006
3	Numero de secuencia, 6 posiciones: Secuencia física del registro.	007 – 012
4	Espacios: 2 posiciones.	013 – 014
5	NSS, 11 posiciones: Número de seguro social.	015 – 025
6	CURP, 18 posiciones: Para los archivos de solicitud insertar blancos.	026 – 043
7	Espacios: 13 posiciones.	044 – 056
8	Apellido paterno: 40 posiciones.	057 – 096
9	Apellido materno: 40 posiciones.	097 – 136
10	Nombres: 40 posiciones.	137 – 176

11	Fecha nacimiento, 8 posiciones: Formato "YYYYMMDD".	177 – 184
12	Espacios: 18 posiciones.	185 – 202
13	Espacios: 9 posiciones.	203 – 211
14	Sexo, de 1 posición: H – Hombre. M – Mujer.	212 – 212
15	Municipio, 3 posiciones: Clave de la Delegación o Municipio correspondiente al lugar de nacimiento.	213 – 215
16	Entidad de nacimiento, 3 posiciones: Conforme al catálogo general "Claves de Entidades Federativas".	216 – 218
17	Nacionalidad, 3 posiciones: Conforme al catálogo de Países.	219 – 221
18	Folio del documento probatorio, 10 posiciones: Para los archivos de solicitud insertar blancos.	222 – 231
19	Número de documento probatorio, 16 posiciones: Para los archivos de solicitud insertar blancos.	232 – 247
20	Resultado de la solicitud, de 2 posiciones: "01" – Aceptado. "02" – Rechazado.	248 – 249
21	Motivo de rechazo, 3 posiciones: Cuando es aceptado (Id 22 = 01) se rellena con ceros y cuando es rechazado (Id 22 = 02) contendrá la clave de rechazo rellenado con ceros a la izquierda.	250 – 252
22	Espacios: 28 posiciones.	253 – 280
23	Folio del afiliado, 12 posiciones: Número de folio asignado al afiliado.	281 – 292
24	Fecha del CURP, 8 posiciones: Formato "YYYYMMDD". Para los archivos de solicitud insertar blancos.	293 – 300
25	Espacios: 42 posiciones.	301 – 342
26	Tipo de operación, 3 posiciones: "007" – Alta.	343 – 345
27	Estatus de salida, 2 posiciones: Clasificado por la entidad encargada de asignar la CURP. Para los archivos de solicitud insertar blancos.	346 - 347
28	Espacios: 5 posiciones.	348 – 352
29	Archivo de inclusión, 6 posiciones: Clave de alta proporcionada por la entidad encargada de asignar la CURP. Para los archivos de solicitud insertar blancos.	353 – 358
30	Archivo de modificación: 6 posiciones, rellenar con ceros.	359 – 364
31	Espacios: 36 posiciones.	365 - 400

Sumario de distribución de CURP.

Id	Nombre del campo: Descripción.	Posición
1	Tipo de registro, 2 posiciones: "09" para indicar que es el sumario.	001 – 002
2	Numero de lote, 12 posiciones: Corresponde al nombre del archivo.	003 – 014
3	Entidad que solicita, 3 posiciones: Clave de la entidad que solicita el servicio, en este caso siempre será "055".	015 – 017
4	Espacios: 2 posiciones.	018 – 019
5	Tipo de operación, 2 posiciones: "71" para solicitud. "72" para distribución (respuesta de la solicitud).	020 – 021
6	Tipo de servicio, 2 posiciones: "07" indica el servicio de CURP.	022 – 023
7	Clave de la entidad encargada de asignar la CURP, 3 posiciones: Siempre será "001".	024 – 026
8	Espacios: 30 posiciones.	027 – 056
9	Fecha del lote, 8 posiciones: Con formato "DDMMYYYY".	057 – 064
10	Total de registros, 9 posiciones: También se cuenta el encabezado y el sumario.	065 – 073
11	Espacios: 327 posiciones.	074 – 400

Motivos de rechazo para los registros procesados (detalle - Id 21).

Clave	Descripción
10	Primer apellido, debe contener valor.
11	Segundo apellido, debe contener valor.
12	Nombre(s), debe contener valor.
13	Sexo, el valor debe ser: H – Hombre; M – Mujer.
14	Fecha de nacimiento, debe ajustarse a los valores de fecha.
15	Entidad federativa de nacimiento, clave incorrecta.
16	Nacionalidad, clave incorrecta.
17	Error de origen, el registro debe tener una longitud de 400 caracteres.
18	El archivo no contiene registros de detalle.

Estatus de salida para los registros procesados (detalle - Id 27).

Clave	Descripción
10	CURP ya existe, asignada correctamente.
11	Registro dado de alta normal, nueva CURP re - calculada.
12	Registro dado de alta normal, nueva CURP asignada correctamente.

DETALLES PARA LA CONSTRUCCION DEL DER

Detalle de las entidades y de las relaciones utilizadas para la construcción del "*Diagrama Entidad Relación*".

ENTIDADES

Entidad: REGISTRO_CURP.
Para almacenar los datos generales de los afiliados que están en espera de asignación de la CRUP, es decir, los registros que son incluidos en cada archivo enviado a la entidad encargada de asignar la CURP. Así como, de los que ya recibieron respuesta, es decir, los registros correspondientes a cada archivo de distribución de CURP.

Name	Seq	Domain	Opt	Format (MaxLen/Dec)	Primary	Comment
ID_REGISTRO	10		*	NUMBER(6)	PK	Numero consecutivo de acuerdo al lote.
COD_RESUL_OPER	20		*	VARCHAR2(2)		Resultado de la operación: '01'=Aceptado; '02'=Rechazado.
TIP_TRANSACCION	30		*	VARCHAR2(3)		Tipo de transacción: '007'=Alta; '008'=Cambio.
STATUS_RENAPO	40		*	NUMBER(2)		Estatus de salida de Renapo.
NOMBRE	50		*	VARCHAR2(60)		Nombre(s).
A_PATERNO	60		*	VARCHAR2(30)		Apellido Paterno.
A_MATERNO	70		*	VARCHAR2(30)		Apellido Materno.
F_NACIMIENTO	80		*	DATE		Fecha de Nacimiento.
SEXO	90		*	VARCHAR2(1)		Seleccione: M(asculino) o F(emenino).
ACTIVA	100		*	VARCHAR2(2)		La CURP esta activa? : S/N. Sólo un 'S' por afiliado.
CURP	110		°	VARCHAR2(18)		Clave única de registro poblacional.
FOLIO_DOCTO	120		°	VARCHAR2(10)		Folio del documento probatorio.
DOCTO_PROBATORIO	130		°	VARCHAR2(16)		Documento probatorio.
F_ALTA_CURP	140		°	DATE		Fecha de alta CURP.
CVE_ALTA_RENAPO	150		°	VARCHAR2(6)		Clave del archivo de alta, asignada por Renapo.
CVE_MODI_RENAPO	160		°	VARCHAR2(6)		Clave del archivo de modificación, asignada por Renapo.
USU_ALTA	170		*	VARCHAR2(30)		Usuario que dio de alta el registro.
F_ALTA	180		*	DATE		Fecha en la que se dio de alta el registro.
USU_MODIF	190		°	VARCHAR2(30)		Usuario que modifico el registro.
F_MODIF	200		°	DATE		Fecha en que se modifico el registro.

Entidad: AFILIADO.
Para almacenar los datos generales de los nuevos afiliados.

Name	Seq	Domain	Opt	Format (MaxLen/Dec)	Primary	Comment
ID_FOLIO	10		*	NUMBER(12)	PK	Numero de Folio que identifica al afiliado.
TIPO_PERSONA	20		*	VARCHAR2(1)		Indicar si es persona F(isica) o M(oral).
NOMBRE	30		*	VARCHAR2(60)		Nombre del afiliado.
A_PATERNO	40		*	VARCHAR2(30)		Apellido Paterno del afiliado.
A_MATERNO	50		*	VARCHAR2(30)		Apellido Materno del afiliado.
F_NACIMIENTO	60		*	DATE		Fecha de nacimiento.
SEXO	70		*	VARCHAR2(1)		M(asculino) o F(emenino).
CALLE	80		*	VARCHAR2(60)		Domicilio: Calle.
COLONIA	90		*	VARCHAR2(60)		Domicilio: Colonia.
CP	100		°	VARCHAR2(6)		Código Postal.
NSS	110		°	NUMBER (11)		Numero de Seguro Social.
RFC	120		°	VARCHAR2(13)		Registro Federal de Contribuyentes.
CURP	130		°	VARCHAR2(18)		Clave única de registro poblacional.
TEL_CASA	140		°	VARCHAR2(20)		Teléfono particular (casa o celular).
TEL_OFICINA	150		°	VARCHAR2(20)		Teléfono oficina.
F_SOLICITUD	160		°	DATE		Fecha de la solicitud.
USU_ALTA	170		*	VARCHAR2(30)		Usuario que dio de alta el registro.
F_ALTA	180		*	DATE		Fecha en que se dio de alta el registro.
USU_MODIF	190		°	VARCHAR2(30)		Usuario que modifico el registro.
F_MODIF	200		°	DATE		Fecha en que se modifico el registro.

Entidad: MUNICIPIO.
Utilizado para el catálogo de "Municipios".

Name	Seq	Domain	Opt	Format (MaxLen/Dec)	Primary	Comment
CVE_MUNICIPIO	10		*	VARCHAR2(6)	PK	Clave del Municipio.
MUNICIPIO	20		*	VARCHAR2(60)		Nombre del Municipio o Delegación.

Entidad: ESTADO.
Utilizado para el catálogo de "Estados".

Name	Seq	Domain	Opt	Format (MaxLen/Dec)	Primary	Comment
CVE_ESTADO	10		*	VARCHAR2(6)	PK	Clave del Estado.
ESTADO	20		*	VARCHAR2(30)		Nombre del Estado.

Entidad: PAIS.
Utilizado para el catálogo de "Países".

Name	Seq	Domain	Opt	Format (MaxLen/Dec)	Primary	Comment
CVE_PAIS	10		*	VARCHAR(6)	PK	Clave del País.
PAIS	20		*	VARCHAR(30)		Nombre del País.
NACIONALIDAD	30		°	VARCHAR(30)		Nacionalidad.

Entidad: ARCHIVO_CURP.
Para almacenar los datos generales de los archivos enviados a la entidad encargada de asignar la CURP. Así como, actualizar los datos correspondientes a cada archivo de repuesta (distribución de CRUP).

Name	Seq	Domain	Opt	Format (MaxLen/Dec)	Primary	Comment
ID_LOTE	10		*	NUMBER(12)	PK	Numero de lote.
F_LOTE	20		*	DATE		Fecha del lote.
TIPO_OPERACION	30		*	NUMBER(2)		Identificador de operación.
TIPO_SERVICIO	40		*	NUMBER(2)		Identificador del servicio.
REG_PROCESADO	50		*	NUMBER(9)		Numero de registros procesados
F_RECEP_LOTE	60		°	DATE		Fecha en que se recibió el lote.
USU_ALTA	70		*	VARCHAR2(30)		Usuario que dio de alta el registro.
F_ALTA	80		*	DATE		Fecha en que se dio de alta el registro.
USU_MODIF	90		°	VARCHAR2(30)		Usuario que modifico el registro.
F_MODIF	100		°	DATE		Fecha en que se modifico el registro.

Entidad: MOTIVO_RECHAZO.
Utilizado para el catálogo "Motivos de rechazo".

Name	Seq	Domain	Opt	Format (MaxLen/Dec)	Primary	Comment
CVE_RECHAZO	10		*	NUMBER(6)	PK	Código del motivo de rechazo.
TIPO	20		*	VARCHAR2(3)	PK	Rechazo para: '007'=Alta; '008'=Cambio.
DESCRIPCION	30		*	VARCHAR2(60)		Descripción del motivo de rechazo.

Nota.
Para la columna *Opt*:

- * = No es opcional, siempre debe tener información.
- ° = Sí es opcional, puede tener información o ser nulo.

RELACIONES.

Entidades	From Name	To Name	Tipo Relación
PAIS ESTADO	de	tener	DE UNO (opcional) A MUCHOS (obligatorio). Pasar la llave primaria de PAIS para formar parte de la llave primaria en ESTADO.
ESTADO MUNICIPIO	de	tener	DE UNO (opcional) A MUCHOS (obligatorio). Pasar la llave primaria de ESTADO para formar parte de la llave primaria en MUNICIPIO.
MUNICIPIO AFILIADO	en	habitado	DE UNO (opcional) A MUCHOS (obligatorio). Pasar sólo como llave foránea.
MUNICIPIO AFILIADO	de origen	lugar de origen	DE UNO (opcional) A MUCHOS (obligatorio). Pasar sólo como llave foránea.
AFILIADO REGISTRO_CURP	asignado	en	DE UNO (opcional) A MUCHOS (obligatorio). Pasar sólo como llave foránea.
MUNICIPIO REGISTRO_CURP	formado por	en	DE UNO (opcional) A MUCHOS (obligatorio). Pasar sólo como llave foránea.
MOTIVO_RECHAZO REGISTRO_CURP	rechazado	en	DE UNO (opcional) A MUCHOS (opcional). Pasar sólo como llave foránea.
ARCHIVO_CURP REGISTRO_CURP	en	formado	DE UNO (opcional) A MUCHOS (obligatorio). Pasar la llave primaria de ARCHIVO_CURP para formar parte de la llave primaria en REGISTRO_CURP.

DIAGRAMA ENTIDAD RELACION

En la Figura A.1 se muestra el DER construido con la definición de entidades y relaciones mencionadas anteriormente.

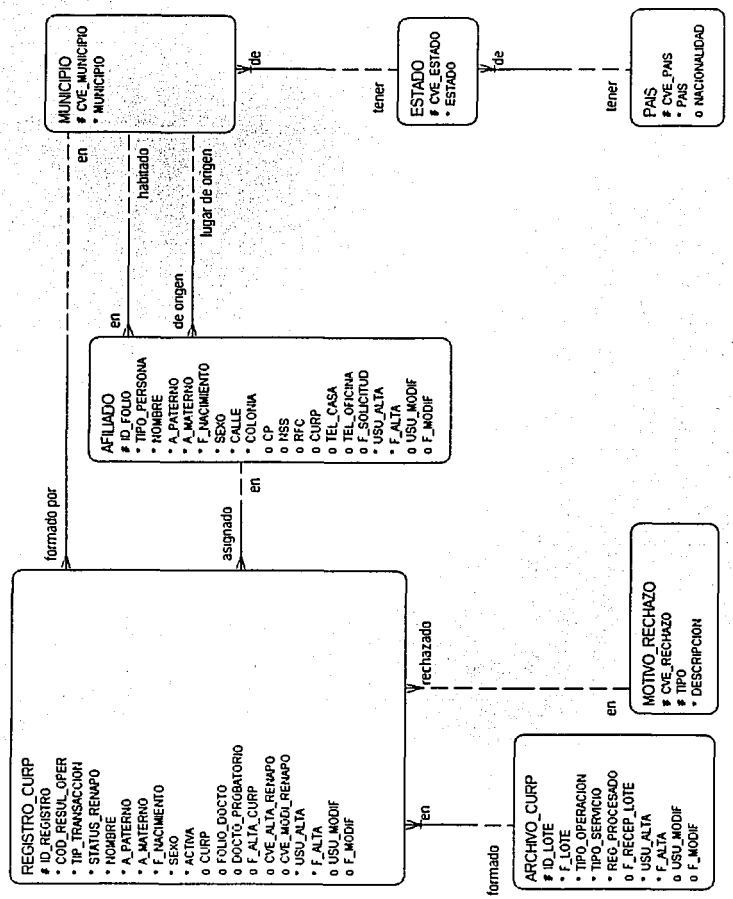

Figura A.1. Diagrama Entidad Relación (DER).

ANEXO B
MODELO JERARQUICO FUNCIONAL

Detalle de las funciones utilizadas para la construcción del "*Diagrama Jerárquico Funcional*".

FUNCION RAIZ/PADRE.

Etiqueta	Definición Corta	Tipo	Padre
AFILIACION	Gestionar las solicitudes de afiliación.	Raíz	- - -
CATALOGOS	Mantener los catálogos.	Padre	AFILIACION
REGISTRO	Soportar el registro de afiliados y la distribución de la CURP.	Padre	AFILIACION
REPORTES	Generar reportes.	Padre	AFILIACION

FUNCION HIJA.

Etiqueta	Definición Corta	Tipo	Entidades	Permiso	Atributos
Función Padre: CATALOGOS					
AFI_C001	Mantener los catálogos de Países, Estados y Municipios	Atómica	PAIS ESTADO MUNICIPIO	CRUD	Todos los atributos. Poner como "no nulos" los atributos que deben tener información de acuerdo al DER.
AFI_C002	Mantener el catálogo de motivos de rechazo.	Atómica	MOTIVO_RECHAZO	CRUD	Todos los atributos. Poner como "no nulos" los atributos que deben tener información de acuerdo al DER.
Función Padre: REGISTRO					
AFI_F001	Registrar las solicitudes de afiliados.	Atómica	AFILIADO PAIS ESTADO MUNICIPIO	CRUD R	Todos los atributos. Poner como "no nulos" los atributos que deben tener información de acuerdo al DER. Todos los atributos.
AFI_F002	Cargar archivos de distribución de la CURP.	Atómica	ARCHIVO_CURP REGISTRO_CURP MOTIVO_RECHAZO	CRUD R	Todos los atributos. Poner como 'no nulos' los atributos que deben tener información de acuerdo al DER. Todos los atributos.
AFI_F003	Crear archivo para solicitar la CURP de nuevos afiliados que no cuentan con la clave.	Atómica	AFILIADO	R	Todos los atributos y sólo de consulta.

Etiqueta	Definición Corta	Tipo	Entidades	Permiso	Atributos
Función Padre: REPORTES					
AFI_R001	Reportar afiliados sin CURP.	Atómica	AFILIADO REGISTRO_CURP	R	Todos los atributos.
AFI_R002	Reportar la distribución de CURP.	Atómica	AFILIADO ARCHIVO_CURP REGISTRO_CURP	R	Todos los atributos.

Donde:

C = Create (crear).
R = Retrive (recuperar).
U = Update (actualizar).
D = Delete (borrar).

DIAGRAMA JERARQUICO FUNCIONAL (DJF)

De las funciones anteriores, se deriva el diagrama de la Figura B.1. La función "AFI_C001" será utilizada para la operación de tres catálogos: Países, Estados y Municipios.
Otra opción sería crear una función para cada catálogo, como se muestra en la Figura B.2, la función "AFI_C001" para Países, "AFI_C002" para Estados y "AFI_C003" para Municipios.
El diagrama utilizado para crear los módulos candidato es el de la Figura B.1.

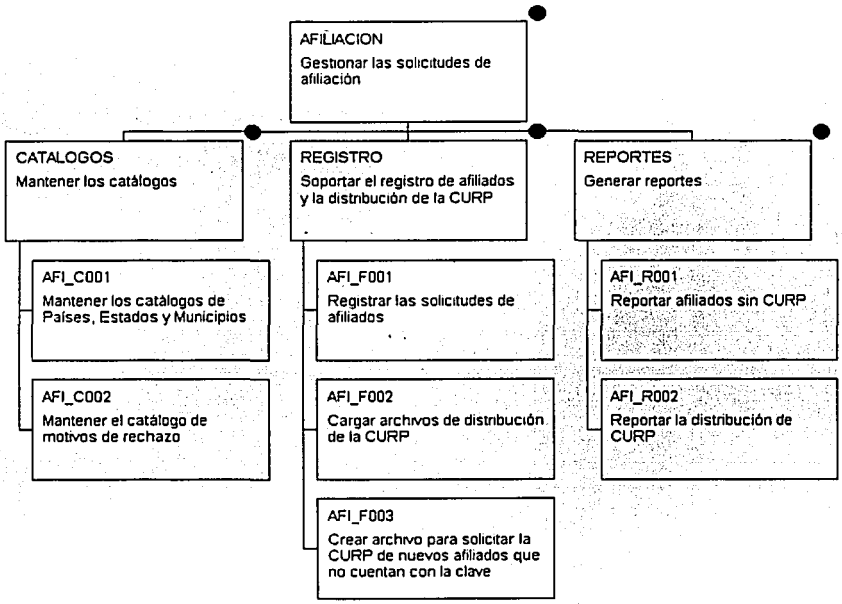

Figura B.1. Diagrama Jerárquico Funcional (DJF). La función "AFI_C001" será utilizada para el mantenimiento de los catálogos de: Países, Estados y Municipios.

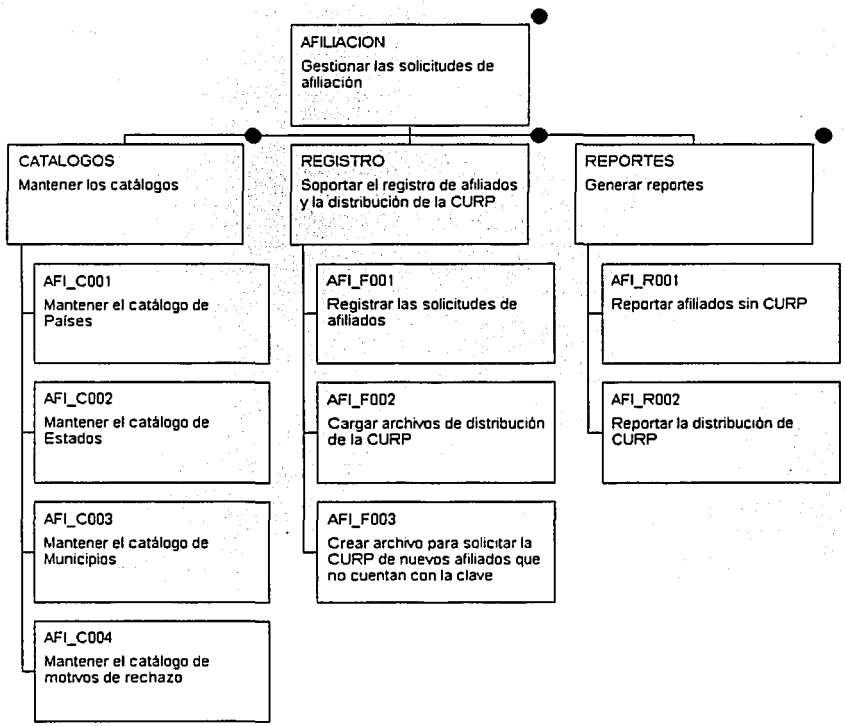

Figura B.2. Diagrama Jerárquico Funcional (DJF). En este diagrama se muestra una función para cada catálogo de: Países, Estados y Municipios.

ANEXO C
ARCHIVOS DLL

La Figura C.1 muestra el diagrama del diseño de la base de datos, que fue generado con la información del "*Diagrama Entidad Relación*".

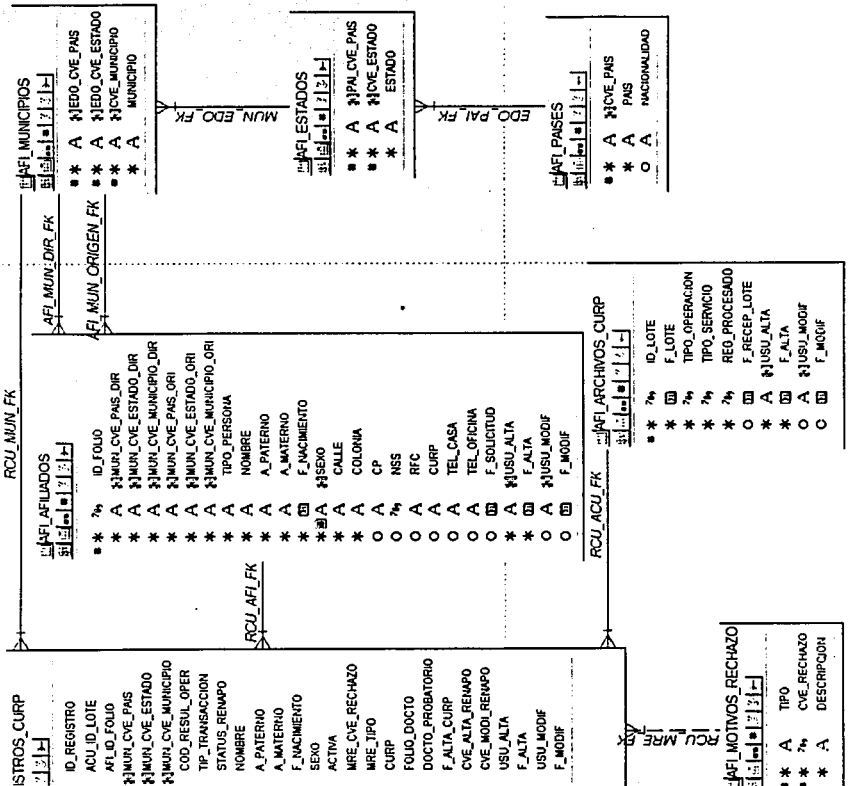

Figura C.1. Diagrama del diseño de la base de datos.

Archivos generados.
Designer/2000 toma la información del diseño de la base de datos, Figura C.1, para generar los archivos DDL:

Nombre.extensión	Contenido
DLL_AFIL.SQL	Es el archivo maestro que hace la llamada de los otros archivos para que sean ejecutados.
DLL_AFIL.TAB	Código para crear las tablas.
DLL_AFIL.IND	Código para crear los índices.
DLL_AFIL.CON	Código para crear los constraint de las Primary Key y Foreign Key.

Código del archivo DLL_AFIL.SQL.

```
-- e:\afiliacion\Script\Dll_Afil.sql
--
-- Generated for Oracle 8 on Sun Jan 14  12:10:35 2001 by Server Generator 6.0.3.3.0

SPOOL Dll_Afil.lst

@@ Dll_Afil.tab
@@ Dll_Afil.ind
@@ Dll_Afil.con

SPOOL OFF
```

Código del archivo DLL_AFIL.TAB.

```
-- e:\afiliacion\Script\Dll_Afil.tab
--
-- Generated for Oracle 8 on Sun Jan 14  12:10:34 2001 by Server Generator 6.0.3.3.0

PROMPT Creating Table 'AFI_REGISTROS_CURP'
CREATE TABLE AFI_REGISTROS_CURP
(ID_REGISTRO NUMBER(6) NOT NULL
,ACU_ID_LOTE NUMBER(12) NOT NULL
,AFI_ID_FOLIO NUMBER(12) NOT NULL
,MUN_CVE_PAIS VARCHAR2(6) NOT NULL
,MUN_CVE_ESTADO VARCHAR2(6) NOT NULL
,MUN_CVE_MUNICIPIO VARCHAR2(6) NOT NULL
,COD_RESUL_OPER VARCHAR2(2) NOT NULL
,TIP_TRANSACCION VARCHAR2(3) NOT NULL
,STATUS_RENAPO NUMBER(2) NOT NULL
,NOMBRE VARCHAR2(60) NOT NULL
,A_PATERNO VARCHAR2(30) NOT NULL
,A_MATERNO VARCHAR2(30) NOT NULL
,F_NACIMIENTO DATE NOT NULL
,SEXO VARCHAR2(1) NOT NULL
,ACTIVA VARCHAR2(2) NOT NULL
,MRE_CVE_RECHAZO NUMBER(6)
,MRE_TIPO VARCHAR2(3)
,CURP VARCHAR2(18)
```

```
,FOLIO_DOCTO VARCHAR2(10)
,DOCTO_PROBATORIO VARCHAR2(16)
,F_ALTA_CURP DATE
,CVE_ALTA_RENAPO VARCHAR2(6)
,CVE_MODI_RENAPO VARCHAR2(6)
,USU_ALTA VARCHAR2(30) NOT NULL
,F_ALTA DATE NOT NULL
,USU_MODIF VARCHAR2(30)
,F_MODIF DATE
)
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.ID_REGISTRO IS 'Numero consecutivo de acuerdo
al lote.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.ACU_ID_LOTE IS 'Numero de lote.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.AFI_ID_FOLIO IS 'Numero de Folio que identifica
al afiliado.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.MUN_CVE_PAIS IS 'Clave del Pais'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.MUN_CVE_ESTADO IS 'Clave del Estado'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.MUN_CVE_MUNICIPIO IS 'Clave del Municipio'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.COD_RESUL_OPER IS 'Resultado de la operacion:
"01"=Aceptado; "02"=Rechazado.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.TIP_TRANSACCION IS 'Tipo de transaccion:
"007"=Alta; "008"=Cambio.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.STATUS_RENAPO IS 'Estatus de salida de Renapo.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.NOMBRE IS 'Nombre(s).'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.A_PATERNO IS 'Apellido Paterno.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.A_MATERNO IS 'Apellido Materno.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.F_NACIMIENTO IS 'Fecha de Nacimiento.'
/
```

```
COMMENT ON COLUMN AFI_REGISTROS_CURP.SEXO IS 'Seleccione: M(asculino)  o  F(emenino).'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.ACTIVA IS 'La CURP esta activa ?: S/N.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.MRE_CVE_RECHAZO IS 'Codigo del motivo de
rechazo'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.MRE_TIPO IS 'Rechazo para: "007"=Alta;
"008"=Cambio.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.CURP IS 'Clave unica de registro poblacional.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.FOLIO_DOCTO IS 'Folio del documento probatorio.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.DOCTO_PROBATORIO IS 'Documento probatorio.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.F_ALTA_CURP IS 'Fecha de alta CURP.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.CVE_ALTA_RENAPO IS 'Clave del archivo de alta,
asignada por Renapo'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.CVE_MODI_RENAPO IS 'Clave del archivo de
modificacion, asignada por Renapo'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.USU_ALTA IS 'Usuario que dio de alta el registro.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.F_ALTA IS 'Fecha en la que se dio de alta el
registro.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.USU_MODIF IS 'Usuario que modifico el registro.'
/

COMMENT ON COLUMN AFI_REGISTROS_CURP.F_MODIF IS 'Fecha en que se modifico el registro.'
/

PROMPT Creating Table 'AFI_ARCHIVOS_CURP'
CREATE TABLE AFI_ARCHIVOS_CURP
 (ID_LOTE NUMBER(12) NOT NULL
 ,F_LOTE DATE NOT NULL
 ,TIPO_OPERACION NUMBER(2) NOT NULL
 ,TIPO_SERVICIO NUMBER(2) NOT NULL
 ,REG_PROCESADO NUMBER(9) NOT NULL
 ,F_RECEP_LOTE DATE
 ,USU_ALTA VARCHAR2(30) NOT NULL
```

```
,F_ALTA DATE NOT NULL
,USU_MODIF VARCHAR2(30)
,F_MODIF DATE
)
/

COMMENT ON COLUMN AFI_ARCHIVOS_CURP.ID_LOTE IS 'Numero de lote.'
/

COMMENT ON COLUMN AFI_ARCHIVOS_CURP.F_LOTE IS 'Fecha del lote.'
/

COMMENT ON COLUMN AFI_ARCHIVOS_CURP.TIPO_OPERACION IS 'Identificador de operacion.'
/

COMMENT ON COLUMN AFI_ARCHIVOS_CURP.TIPO_SERVICIO IS 'Identificador del servicio.'
/

COMMENT ON COLUMN AFI_ARCHIVOS_CURP.REG_PROCESADO IS 'Numero de registros procesados'
/

COMMENT ON COLUMN AFI_ARCHIVOS_CURP.F_RECEP_LOTE IS 'Fecha en que se recibio el lote.'
/

COMMENT ON COLUMN AFI_ARCHIVOS_CURP.USU_ALTA IS 'Usuario que dio de alta el registro.'
/

COMMENT ON COLUMN AFI_ARCHIVOS_CURP.F_ALTA IS 'Fecha en que se dio de alta el registro.'
/

COMMENT ON COLUMN AFI_ARCHIVOS_CURP.USU_MODIF IS 'Usuario que modifico el registro.'
/

COMMENT ON COLUMN AFI_ARCHIVOS_CURP.F_MODIF IS 'Fecha en que se modifico el registro.'
/

PROMPT Creating Table 'AFI_PAISES'
CREATE TABLE AFI_PAISES
 (CVE_PAIS VARCHAR2(6) NOT NULL
 ,PAIS VARCHAR2(30) NOT NULL
 ,NACIONALIDAD VARCHAR2(30)
 )
/

COMMENT ON COLUMN AFI_PAISES.CVE_PAIS IS 'Clave del Pais'
/

COMMENT ON COLUMN AFI_PAISES.PAIS IS 'Nombre del Pais'
/

COMMENT ON COLUMN AFI_PAISES.NACIONALIDAD IS 'Nacionalidad.'
/

PROMPT Creating Table 'AFI_MUNICIPIOS'
CREATE TABLE AFI_MUNICIPIOS
```

```
(EDO_CVE_PAIS VARCHAR2(6) NOT NULL
,EDO_CVE_ESTADO VARCHAR2(6) NOT NULL
,CVE_MUNICIPIO VARCHAR2(6) NOT NULL
,MUNICIPIO VARCHAR2(60) NOT NULL
)
/

COMMENT ON COLUMN AFI_MUNICIPIOS.EDO_CVE_PAIS IS 'Clave del Pais'
/

COMMENT ON COLUMN AFI_MUNICIPIOS.EDO_CVE_ESTADO IS 'Clave del Estado'
/

COMMENT ON COLUMN AFI_MUNICIPIOS.CVE_MUNICIPIO IS 'Clave del Municipio'
/

COMMENT ON COLUMN AFI_MUNICIPIOS.MUNICIPIO IS 'Nombre del Municipio o Delegacion'
/

PROMPT Creating Table 'AFI_AFILIADOS'
CREATE TABLE AFI_AFILIADOS
(ID_FOLIO NUMBER(12) NOT NULL
,MUN_CVE_PAIS_DIR VARCHAR2(6) NOT NULL
,MUN_CVE_ESTADO_DIR VARCHAR2(6) NOT NULL
,MUN_CVE_MUNICIPIO_DIR VARCHAR2(6) NOT NULL
,MUN_CVE_PAIS_ORI VARCHAR2(6) NOT NULL
,MUN_CVE_ESTADO_ORI VARCHAR2(6) NOT NULL
,MUN_CVE_MUNICIPIO_ORI VARCHAR2(6) NOT NULL
,TIPO_PERSONA VARCHAR2(1) NOT NULL
,NOMBRE VARCHAR2(60) NOT NULL
,A_PATERNO VARCHAR2(30) NOT NULL
,A_MATERNO VARCHAR2(30) NOT NULL
,F_NACIMIENTO DATE NOT NULL
,SEXO VARCHAR2(1) DEFAULT 'M' NOT NULL
,CALLE VARCHAR2(60) NOT NULL
,COLONIA VARCHAR2(60) NOT NULL
,CP VARCHAR2(6)
,NSS NUMBER(11)
,RFC VARCHAR2(13)
,CURP VARCHAR2(18)
,TEL_CASA VARCHAR2(20)
,TEL_OFICINA VARCHAR2(20)
,F_SOLICITUD DATE
,USU_ALTA VARCHAR2(30) NOT NULL
,F_ALTA DATE NOT NULL
,USU_MODIF VARCHAR2(30)
,F_MODIF DATE
)
/

COMMENT ON COLUMN AFI_AFILIADOS.ID_FOLIO IS 'Numero de Folio que identifica al afiliado.'
/

COMMENT ON COLUMN AFI_AFILIADOS.MUN_CVE_PAIS_DIR IS 'Clave del Pais'
/
```

```
COMMENT ON COLUMN AFI_AFILIADOS.MUN_CVE_ESTADO_DIR IS 'Clave del Estado'
/

COMMENT ON COLUMN AFI_AFILIADOS.MUN_CVE_MUNICIPIO_DIR IS 'Clave del Municipio'
/

COMMENT ON COLUMN AFI_AFILIADOS.MUN_CVE_PAIS_ORI IS 'Clave del Pais'
/

COMMENT ON COLUMN AFI_AFILIADOS.MUN_CVE_ESTADO_ORI IS 'Clave del Estado'
/

COMMENT ON COLUMN AFI_AFILIADOS.MUN_CVE_MUNICIPIO_ORI IS 'Clave del Municipio'
/

COMMENT ON COLUMN AFI_AFILIADOS.TIPO_PERSONA IS 'Indicar si es persona F(isica) o
M(oral).'
/

COMMENT ON COLUMN AFI_AFILIADOS.NOMBRE IS 'Nombre del afiliado.'
/

COMMENT ON COLUMN AFI_AFILIADOS.A_PATERNO IS 'Apellido Paterno del afiliado.'
/

COMMENT ON COLUMN AFI_AFILIADOS.A_MATERNO IS 'Apellido Materno del afiliado.'
/

COMMENT ON COLUMN AFI_AFILIADOS.F_NACIMIENTO IS 'Fecha de Nacimiento.'
/

COMMENT ON COLUMN AFI_AFILIADOS.SEXO IS 'M(asculino) o F(emenino).'
/

COMMENT ON COLUMN AFI_AFILIADOS.CALLE IS 'Domicilio: Calle.'
/

COMMENT ON COLUMN AFI_AFILIADOS.COLONIA IS 'Domicilio: Colonia.'
/

COMMENT ON COLUMN AFI_AFILIADOS.CP IS 'Codigo Postal.'
/

COMMENT ON COLUMN AFI_AFILIADOS.NSS IS 'Numero de Seguro Social.'
/

COMMENT ON COLUMN AFI_AFILIADOS.RFC IS 'Registro Federal de Contribuyentes.'
/

COMMENT ON COLUMN AFI_AFILIADOS.CURP IS 'Clave unica de registro poblacional.'
/

COMMENT ON COLUMN AFI_AFILIADOS.TEL_CASA IS 'Telefono particular (casa o celular).'
/
```

```
COMMENT ON COLUMN AFI_AFILIADOS.TEL_OFICINA IS 'Telefono oficina.'
/

COMMENT ON COLUMN AFI_AFILIADOS.F_SOLICITUD IS 'Fecha de la solicitud.'
/

COMMENT ON COLUMN AFI_AFILIADOS.USU_ALTA IS 'Usuario que dio de alta el registro.'
/

COMMENT ON COLUMN AFI_AFILIADOS.F_ALTA IS 'Fecha en que se dio de alta el registro.'
/

COMMENT ON COLUMN AFI_AFILIADOS.USU_MODIF IS 'Usuario que modifico el registro.'
/

COMMENT ON COLUMN AFI_AFILIADOS.F_MODIF IS 'Fecha en que se modifico el registro.'
/

PROMPT Creating Table 'AFI_ESTADOS'
CREATE TABLE AFI_ESTADOS
 (PAI_CVE_PAIS VARCHAR2(6) NOT NULL
 ,CVE_ESTADO VARCHAR2(6) NOT NULL
 ,ESTADO VARCHAR2(30) NOT NULL
 )
/

COMMENT ON COLUMN AFI_ESTADOS.PAI_CVE_PAIS IS 'Clave del Pais'
/

COMMENT ON COLUMN AFI_ESTADOS.CVE_ESTADO IS 'Clave del Estado'
/

COMMENT ON COLUMN AFI_ESTADOS.ESTADO IS 'Nombre del Estado'
/

PROMPT Creating Table 'AFI_MOTIVOS_RECHAZO'
CREATE TABLE AFI_MOTIVOS_RECHAZO
 (TIPO VARCHAR2(3) NOT NULL
 ,CVE_RECHAZO NUMBER(6) NOT NULL
 ,DESCRIPCION VARCHAR2(60) NOT NULL
 )
/

COMMENT ON COLUMN AFI_MOTIVOS_RECHAZO.TIPO IS 'Rechazo para: "007"=Alta; "008"=Cambio.'
/

COMMENT ON COLUMN AFI_MOTIVOS_RECHAZO.CVE_RECHAZO IS 'Codigo del motivo de rechazo'
/

COMMENT ON COLUMN AFI_MOTIVOS_RECHAZO.DESCRIPCION IS 'Descripcion del motivo de rechazo.'
/
```

Código del archivo DLL_AFIL.IND.

```
-- e:\afiliacion\Script\Dll_Afil.ind
--
-- Generated for Oracle 8 on Sun Jan 14  12:10:34 2001 by Server Generator 6.0.3.3.0

PROMPT Creating Index 'RCU_ACU_FK_I'
CREATE INDEX RCU_ACU_FK_I ON AFI_REGISTROS_CURP
 (ACU_ID_LOTE)
/

PROMPT Creating Index 'RCU_AFI_FK_I'
CREATE INDEX RCU_AFI_FK_I ON AFI_REGISTROS_CURP
 (AFI_ID_FOLIO)
/

PROMPT Creating Index 'RCU_MUN_FK_I'
CREATE INDEX RCU_MUN_FK_I ON AFI_REGISTROS_CURP
 (MUN_CVE_PAIS
 ,MUN_CVE_ESTADO
 ,MUN_CVE_MUNICIPIO)
/

PROMPT Creating Index 'RCU_MRE_FK_I'
CREATE INDEX RCU_MRE_FK_I ON AFI_REGISTROS_CURP
 (MRE_CVE_RECHAZO
 ,MRE_TIPO)
/

PROMPT Creating Index 'MUN_EDO_FK_I'
CREATE INDEX MUN_EDO_FK_I ON AFI_MUNICIPIOS
 (EDO_CVE_PAIS
 ,EDO_CVE_ESTADO
 ,CVE_MUNICIPIO)
/

PROMPT Creating Index 'AFI_MUN_FK_I'
CREATE INDEX AFI_MUN_FK_I ON AFI_AFILIADOS
 (MUN_CVE_PAIS_DIR
 ,MUN_CVE_ESTADO_DIR
 ,MUN_CVE_MUNICIPIO_DIR)
/

PROMPT Creating Index 'AFI_MUN_DE_ORIGEN_FK_I'
CREATE INDEX AFI_MUN_DE_ORIGEN_FK_I ON AFI_AFILIADOS
 (MUN_CVE_PAIS_ORI
 ,MUN_CVE_ESTADO_ORI
 ,MUN_CVE_MUNICIPIO_ORI)
/

PROMPT Creating Index 'EDO_PAI_FK_I'
CREATE INDEX EDO_PAI_FK_I ON AFI_ESTADOS
 (PAI_CVE_PAIS)
/
```

Código del archivo DLL_AFIL.CON.

```
-- e:\afiliacion\Script\Dll_Afil.con
--
-- Generated for Oracle 8 on Sun Jan 14  12:10:34 2001 by Server Generator 6.0.3.3.0

PROMPT Creating Primary Key on 'AFI_REGISTROS_CURP'
ALTER TABLE AFI_REGISTROS_CURP
 ADD CONSTRAINT RCU_PK PRIMARY KEY
  (ACU_ID_LOTE
  ,ID_REGISTRO)
/

PROMPT Creating Primary Key on 'AFI_ARCHIVOS_CURP'
ALTER TABLE AFI_ARCHIVOS_CURP
 ADD CONSTRAINT ACU_PK PRIMARY KEY
  (ID_LOTE)
/

PROMPT Creating Primary Key on 'AFI_PAISES'
ALTER TABLE AFI_PAISES
 ADD CONSTRAINT PAI_PK PRIMARY KEY
  (CVE_PAIS)
/

PROMPT Creating Primary Key on 'AFI_MUNICIPIOS'
ALTER TABLE AFI_MUNICIPIOS
 ADD CONSTRAINT MUN_PK PRIMARY KEY
  (CVE_MUNICIPIO
  ,EDO_CVE_ESTADO
  ,EDO_CVE_PAIS)
/

PROMPT Creating Primary Key on 'AFI_AFILIADOS'
ALTER TABLE AFI_AFILIADOS
 ADD CONSTRAINT AFI_PK PRIMARY KEY
  (ID_FOLIO)
/

PROMPT Creating Primary Key on 'AFI_ESTADOS'
ALTER TABLE AFI_ESTADOS
 ADD CONSTRAINT EDO_PK PRIMARY KEY
  (CVE_ESTADO
  ,PAI_CVE_PAIS)
/

PROMPT Creating Primary Key on 'AFI_MOTIVOS_RECHAZO'
ALTER TABLE AFI_MOTIVOS_RECHAZO
 ADD CONSTRAINT MRE_PK PRIMARY KEY
  (CVE_RECHAZO
  ,TIPO)
/

PROMPT Creating Foreign Keys on 'AFI_REGISTROS_CURP'
ALTER TABLE AFI_REGISTROS_CURP ADD CONSTRAINT
 RCU_ACU_FK FOREIGN KEY
```

```
 (ACU_ID_LOTE) REFERENCES AFI_ARCHIVOS_CURP
 (ID_LOTE) ADD CONSTRAINT
RCU_AFI_FK FOREIGN KEY
 (AFI_ID_FOLIO) REFERENCES AFI_AFILIADOS
 (ID_FOLIO) ADD CONSTRAINT
RCU_MUN_FK FOREIGN KEY
 (MUN_CVE_MUNICIPIO
 ,MUN_CVE_ESTADO
 ,MUN_CVE_PAIS) REFERENCES AFI_MUNICIPIOS
 (CVE_MUNICIPIO
 ,EDO_CVE_ESTADO
 ,EDO_CVE_PAIS) ADD CONSTRAINT
RCU_MRE_FK FOREIGN KEY
 (MRE_CVE_RECHAZO
 ,MRE_TIPO) REFERENCES AFI_MOTIVOS_RECHAZO
 (CVE_RECHAZO
 ,TIPO)
/

PROMPT Creating Foreign Keys on 'AFI_MUNICIPIOS'
ALTER TABLE AFI_MUNICIPIOS ADD CONSTRAINT
 MUN_EDO_FK FOREIGN KEY
 (EDO_CVE_ESTADO
 ,EDO_CVE_PAIS) REFERENCES AFI_ESTADOS
 (CVE_ESTADO
 ,PAI_CVE_PAIS)
/

PROMPT Creating Foreign Keys on 'AFI_AFILIADOS'
ALTER TABLE AFI_AFILIADOS ADD CONSTRAINT
 AFI_MUN_DIR_FK FOREIGN KEY
 (MUN_CVE_MUNICIPIO_DIR
 ,MUN_CVE_ESTADO_DIR
 ,MUN_CVE_PAIS_DIR) REFERENCES AFI_MUNICIPIOS
 (CVE_MUNICIPIO
 ,EDO_CVE_ESTADO
 ,EDO_CVE_PAIS) ADD CONSTRAINT
 AFI_MUN_ORIGEN_FK FOREIGN KEY
 (MUN_CVE_MUNICIPIO_ORI
 ,MUN_CVE_ESTADO_ORI
 ,MUN_CVE_PAIS_ORI) REFERENCES AFI_MUNICIPIOS
 (CVE_MUNICIPIO
 ,EDO_CVE_ESTADO
 ,EDO_CVE_PAIS)
/

PROMPT Creating Foreign Keys on 'AFI_ESTADOS'
ALTER TABLE AFI_ESTADOS ADD CONSTRAINT
 EDO_PAI_FK FOREIGN KEY
 (PAI_CVE_PAIS) REFERENCES AFI_PAISES
 (CVE_PAIS)
/
```

ANEXO D
FORMAS Y REPORTES

En la Figura D.1 se muestra la pantalla inicial, la cual contiene el menú con el que se podrá llamar a cada módulo (forma o reporte) generado con Oracle Designer/2000. Esta primer pantalla y el menú fueron creados con Developer/2000.

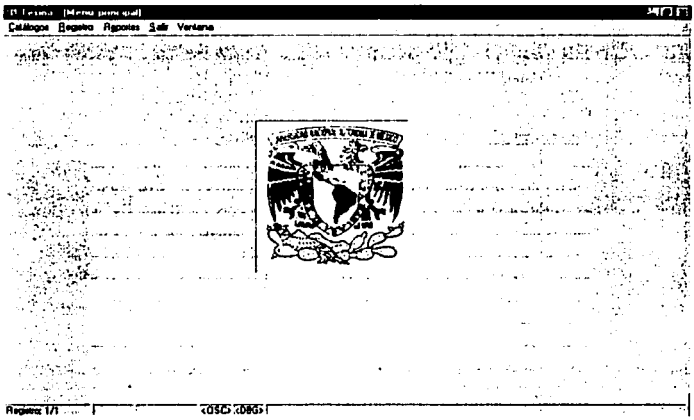

Figura D.1. Pantalla inicial que contiene el menú para llamar a cada módulo.

La estructura del menú, con las opciones para llamar a cada módulo, se muestra en la Figura D.2.

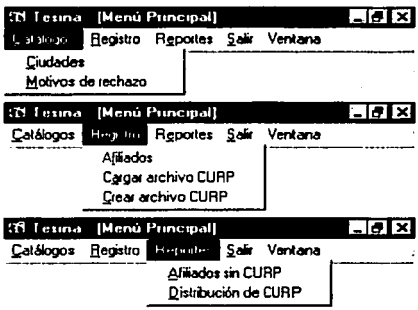

Figura D.2. Estructura del menú.

Ciudades.

Al seleccionar la opción "*Catálogos/Ciudades*", se mostrará la pantalla de la Figura D.3 que corresponde al módulo "AFI_C001". Esta pantalla cuenta con tres carpetas para capturar la información relacionada a los catálogos de: Países, Estados y Municipios.

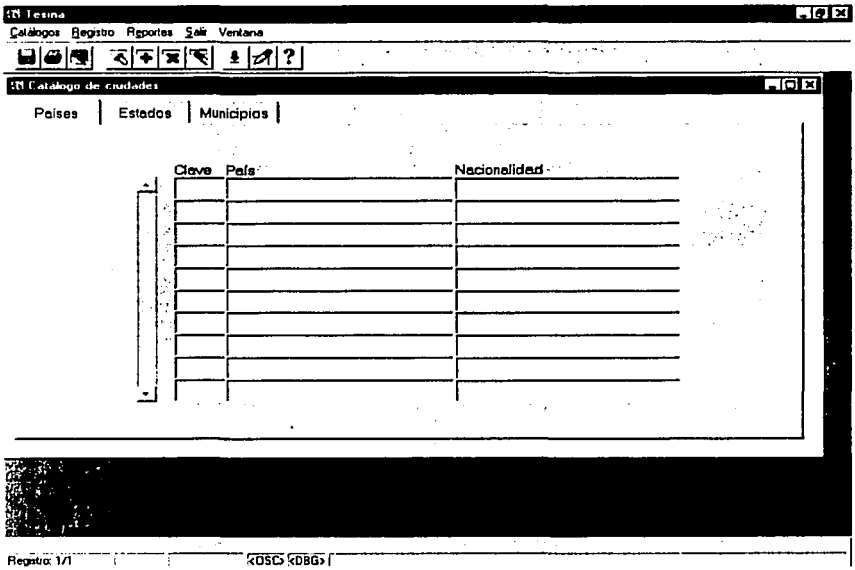

Figura D.3. Pantalla para el mantenimiento de los catálogos de: Países, Estados y Municipios.

Motivos de rechazo.

Con la opción "*Catálogos/Motivos de rechazo*", se mostrará la pantalla de la Figura D.4 que corresponde al módulo "AFI_C002".
El archivo con los registros de distribución de CURP podrá contener tanto registros aceptados como registros rechazados. Cada registro rechazado contiene una clave para identificar el motivo por el cual fue rechazado. Esta pantalla podrá ser utilizada para capturar o actualizar el catálogo de "motivos de rechazo".

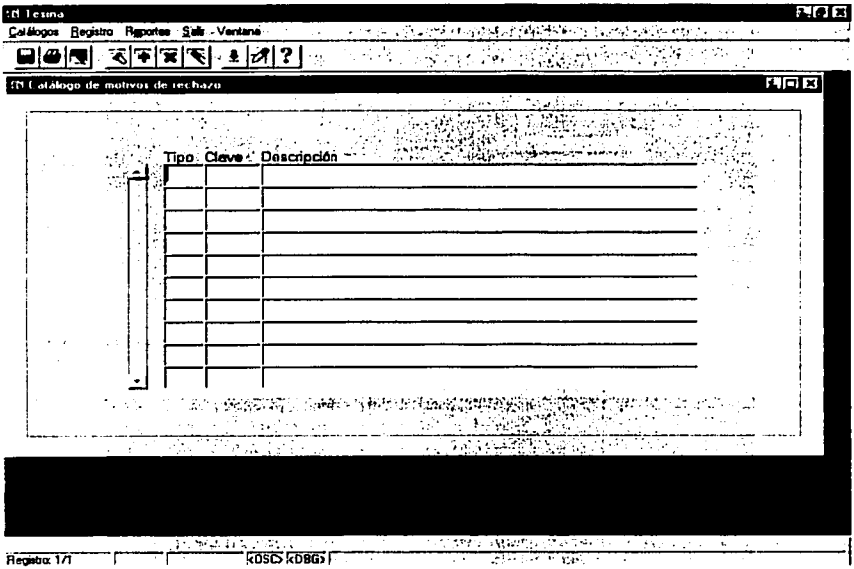

Figura D.4. Pantalla para el catálogo de motivos de rechazo.

Registro de afiliados.

Con la opción "*Registro/Afiliados*", se mostrará la pantalla de la Figura D.5 que corresponde al módulo "AFI_F001 ", generado desde Designer/2000. Esta pantalla podrá ser utilizada para capturar los datos generales de nuevos afiliados.

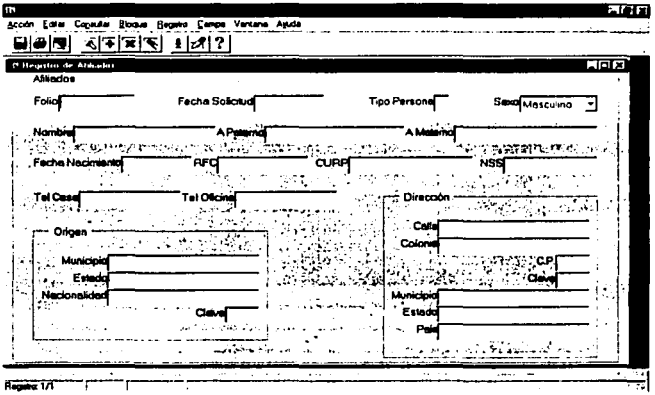

Figura D.5. Pantalla del módulo "AFI_F001", generado desde Designer/2000.

La forma "AFI_F001" fue modificada en Developer/2000 para mejorar su apariencia y funcionalidad, como se muestra en la Figura D.6.

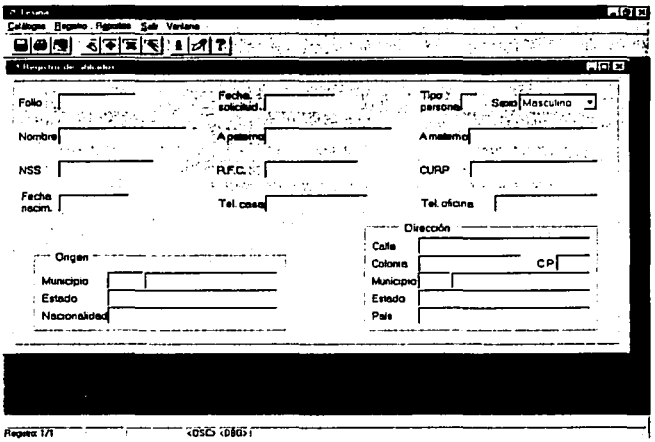

Figura D.6. Pantalla del módulo "AFI_F001", modificado en Developer/2000.

Cargar archivo CURP.

La Figura D.7 muestra la pantalla del módulo "AFI_F002", generado desde Designer/2000, y corresponde a la opción "*Registro\Cargar archivo CURP*".

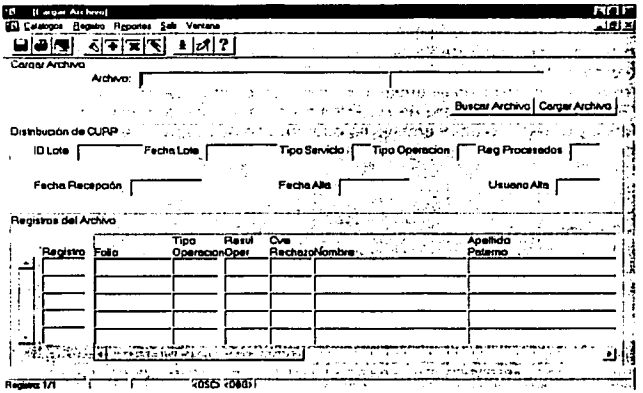

Figura D.7. Pantalla del módulo "AFI_F002", generado desde Designer/2000.

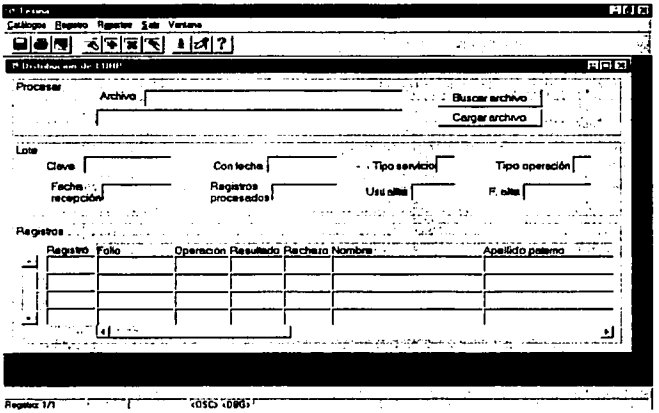

Figura D.8. Pantalla del módulo "AFI_F002", modificado en Developer/2000.

La forma "AFI_F002" también fue modificada en Developer/2000 para mejorar su apariencia y funcionalidad, como se muestra en la Figura D.8. El primer bloque (Procesar) es para realizar la

búsqueda del archivo extensión ".002" que contiene los registros con el resultado de la asignación de la CURP y que se desea cargar en la base de datos. Los siguientes dos bloques son para mostrar los datos relacionados al archivo (Lote) y el contenido del archivo (Registros); estos dos bloques son sólo de consulta.
Cuando se realiza la carga del archivo se valida que el lote exista en la base de datos (tabla "AFI_ARCHIVOS_CURP") y que el número de registros corresponda al archivo que se envió solicitando la asignación de la CURP para actualizar los datos del registro correspondiente al lote. Los registros correspondientes al detalle del archivo, es decir, los registros de los afiliados, serán buscados en la base de datos (tabla "AFI_REGISTROS_CURP") para registrar el resultado de la aceptación o el rechazo en la asignación de la CURP.
En caso de existir algún error en el archivo, se rechaza el lote y se reenvía nuevamente el archivo de solicitud. Por ejemplo, el número de registros no corresponde al registrado en la base de datos o la longitud de algún registro en el archivo es diferente de 400 caracteres.

Crear archivo CURP.

La Figura D.9 muestra la última forma generada con Designer/2000 (módulo AFI_F003) y corresponde a la opción "*Registro\Crear archivo CURP*".

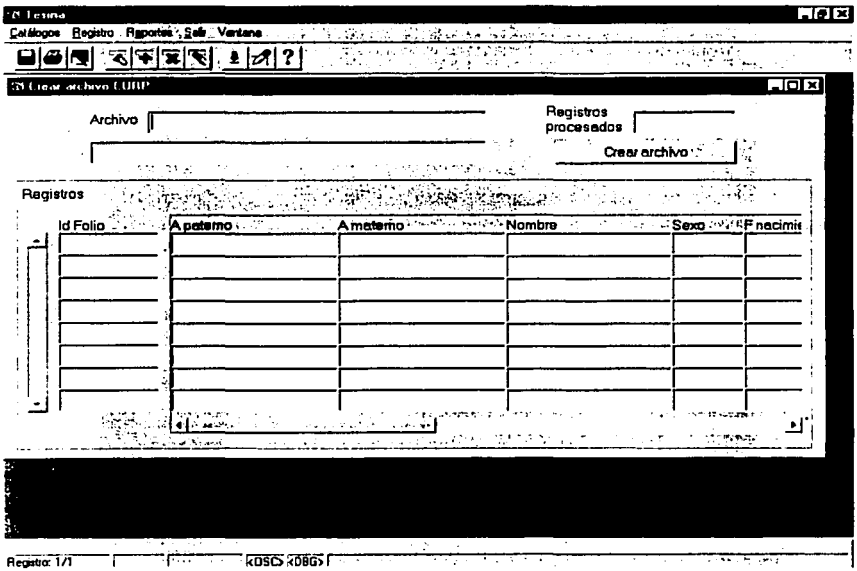

Figura D.9. Pantalla del módulo "AFI_F003". Muestra el nombre del archivo generado, automáticamente por la aplicación, y los datos de los afiliados que serán incluidos en el archivo, sólo de consulta.

En esta pantalla se podrán generar los archivos (extensión ".001") para solicitar la asignación de la CURP de los nuevos afiliados que no cuentan con esta clave. Los archivos generados serán depositados en la ruta: "*E:\AFILIACION\ARCHIVOS\SOLICITUD*".
Los detalles de cada archivo generado serán insertados en la base de datos (tabla "AFI_ARCHIVOS_CURP"); además, se realizará una copia en la base de datos (tabla "AFI_REGISTROS_CURP") de los registros correspondientes a los afiliados que estarán en espera de la asignación de la CURP y que serán incluidos en el archivo de solicitud.

Reporte de afiliados sin CURP.

En la Figura D.10. se muestra el reporte de la opción "*Reportes/Afiliados sin CURP*". Corresponde al módulo "AFI_R001" y fue generado desde Designer/2000.

AFILIADOS SIN CURP

Page 1 of 1

Afiliados

Id folio	Nombre	A paterno	A materno	F solicitud	Sexo
5	LUIS	GUZMAN	???	10-MAY-00	M
6	AGUSTIN	DIAZ DEL CASTILLO	ELGUERO	10-MAY-00	M
7	LORENZO EMILIO	ALONSO	MENDOZA	10-MAY-00	M
8	VERONICA	OLIVARES	VIVEROS	10-MAY-00	F
9	PEDRO	SERNA	MENDOZA	10-MAY-00	M
10	FRANCISCO JAVIER	VEGA	CENTENO	10-MAY-00	M
11	ANTONIO	AYALA	SALAZAR	10-MAY-00	M
12	UBALDO	SANTOS	VEGA	10-MAY-00	M
13	MONICA SUSANA	REYES	CARDENAS	10-MAY-00	F
14	MIRIAM	GABRIEL	CRUZ	10-MAY-00	F

Figura D.10. Reporte para mostrar los afiliados que no cuentan con la CURP.

Reporte de distribución de CURP.

Por ultimo, el reporte de la opción "*Reportes/Distribución de CURP*" se muestra en la Figura.D.11. Corresponde al módulo "AFI_R002" y fue generado desde Designer/2000.

DISTRIBUCION DE CURP

Page 1 of 1

Archivos Curp

ID lote: 200100000001 Fecha lote: 15-MAY-00 Usuario ejecutó: JULIO Fecha ejecución: 16-MAY-00

Reg procesados: 10 Fecha recepción: 16-MAY-00

Registros Curp

Registro	Resultado distribución	Tipo operación	Status registro	Folio situado	Nombre(s)	Apellido paterno	Apellido materno	Sexo	Curp	Folio docto	Docto probatorio	Fecha Curp	Activa
2	01	007	12	6	AGUSTIN	DIAZ DEL CASTILLO	ELGUERO	M	DEAS 00415 HDFZL 008	00012 59966	0900050000 400284	10-MAY-00	S
3	01	007	12	7	LORENZO EMILIO	ALONSO	MENDOZA	M	AOML$ 31005 HDFLN R04	00025 74999	0900054001 300398	10-MAY-00	S
4	01	007	12	8	VERONICA	OLIVARES	VIVEROS	F	OIVV7 71117 MDFLV R03	00056 59581	0900150482 036479	10-MAY-00	S
5	01	007	12	9	PEDRO	SERNA	MENDOZA	M	SEMP3 91023 HFLRL D06	00000 63623	0190330100 005680	10-MAY-00	S
6	01	007	12	10	FRANCISCO JAVIER	VEGA	CENTENO	M	VECF4 81220 HJCGN R08	00039 04036	1402348000 101082	10-MAY-00	S
7	01	007	12	11	ANTONIO	AYALA	SALAZAR	M	AASA3 60525 HJCYL N06	00035 27481	1403336000 100207	10-MAY-00	S
8	01	007	12	12	UBALDO	SANTOS	VEGA	M	SAVU3 60518 HPLNG B08	00049 94601	2110336000 100020	10-MAY-00	S
9	01	007	12	13	MONICA SUSANA	REYES	CARDENAS	F	RECM7 41006 MMCYR N02	00056 23640	1510474000 100575	10-MAY-00	S
10	01	007	12	14	MIRIAM	GABRIEL	CRUZ	F	GACM9 30121 MCSBR R04	00056 10386	0701063000 100120	10-MAY-00	S
1	02	007	11	5	LUIS	GUZMAN	???	M					N

Figura D.11. Reporte del módulo "AFI_R002", generado por Designer/2000.

Para ejecutar este reporte es necesario proporcionar el número de lote que corresponde al archivo de distribución de CURP que se desea consultar y que se cargo previamente en la base de datos. El reporte es de tipo "maestro - detalle": en el bloque del "maestro" se muestra la información relacionada al archivo de distribución de CURP, tabla "AFI_ARCHIVOS_CURP"; y en el bloque de "detalle" se muestran los registros correspondientes al archivo, tabla "AFI_REGISTROS_CURP".

El reporte "AFI_R002" fue modificado en Developer/2000 para mejorar su apariencia y funcionalidad, como se muestra en la Figura D.12.

DISTRIBUCION DE CURP

Pag 1 de 1

ARCHIVO

Lote 20010000001 — Reg procesados 10 — Usuario ale — ALEJO

Fecha envio 15-MAY-00 — Fecha recepcion 16-MAY-00 — Fecha ale — 16-MAY-00

REGISTROS

Folio solicitud	Nombre(s)	Apellido paterno	Apellido materno	Sexo	CURP	Folio documento	Documento probatorio	Fecha ale Curp	Activo	Status registro	Registro del archivo
Tipo de operacion: Alta (007)											
Aceptados (01)											
6	AGUSTIN	DIAZ DEL CASTILLO	ELIQUEPO	M	DIEA500415HDFZLG08	0001259966	0900050000400294	10-MAY-00	S	12	2
7	LORENZO EMILIO	ALONSO	MENDOZA	M	AOML531005HDFLNR04	0002574999	0900054001300396	10-MAY-00	S	12	3
8	VERONICA	OLIVARES	VIVEROS	F	OIVV771117MDFLVR03	0005659581	0090150482036479	10-MAY-00	S	12	4
9	PEDRO	SERNA	MENDOZA	M	SEMP391023HNLRLD06	0000083623	0190330100005660	10-MAY-00	S	12	5
10	FRANCISCO JAVIER	VEGA	CENTENO	M	VECF481220HJCGNP08	0003904036	1402348000101052	10-MAY-00	S	12	6
11	ANTONIO	AYALA	SALAZAR	M	AASA360825HJCYLN08	0003527481	1403336000100207	10-MAY-00	S	12	7
12	UBALDO	SANTOS	VEGA	M	SAVU360516HPLNGB08	0004994601	2110336000100020	10-MAY-00	S	12	8
13	MONICA SUSANA	REYES	CARDENAS	F	RECM741006MMCYRN02	0005623640	1510474000100575	10-MAY-00	S	12	9
14	MIRIAM	GABRIEL	CRUZ	F	GACM630121MCSBRR04	0005610366	0731083000100120	10-MAY-00	S	12	10
Rechazados (02)											
5	LUIS	GUZMAN	???	M					N	11	1

Figura D.12. Reporte del módulo "AFI_R002", modificado en Developer/2000.

GLOSARIO

Almacén de datos (Datastore) Concepto de almacén temporal o permanente de datos lógicos, es usado por funciones o procesos. El *datastore* generalmente hace referencia a las entidades y usualmente contiene atributos de más de una entidad.

Arco Utilizado para identificar cuando dos o más relaciones de la misma entidad pueden ser mutuamente exclusivas, también llamado arco de exclusividad.

Atributo Un atributo es una propiedad que sirve para caracterizar, identificar, clasificar, cuantificar o expresar el estado de la entidad. Hay tres tipos de atributos: los que identifican a cada ocurrencia de identidad (identificador único o llave primaria); los que describen cada ocurrencia de la entidad; los que se refieren a ocurrencias de otra entidad (clave ajena o llave foránea).

Base de Datos (BD) Colección arbitraria de tablas o archivos bajo el control de un sistema administrador de base de datos, es decir, una base de datos es una colección integrada de datos almacenados en distintos tipos de registros, de forma que sean accesibles para múltiples aplicaciones. La interrelación de los registros se obtiene de las relaciones entre los datos, no de su lugar de almacenamiento físico.

Base de datos distribuidas Un sistema distribuido de base de datos consiste en una única base de datos en la que los datos, tablas y consultas se almacenan físicamente en distintos ordenadores, comunicándose mediante redes. A cada uno de los ordenadores de una red se le llama nodo. Una base de datos distribuida no es una sola base de datos en la que sus datos están repartidos en distintos nodos, sino un conjunto de base de datos locales que comparten información teniendo la apariencia de una base de datos única para todos los nodos.

Campo Cada una de las columnas de una tabla; utilizado para definir cada elemento de dato que formará parte de un registro. Este puede ser de tipo carácter, fecha, numérico u otro tipo de dato, y puede ser opcional u obligatorio.

CASE (Computer Aided Systems Engineering) Estas siglas se utilizan para hacer referencia a la ingeniería de sistemas asistida por computadora. Es la combinación de diagramadores, generadores, un administrador de proyectos y otras herramientas (software) asistidas por computadora para el desarrollo y mantenimiento de sistemas.

Ciclo de vida del desarrollo de sistemas (SDLC, System Development Life Cycle) El SDLC es el conjunto de actividades que emprenden los analistas y diseñadores para desarrollar e implantar un sistema de información, incluye la investigación preliminar, la recolección de datos junto con la determinación de requerimientos, el diseño de un sistema, el desarrollo de software, la prueba de los sistemas y su implantación.

DDL (Data Definition Language) Es el lenguaje de definición de datos, es decir, son los comandos utilizados para definir los objetos en una base de datos. Por ejemplo, los comandos para crear, modificar o borrar una tabla de la base de datos.

Diagrama Entidad Relación (DER) Es un modelo exacto de las necesidades de información de una organización, que actuará como marco de trabajo para el desarrollo de sistemas nuevos o para modificar el existente. Proporciona un modelo independiente de cualquier almacenamiento de datos y cualquier método de acceso. El diagrama representa gráficamente entidades, las relaciones entre estas y los atributos de cada entidad.

Diccionario de datos Almacena las definiciones y propiedades de los objetos que configuran la base de datos. En una herramienta CASE, todas las definiciones de los elementos de un sistema (diagramas, procesos, entidades, tablas, etc.) están descritos en forma detallada en el diccionario de datos.

Disparadores (triggers) Los disparadores son un conjunto de acciones que se ejecutan automáticamente cuando se añade, modifica o borra el contenido de un registro o todo éste. Son, además, los responsables del mantenimiento de la integridad de la base de datos, sobre todo en lo referente a la integridad referencial. Para diseñar un disparador hay que tener en cuenta las condiciones que lo van a ejecutar y las acciones que van a realizar cuando se ejecute.

DML (Data Manipulation Language) Es el lenguaje de manipulación de datos, es decir, son los comandos que se utilizan para acceder y manejar la información almacenada en una base de datos. Por ejemplo, los comandos para actualizar o consultar los datos de una o varias tablas.

Dominio Es un conjunto de reglas de validación, restricciones de formato y otras propiedades que se aplican a un grupo de atributos. Los atributos que se encuentran en el dominio están sujetos al mismo conjunto de validaciones. Por ejemplo: Un conjunto de todos los valores posibles para una columna o un rango.

Entidad Es un ente con significado real o conceptual, acerca del cual el negocio o sistema que se está modelando necesita guardar información. La entidad tiene atributos, que agrupan los datos más representativos para identificar las diferentes ocurrencias.

Flujo de datos (Dataflow) El nombre de un flujo de datos entre funciones de negocios, almacenamientos (datastore) y entidades externas.

Formato Es el tipo de dato que un atributo o columna puede representar, por ejemplo, un tipo de dato numérico, carácter, fecha u otro tipo de dato.

Función común Una función común es una copia de otra función conocida como maestra, que se usa cuando la misma función aparece en más de un lugar en un diagrama jerárquico funcional. Durante el análisis, el objetivo es eliminar las funciones idénticas (duplicadas) donde quiera que sea posible.

Función de negocio Una función es una actividad que es o será ejecutada, que permite cumplir

la misión del negocio y lograr sus objetivos, independientemente de como se realice. La función representa lo que un negocio hace o necesita hacer y no el cómo lo hace.

Función elemental Es una función que si comienza, debe ser completada exitosamente. De lo contrario, debe regresarse a su estado inicial de tal forma que los cambios no se realicen, como si nada hubiera ocurrido. No contiene pasos intermedios, por lo tanto, las funciones elementales siempre están en el nivel más bajo de un Diagrama Jerárquico Funcional (DJF) y no pueden ser descompuestas.

Identificador único Corresponde a cualquier combinación de atributos y/o relaciones que sirve, en todos los casos, para identificar únicamente una ocurrencia de una entidad. Un identificador único siempre será obligatorio.
O
Una o más columnas utilizadas para asegurar la existencia de registros únicos en una tabla.

Inconsistencia Una misma información está en varias tablas, y algunos de esos datos no concuerdan entre sí.

Indice (Index) Es una estructura opcional asociada con las tablas, la cual puede ser creada para acelerar la ejecución (performance) de una consulta, es decir, acelera el acceso a los registros de una tabla. Un índice es lógicamente y físicamente independiente de los datos, puede involucrar una o más columnas de una tabla, puede ser creado o borrado en cualquier momento sin afectar las tablas u otros índices.

Integridad Hay tres tipos de operaciones sobre una base de datos que se traduce en pérdida de la integridad: agregar, modificar o borrar datos, bien sea de forma accidental o intencionada. El administrador de la base de datos ha de ser capaz de crear las reglas de integridad necesarias para mantener consistente la información.

Integridad referencial La integridad referencial parte del supuesto que, si una tabla tiene un conjunto de campos que forman la llave primaria en otra tabla (llave foránea), será una tabla hija de la tabla que posee como llave primaria la llave foránea. La tabla hija no puede contener, en los campos que forma la llave foránea, valores que no existan en la tabla padre o valores nulos.

Llave foránea Una o más columnas en una tabla que forman la clave primaria en otra tabla. En el diagrama entidad relación, es el conjunto de atributos de una entidad que forman el identificador único en otra entidad.

Llave primaria Es el conjunto de columnas obligatorias en una tabla. La llave primaria es utilizada para identificar y obligar a tener registros únicos en una tabla. Sus características son: no puede contener valores nulos; ha de ser conocida y sencilla de crear; no ha de variar con el tiempo; y es el medio más usado para realizar consultas sobre una tabla.

Metodología CASE (CASE*Method) Es una estructura enfocada a la ingeniería de sistemas. Consiste en una serie de etapas, tareas, entregas y técnicas, que conducen a través de todos los pasos en el ciclo de vida de un sistema. Consta de siete etapas: Estrategia, análisis, diseño, construcción, documentación, transición y producción.

Normalización La normalización es un proceso que pretende conseguir tablas con una estructura óptima y eficaz. Por medio de la normalización, un conjunto de datos en un registro se reemplaza por varios registros que son más simples y predecibles y, por lo tanto, más manejables. La normalización se lleva a cabo por cuatro razones:

- Estructurar los datos de forma que se puedan representar las relaciones pertinentes entre los datos.
- Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consulta y reportes.
- Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.
- Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.

Nulo (Null) Un nulo es la ausencia de valor en una columna de un registro. Por ejemplo, un campo contiene valores nulos cuando no hay información en dicho campo, no es lo mismo un campo con un valor nulo que con un espacio en blanco.

Redundancia Una misma información que puede estar repetida en varias tablas.

Registro Conjunto completo de datos relacionados pertenecientes a una entrada. En una tabla, es la información contenida en cada fila. Por ejemplo, el registro de una persona podría estar formado por los siguientes campos: nombre, sexo, edad, etc.

Relación Es la asociación que existe entre las entidades y es binaria, en el sentido de que es siempre una asociación entre exactamente dos entidades; con una excepción, entre una entidad y ella misma (relación recursiva).

Sistema Administrador de Base de Datos (DBMS – Data Base Management System) Es un programa que tiende un puente entre las estructuras de archivos que guardan los datos y las estructuras de datos que representan las necesidades de datos de los usuarios. Proporciona la flexibilidad en el almacenamiento y recuperación de datos y producción de la información.

SQL (Structured Query Language) Es el estándar internacionalmente aceptado para sistemas relacionales. Los comandos de SQL se utilizan para crear, almacenar, cambiar, recuperar y mantener la información en una base de datos. Está formado por comandos que pueden agruparse en dos tipos: comandos DDL y comandos DML.

Tabla En un sistema de base de datos relacional, es una estructura formada por filas y columnas que sirven para organizar y almacenar la información de un mismo tipo de objeto. A cada fila le corresponde un registro. Las columnas son los atributos de la entidad y se corresponden con los campos. La tabla es la implementación de una entidad.

Vista Es una "tabla virtual" creada a partir de una o más tablas, vistas o mediante el calculo entre columnas de diversas tablas o vistas. Es el resultado de una consulta y podrá ser usada como una tabla.