CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS
INSTITUTO POLITECNICO NACIONAL

DEPARTAMENTO DE INGENIERIA ELECTRICA
SECCION DE COMPUTACION

TITULO

DISEÑO DE BASE DE DATOS CON EVEX ENTIDAD VINCULO EXTENDIDO PARA XWINDOWS

TESIS PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS
EN LA ESPECIALIDAD DE INGENIERIA ELECTRICA

QUE PRESENTA:

ING. PEDRO ENRIQUE ALDAY ECHAVARRIA

DIRECTOR DE TESIS:

DR. SERGIO VICTOR CHAPA VERGARA

México, DF. Febrero 1997

A mis padres: José Antonio y Dulce María

A mis hermanos: Ma. Elisa, Alberto Santiago y José Antonio

A mis sobrinos: Fabián y Alina

A mis tíos y primos

A Carla

AGRADECIMIENTOS

Quiero agradecer primero a mis padres Jose Antonio y Dulce María,a mi hermano Alberto Santiago y a mi tía Guadalupe por el apoyo recibido durante mis estudios.

A mi primo Salvador Campos Echavarría por proporcionarme los medios para la impresión del trabajo de tesis.

Agradezco a PEMEX-Exploración por dar las facilidades y recursos para la superación profesional de sus trabajadores especialmente al: Ing. Alberto Zárate Ortega, Ing. Hermilo Cruz Rivera, Ing. Adrián Díaz, Ing. Fernando López Arriaga, Ing. Rodolfo Juárez Gómez.

A CONACYT por otorgarme la beca de manutención y colegiatura durante el segundo año de mi estancia en el posgrado.

También agradezco a las personas que en el ámbito académico apoyaron en la culminación de mis créditos y del proyecto de tesis: Ing. Enrique Sandoval, Ing. Fernando Lerios, Dr. Ju Shiguang, M. en C. Noé Sierra Romero, M. en C. Cristóbal Villegas, M. en C. Guillermo Domínguez de León y M. en C. Oscar Olmedo Aguirre.

Por último quiero agradecer al Dr. Sergio Chapa Vergara no sólo por la oportunidad que me ofreció para colaborar con él en diversos proyectos, por la revisión y sugerencias al trabajo de tesis, sino también por el estímulo, enseñanza y amistad que me ha brindado.

OBJETIVO GENERAL

El proyecto de "Diseño de base de datos con EVEX" tiene como propósito implementar una herramienta automática de apoyo al diseño de bases de datos con el modelo entidad vínculo extendido de tal manera que conserve la expresividad del modelo en su fase conceptual, en las fases de diseño lógico y fisico, hasta finalmente incorporarla en un sistema manejador de base de datos.
Este propósito principal no se lograría sin la realización de dos objetivos complementarios.
Un objetivo consiste en dar a conocer la tareas que se realizan en cada fase de la metodología de diseño con entidad vínculo extendido (eve). Con lo que en el desarrollo del trabajo se explican los conceptos y elementos para la construcción de los diagramas eve; las reglas de transformación de los diagramas a esquemas relacionales; los algoritmos de normalización por síntesis usados en el diseño lógico; y la creación del lenguaje de definición de datos (ldd) para su diseño fisico.
El segundo y último objetivo complementario es el implementar un sistema manejador de base de datos (SMBD) en ambiente Xwindows para validar la traducción del diseño fisico y verificar con la manipulación de los datos, la base de datos generada con evex.

INDICE

INTRODUCCION

La tesis "Diseño de bases de datos con EVEX" propone una metodología de diseño de base de datos con el modelo entidad vínculo extendido que abarca las fases de diseño conceptual, lógico y físico, utilizando el apoyo de una herramienta de software asistida por computador (CASE, Computer Aided Software Engineering).

En la fase de diseño conceptual, la tarea fundamental es construir el modelo entidad vínculo extendido completo del problema. Para lo cual el diseñador requiere la identificación de los elementos del problema como elementos básicos de diagrama eve (entidad, atributo, vínculo o interrelación) y la consolidación de vistas parciales del problema.
La herramienta CASE evex, auxilia en el modelado conceptual con los módulos de edición de atributos y de diagramas eve.
En la fase de diseño lógico dos son las tareas importantes que se deben llevar a cabo, la transformación del diagrama entidad a esquemas de relación y la verificación de los esquemas relacionales como esquemas en tercera forma normal.
Con la herramienta CASE evex, se aplican los módulos de análisis y traducción de diagramas para la obtención automática de los esquemas relacionales. Con el empleo del normalizador de evex, se asegura que los esquemas relacionales resultantes de la transformación cumplan las tres primeras formas normales.
El diseño físico con el modelo entidad vínculo extendido continua con la transformación de los esquemas normalizados a especificaciones en un lenguaje de definición de datos (ldd) que puede posteriormente transformarse a estructuras de datos estrechamente ligados a la arquitectura de un sistema manejador específico.
La herramienta CASE evex utiliza el constructor del ldd para generar el lenguaje, no sql, entendible para el SMBD CdataX.
Para la creación de la instancia de la base de datos donde se valida con operaciones de manipulación de datos de la base de datos diseñada, nos apoyaremos en dos productos más de programación el compilador del ldd y el SMBD CdataX.
El compilador lee el lenguaje de definición, enlaza las estructuras particulares de la base de datos con los procedimientos del manejador de base de datos para la creación de la instancia.

El proyecto de tesis para lograr sus objetivos tuvo que tener trabajos antecedentes. En la fase de diseño conceptual son relevantes las aportaciones sobre editores basados en iconos e interfaces gráficas de tres tesis, la tesis de doctorado de Sergio Chapa [Chapa91], la tesis de maestría de Noé Sierra [Sierra95] y la tesis de maestría de Raúl Hernández [Hernan94]. En el diseño lógico toma aportaciones de los algoritmos de transformación de esquemas eve a relaciones de Raúl Hernández [Hernan94] y de los algoritmos de normalización de Sergio Chapa [Chapa90]. En el diseño físico se contó con la implementación de un SMBD por Fernando Fiorentino [Fioren96] basado en las ideas del sistema manejador Cdata de Al Stevens [Steven87].

El desarrollo de los capítulos de la tesis se inicia con un capítulo enfocado a la metodología de diseño de base de datos. De lo general a lo particular distinguiremos las fases una metodología de diseño, de una metodología con una herramienta CASE y de una metodología basada en entidad vínculo con una herramienta CASE.
Continuamos con un capítulo dedicado al sistema de ventanas cliente/servidor Xwindows, que sirvió como plataforma de trabajo.
Los siguientes tres capítulos se enfocan conservando el orden, a las fases de diseño conceptual, lógico y físico de diseño de base de datos. Cada uno de ellos se inicia con la explicación de los conceptos y tareas de la fase, continuan con la descripción de las pantallas y procedimientos utilizados en la herramienta CASE y concluyen con las estructuras de datos y algoritmos empleados en la construcción de los módulos de la herramienta.
Adicionamos un capítulo para el desarrollo de un caso de estudio, donde el problema de estudio es de administración de un almacén general . También se añade un capítulo para las conclusiones.

CAPITULO 1

METODOLOGIA DE DISEÑO DE BASES DE DATOS

Comenzaremos este primer capítulo con una descripción general de las metodologías de diseño de bases de datos basadas en tres fases de diseño: conceptual, lógico y fisico. Comprenderemos como aplicar esta metodología con el modelo entidad vínculo extendido y sabremos la importancia que tiene guiarla con una herramienta automática de ingeniería de software.
La herramienta automática propuesta en el proyecto, evex (entidad vínculo extendido con xwindows) agrupa sus funciones en módulos de edición y transformación para el diseño conceptual y lógico de base de datos. Para el diseño fisico y la implementación de la base de datos, se construye un pequeño sistema manejador de base de datos, CdataX, que compila y transforma en estructuras internas a las especificaciones del lenguaje de definición de datos del diseño lógico.

En la última década han proliferado las implementaciones de los sistemas manejadores de base de datos en todo tipo de computadoras, desde supercomputadoras y minis hasta personales, cada cual con su diferente grado de complejidad.
La dificultad que tienen los usuarios para obtener bases de datos confiables y útiles han dirigido las investigaciones en base de datos hacia el estudio de metodologías para el diseño.
Una metodología es el conjunto de métodos, técnicas y herramientas utilizadas para conseguir un modelo interno de un sistema manejador de base de datos partiendo de los requerimientos del usuario. Aún cuando los modelos en los que se apoyan las metodologías pueden tener distinto grado de abstracción, para todas las metodologías se han definido tres fases imprescindibles.
Una fase de diseño conceptual en donde se representa el contenido e interrelación de los datos del problema a un nivel alto de abstracción, pero fácil de entender e interpretar.
Una fase de diseño lógico en donde se deriva del esquema conceptual, un esquema del modelo de datos en red, jerárquico o relacional.
Y una fase de diseño fisico que tiene por objetivo adaptar el esquema lógico a estructuras internas de un SMBD particular.

1.1 Metodología con el Modelo Entidad Vínculo Extendido (EVE).

El modelo más difundido actualmente es el modelo entidad vínculo extendido (eve), refinamiento del modelo entidad vínculo (ev) de Peter Chen [Chen76], al cual se le añadió la capacidad para manejar relaciones jerárquicas y de grado mayor a dos.
A continuación presentamos las funciones efectuadas en cada una de las fases con el modelo eve, con la agregación de una fase de implementación y carga de datos después del diseño fisico de la base de datos.

Diseño Conceptual.
Previo al diseño conceptual de base de datos existe una fase en donde se analizan los requerimientos del problema, en la cual los analistas definen bajo un universo de discurso el objetivo y alcance del problema, documentándose en una especificación de requerimientos.
El diseño conceptual utiliza esta especificación para reconocer en ella, los elementos y relaciones generales del problema, las cuales se representan mediante diagramas y texto.
El refinamiento de los diagramas generales debe resultar en un diagrama de entidad vínculo completo asociado a un diccionario de atributos.

Diseño Lógico.
Continuando con la fase siguiente, el modelo conceptual de eve, se traduce a uno de los modelos lógicos de bases de datos, esto es al modelo de redes, al modelo jerárquico o al modelo relacional.
Cuando se desea una transformación hacia el último modelo, la transformación debe cumplir las reglas para producir esquemas en tercera forma normal [Teorey86].
Puede resultar, sin embargo que las relaciones obtenidas de la transformación no se encuentran en 3FN, por una mala interpretación semántica del modelo, lo cual a su vez se refleja en un diagrama redundante o incompleto.
Para tales casos, se recomienda aplicar a los esquemas de la transformación algún algoritmo de normalización, dando como entrada el conjunto de todos los atributos del problema y el conjunto de dependencias funcionales para cada relación.

Diseño Físico de la BD.
Los esquemas normalizados y los diccionarios de atributos se mapean en la fase de diseño físico a estructuras lógicas de un SMBD mediante un lenguaje de definición de datos en donde se especifican las tablas, atributos sinónimos, llaves primarias, secundarias y ajenas.

Implementación y carga de datos.
En esta fase, el diseñador compila la definición de la base de datos con el SMBD particular. La compilación crea las estructuras físicas propias del problema y la instancia de la base de datos.
El diseñador entonces procede a cargar los datos para probar y evaluar el desempeño del modelo, para lo cual generalmente se auxilia de aplicaciones que facilitan la manipulación de datos en operaciones de alta, baja, consulta o búsqueda. Dependiendo del resultado de la prueba pondrá en operación la base de datos o regresa a la fase de diseño físico.

1.2 Herramientas de diseño de base de datos

La metodología propuesta con eve resulta útil si se lleva a la práctica de la mano de una herramienta de ingeniería de software asistida por computador (CASE).

Las herramientas que utilizan como soporte al modelo entidad vínculo extendido, contienen módulos especializados en cada fase de diseño, como lo indica el siguiente cuadro:

diseño conceptual	editor de diccionario de atributos editor de diagramas eve analizador sintáctico de diagramas
diseño lógico	traductor de diagramas a esquemas relacionales normalizador de esquemas relacionales
diseño fisico	constructor del lenguaje de definición de datos

A continuación detallaremos las tareas de cada fase con el fin de identificar aquellas que realizan las herramientas automáticas e identificar las funciones de la herramienta automática propuesta en el proyecto.

1.2.1 Módulos de una herramienta automática de diseño

El editor del diccionario de atributos, es generalmente un editor de textos o un editor con ayudas visuales, en el cual se declara nombre, tipo de datos, extensión, descripción y restricciones de valor para los atributos referidos a un problema.

El editor de diagramas, es un editor que permite la representación gráfica del modelado del problema y se utiliza como medio de comunicación entre usuario(s) y diseñador(es).
En el editor se crean, modifican, resguardan y recuperan los diagramas entidad interrelación. En un área de dibujo se disponen los iconos seleccionados de un diccionario y se unen aquellos que se interrelacionan. Cada entidad e interrelación contienen atributos identificadores o simples, los cuales se asocian a los elementos de un diccionario de atributos.

El analizador de diagramas es el componente que examina los diagramas eve para ver si encuentra en ellos inconsistencia. Ejemplos de estas son la falta de nombre en las entidades e interrelaciones, la existencia de entidades aisladas y la carencia de uniformidad en los atributos para una entidad que se repite en un diagrama.

El traductor de diagramas eve es el componente que transforma un diagrama entidad interrelación extendido en esquemas lógicos para bases de datos relacional aplicando reglas de transformación generalizadas pero informales. Inicialmente cada entidad se tranforma en un esquema relacional, las siguientes reglas van agregando nuevos atributos a los esquemas relacionales originales o van agregando nuevos esquemas, dependiendo del tipo, grado y cardinalidad de la interrelación en que participe la entidad.

El normalizador es la herramienta que asegura la depuración de los esquemas relacionales hasta la 3FN, haciendo uso de algoritmos de normalización por análisis o síntesis, siendo estos últimos los generalmente implementados.

El generador del lenguaje de definición de datos, es el componente que une los esquemas normalizados y el diccionario de atributos asociado en una gramática de especificación de datos entendible por un sistema manejador de base de datos particular, el cual finalmente traducirá la definición en estructuras fisicas de la base de datos.

1.2.2 Módulos de evex

Los módulos que identificamos en las herramientas automáticas de diseño de bases de datos, en evex, las hemos agrupado por el tipo de datos que administran y operan.
De tal forma el editor del diccionario de atributos, que maneja datos no visuales, se consideró como un módulo único.

El editor, el analizador de sintaxis y el traductor de diagramas se definieron como el programa diagramador de evex, puesto que todos ellos efectúan operaciones con los diagramas eve. El primero realiza tareas de edición, recuperación y resguardo de diagramas. El segundo de análisis de consistencia. El tercero, de derivación de esquemas relaciones a partir de los diagramas.

Las funciones de normalización las ejecuta el normalizador de esquemas relacionales, reconocido como un módulo único, dada la mayor complejidad de sus algoritmos.

Las función de generación del lenguaje de definición de datos, por ser propia del diseño fisico, aparte de la normalización, que es una función del diseño lógico.

Los módulos y las tareas de evex se pueden apreciar con el siguiente cuadro:

tarea	programa individual	programa integrado
editor del diccionario de atributos	diccionario	evex
editor de diagramas eve	diagramador	evex
analizador de diagramas	diagramador	evex
traductor de diagramas	diagramador	evex
normalizador de esquemas relacionales	normalizador	evex
generador del lenguaje de definición de datos	constructor_ldd	evex

Las interacciones entre los módulos, se distinguen en la figura siguiente.

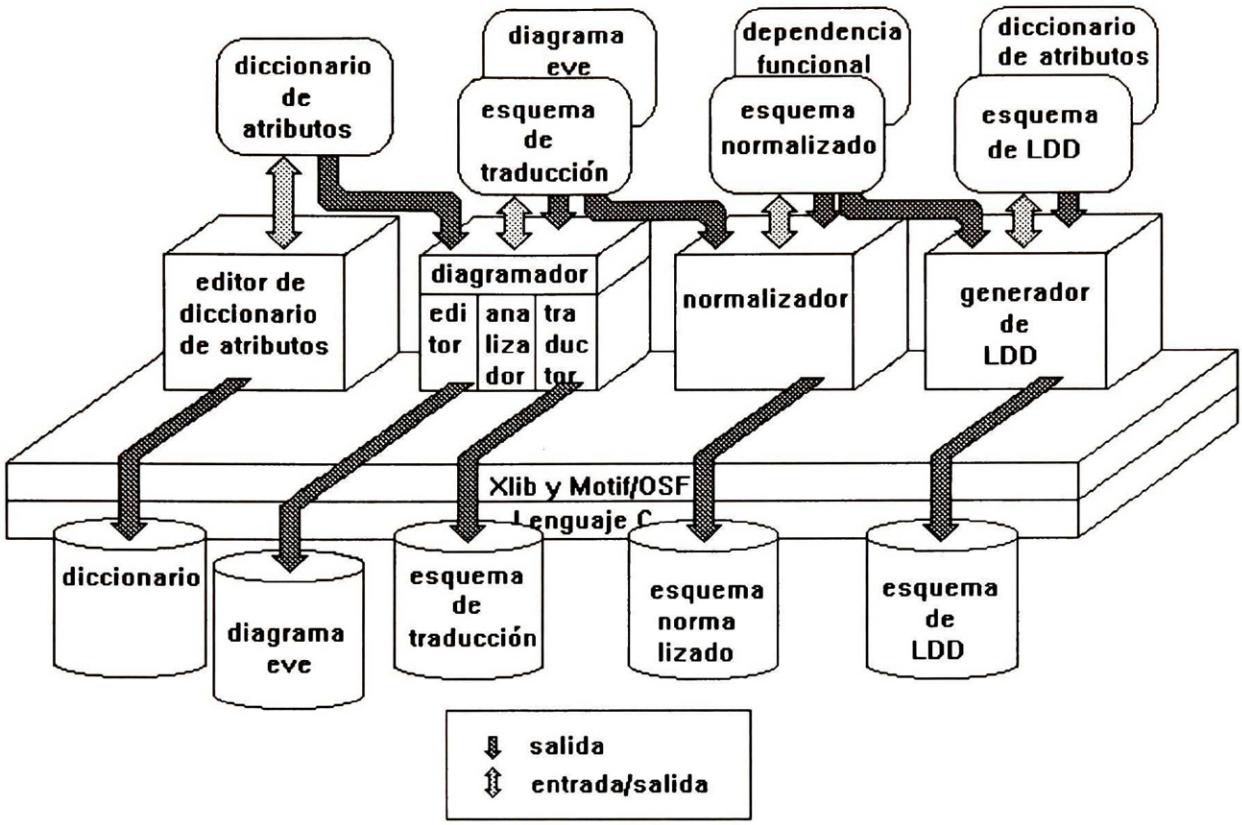


Figura 1.1 El Proyecto EVEX(diseño conceptual y lógico)

El editor del diccionario de atributos realiza sus funciones en memoria y graba la definición de los atributos de un universo de discurso en un archivo con extensión dic.

El diagramador evex toma de memoria el diccionario de atributos del primer módulo, edita los diagramas entidad vínculo extendido, enlaza los atributos al diagrama, analiza la sintaxis del diagrama y lo convierte en esquemas relacionales. En este módulo se graban los diagramas eve asociados con los atributos en archivos con extensión eve y los esquemas de traducción en archivos con extensión rel.

El normalizador lee los esquemas de traducción y nuevas dependencias funcionales para en memoria obtener el esquema normalizado en 3FN, el cual puede grabarse en un archivo con extensión nor.

El generador del lenguaje de definición de datos recibe como entrada un esquema normalizado y con su diccionario de atributos asociado y produce en memoria el esquema del ldd, el cual se graba en un archivo con extensión ldd.

El proceso de generación del lenguaje de definición de datos puede ser el último proceso de las herramientas automáticas de apoyo al diseño de base de datos.

Sin embargo una herramienta completa, debe verificar el diseño de la BD con su implementación, sea en un SMBD embebido en la herramienta o mediante la compilación del lenguaje de definición de datos.
El proyecto evex sigue la segunda opción, en donde se implanta la base de datos en un SMBD. El ldd producido por el constructor de evex, se compila para el SMBD CdataX.
La operación del modelo en el SMBD se verifica mediante una interfaz de programas para consulta y modificación de relaciones. A continuación presentamos una tabla de los programas que sirven como interfaz a la compilación del ldd y para operación básica de las relaciones.

tarea	programa individual	SMBD
visualizador y analizador del ldd	compila_ldd	
constructor de la instancia de la base de datos	compila_ldd	CdataX
aplicador de operaciones básicas con relaciones	captura_<nom_bd>	CdataX

Se presenta a continuación un esquema de la interfaz de evex con el SMBD CdataX y con los programas compila_ldd y captura_<nom_bd>

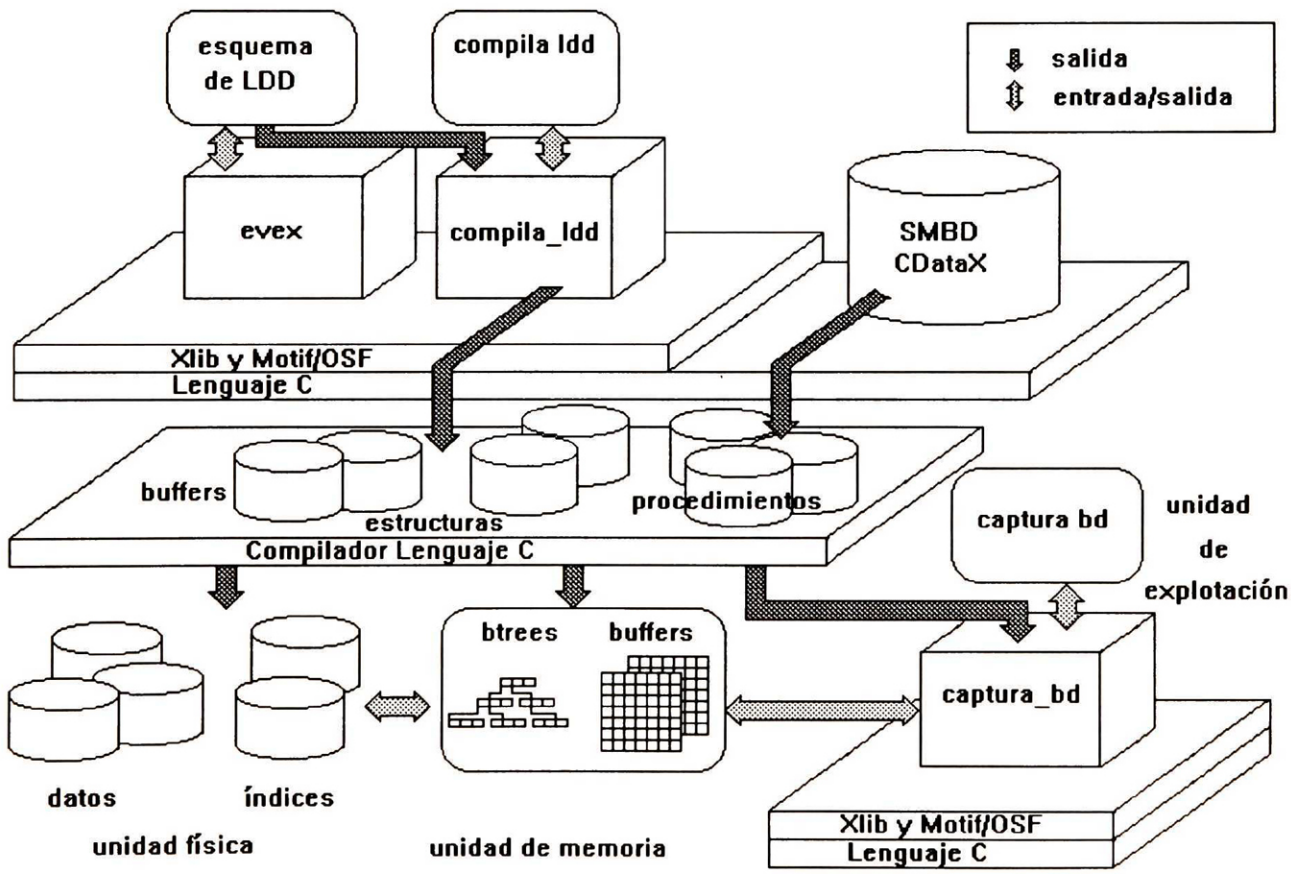


Figura 1.2 El Proyecto EVEX(diseño fisico e implementación)

En la figura 1.2 la compilación del esquema del lenguaje -segundo módulo superior- es el primer proceso del diseño fisico e implementación de evex.
El esquema del ldd se lee, visualiza por el compilador del ldd. El módulo en la compilación crea los archivos de administración de memoria y estructura de la base de datos particular. Estos archivos junto con los procedimientos para manipulación del SMBD CdataX se enlazan y compilan como un programa en C, y generan la instancia de la base de datos, consistente de tres unidades: fisica, de memoria, explotación de datos. La unidad fisica es aquella en donde se mantiene la estructura de las relaciones que será persistente al guardarse en archivos de índices y de datos. La unidad de memoria es aquella en donde se administran los buffers y estructuras de datos volátiles como son los btrees. Por último la unidad de explotación de la base de datos -extremo inferior de la figura - se encarga de las funciones de consulta y actualización de datos, para lo cual interactúa con las unidades fisicas y de memoria del SMBD.

Evex y el SMBD CdataX se ejecutan en plataformas que soportan las librerías de Xlib, los Xtoolkits y Motif de OSF (Open Software Foundation).
X está llamado a ser el protocolo estándar en sistemas gráficos de red para estaciones de trabajo como DEC, HP, Sun e IBM.
Originalmente el proyecto se desarrolló para estaciones de trabajo DEC 5000/240, 5000/250 y 5000/300 sobre Xwindows para Ultrix 4.2. Posteriormente la herramienta de diseño conceptual evex se implementó para estaciones de trabajo Sun Sparc 20.
Seleccionamos los intrínsecos de Xwindow y Motif como herramienta para el desarrollo de la interfaz porque los productos resultaban fáciles de usar , agradables a la vista y consistentes en toda la aplicación.

El modelo entidad vínculo extendido utiliza una metodología de diseño dividida en fases conceptual, lógica y fisica.
La expresividad y sencillez del modelo tanto en su representación conceptual como en su derivación a esquemas relacionales tienen elementos suficientes para concebir la construcción de una herramienta CASE de apoyo al diseño conceptual, lógico y fisico de la metodología con eve.
Evex es el nombre de la herramienta propuesta para el diseño conceptual y lógico de base de datos.
El sistema manejador CdataX es la herramienta empleada para el diseño e implementación de la instancia de la base de datos.
Evex auxilia en la elaboración del diccionario de atributos, el diagrama eve, el análisis sintáctico del diagrama, la transformación de los diagramas a esquemas relacionales y estos a su vez a esquemas normalizados y la producción del lenguaje de definición de datos.
El SMBD CdataX compila el lenguaje de definición e instrumenta la instancia de la base de datos.
La plataforma de construcción de ambas herramientas bajo las librerías de Xlib, los Xtoolkits y Motif permiten una interfaz amigable y consistente a través del ciclo de vida del diseño.

CAPITULO 2

EL SISTEMA XWINDOW

La herramienta EVEX fué pensada para su ejecución en plataformas unix, dado que esta plataforma contaba con pocas herramientas automáticas de apoyo al diseño de bases de datos.
Para su construcción, requeríamos de un sistema gráfico que proporcionase el servicio de despliegue y que permitiese la construcción de programas para el despliegue de de los algoritmos de visualización y de las interfaces de usuario final.
Ese sistema lo encontramos en Xwindow, bajo cuya denominación agrupamos al protocolo X para redes tcp/ip, a las primitivas de graficación de bajo nivel Xlib y a las librerías de alto nivel Xtoolkits y OSF/Motif.
En el presente capítulo describimos la filosofía cliente/servidor con que opera Xwindow y sus diferentes configuraciones.
Examinamos brevemente cada una de las librerías, delimitando su entorno, nivel de abstracción y alcance, detallando para aquellas que son de alto nivel su jerarquía de clases.
Finalmente definimos la sintaxis del pseudocódigo utilizado a partir del siguiente capítulo en las descripciones de los algoritmos de interfaz.

2.1 La arquitectura cliente/servidor en X.

En una arquitectura cliente/servidor pura, se tiene una máquina servidora atendiendo las requisiciones de servicio de las computadoras clientes, conectadas mediante una red.
En Xwindow la arquitectura se lleva a cabo sobre una red tcp/ip o decnet utilizando el protocolo de comunicación X. Pero a diferencia de una arquitectura cliente/servidor pura, en donde el cliente y el servidor son ambas computadoras, con Xwindow, la arquitectura se amplía a nivel de programas, en donde el programa servidor se ejecuta en un equipo con despliegue gráfico, teclado y ratón, y el cliente en alguna computadora conectada física y lógicamente con el servidor.
Al extenderse la arquitectura a nivel de programas se dan los siguientes casos de operación:
Una estación de trabajo ejecutando al cliente y servidor. e.g. un despliegue de la calculadora de Xwindow.
Una estación de trabajo ejecutando un servidor y varios cliente. e.g. un despliegue de la calculadora y el reloj de Xwindow.
Varias estaciones de trabajo ejecutando su respectivo programa servidor y un cliente en una de ellas pero haciendo peticiones a todas. eg. una aplicación desplegándose con fines de capacitación en todas las estaciones de trabajo de una red.

El programa servidor X, sólo se ocupa de controlar los eventos entre clientes, el despliegue gráfico, el ratón y el teclado. El cliente envía las peticiones de servicio de alguno de los componentes del despliegue gráfico al servidor, este responde a las solicitudes del cliente, quien cuando recibe la respuesta, efectúa las acciones correspondientes a la aplicación.
La comunicación entre servidor y clientes la lleva a cabo el protocolo X, quien permite la comunicación simultánea en ambos sentidos -cliente a servidor, servidor a cliente-.

El desarrollo de una aplicación con Xwindow tiene la ventaja de que si nuestra red consta de equipos con despliegues visuales heterogéneos como son dec, sun, hp, ibm, todos ellos ejecutando su servidor X, la aplicación puede visualizarse en cualesquiera de ellos con solo especificar la dirección ip del destino.
Los especialistas de Xwindows claman que las aplicaciones de X son fáciles de transportar entre equipos de la osf (dec, sun, hp, ibm, etc.). Sin modificaciones substanciales al código una aplicación se enlaza con las librerías que invoca de Xwindows, se compila en el equipo deseado y se genera el ejecutable.

2.2 Eventos de X.

Un programa cliente X se compone en su forma más básica de tres partes: las rutinas de inicialización o apertura del control de despliegue gráfico, un control del ciclo de eventos y las rutinas de finalización o cierre del despliegue de la computadora.
Es en el ciclo de eventos del cliente en donde se identifican las peticiones que el servidor X debe atender. Dichas peticiones se clasifican según su procedencia en eventos de teclado, del ratón, de la pantalla y de comunicación entre clientes.
Los del ratón pueden ser: de presión (clic) y liberación de los botones.
Los del teclado pueden ser de: presión y liberación de alguna tecla o juego de teclas.
Los de la pantalla se subdividen en selección, activación y exposición de ventanas, de modificación del estado de la ventana, de modificación de la tabla de colores, de control de estado.
Los eventos de comunicación entre clientes son aquellos que permiten el envío de información entre aplicaciones de X. Se emplean cuando una aplicación quiere comunicarse con otra instancia del mismo programa o con otra aplicación de Xwindows.

2.3 Xlib

Para programar con Xwindow, el sistema proporciona un conjunto de librerías de bajo nivel, escritas en lenguaje c, denominado Xlib, estas librerías forman la interfaz procedural con el protocolo X.
Las funciones ofrecidas por Xlib pueden ser para despliegue gráfico, de ventanas, de construcción de gráficas y de recursos gráficos.

Las de despliegue permiten la apertura, cierre y obtención de datos generales del despliegue.
Las de ventanas permiten la creación, destrucción, mapeo, cambio de configuración, cambio del orden de activación y traducción de coordenadas de ventanas.
Las de construcción de gráficas borran y copian áreas de dibujo; trazan puntos, líneas, rectángulos y arcos; rellenan áreas; dibujan texto, manejan el cursor y los tipos de letras; transfieren imágenes.
Las de los recursos gráficos crean, copian y destruyen los mapas de colores; asignan, cambian y liberan las celdas de colores; crean los mapas de pixeles y crean, copian y modifican el contexto gráfico.
En donde se conoce como mapa de pixeles a un conjunto de puntos posibles de desplegarse en un área de dibujo, como celdas de colores se conoce a aquellas celdas de memoria conteniendo información de los colores utilizados en los despliegues y como contexto gráfico a un conjunto de preferencias de dibujo formado por los tipos o fuentes de caracteres, al tipo y aspecto de la línea de dibujo, el fondo y frente del área de dibujo, etc.
Si bien es posible construir aplicaciones empleando únicamente Xlib, su presentación no es ergonómica, ya que los límites de las ventanas, botones y menúes tienen apariencia plana.
Una interfaz visual de usuario construida con Xlib difícilmente podrá particularizar su comportamiento al gusto del usuario en cada ejecución. Para generalizar el comportamiento de una interfaz con Xlib, debemos utilizar el mismo conjunto de funciones, plenamente verificadas en cada aplicación.
Construir una interfaz de este tipo con Xlib, no es muy recomendable, máxime si existen las utilerías probadas que realizan esta función: los Xtoolkits.
En la construcción de aplicaciones con Xwindow los especialistas recomiendan implementar la interfaz visual de usuario con Xtoolkits u otro producto semejante y Xlib para las primitivas de dibujo.

2.4 Xtoolkits.

Las librerías de Xtoolkits están construidas sobre las de Xlib, con el fin de controlar eficientemente las interfaces visuales para usuario final en Xwindow.
Un código de interfaz elaborado con los Xtoolkits reduce considerablemente el número de líneas de la misma elaborada con Xlib. Como ejemplo un programa de despliegue de un letrero en una ventana, se codificaría en Xlib en un programa de 50 a 100 líneas, mientras con los Xtoolkits se limitaría a una decena de líneas.
Sin embargo, no todo es bondad, el tamaño del ejecutable de los Xtoolkits aumenta, pues incorpora todas las rutinas de los objetos de la interfaz utilizados .

Los Xtoolkits se desarrollaron empleando programación basada en objetos, y el programador ya sea que desee crear sus propios objetos o utilizar los existentes, debe continuar con la filosofía basada en objetos.

El conjunto de funciones de Xtoolkits soportan una jerarquía básica de clases de objetos gráficos de interfaz, denominados widgets en inglés.
Para cada clase básica de la interfaz, se describen sus recursos y su comportamiento. En terminología orientada a objetos equivale a la descripción de sus variables y funciones miembro.
Las clases de la interfaz heredan propiedades de sus superclases. La clase con jerarquía mayor o superclase base es el "Core". La Figura. 2.1 representa la jerarquía de clases de los Xtoolkits.

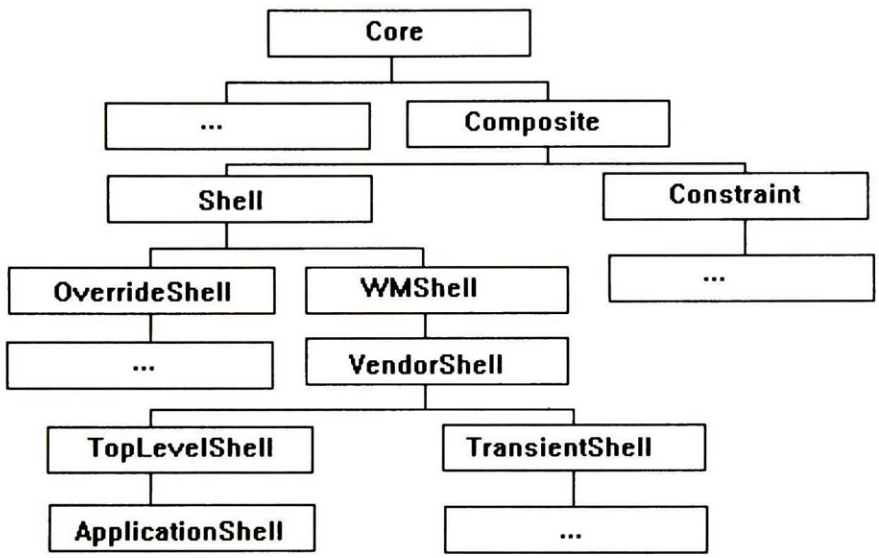


Figura 2.1 Jerarquía de clases de los Xtoolkits.

En una aplicación los objetos de la interfaz son instancias de las clases.
Un objeto tiene ciertas abstracciones de comportamiento (de fuentes, cursores, mapas de colores, etc.) llamadas recursos. Los recursos pueden establecerse completamente en el código de la aplicación, o establecerse para los objetos básicos en el código, y personalizarse a través de los archivos de recursos. La aplicación a tiempo de ejecución lee los archivos de recursos y cambia su apariencia.

Los Xtoolkits proporcionan rutinas básicas de manejo de los objetos de la interfaz, Intrínsecos de Xt: de creación, manipulación y destrucción; junto con rutinas de propósito general (por ejemplo XtCalloc) y rutinas relacionadas con X.

Las aplicaciones con X controlan la lógica de ejecución mediante una cola de eventos por cliente X. Cada cliente, monitorea la cola de eventos, en el momento que detecta un evento manda ejecutar una función. Al finalizar la ejecución el ciclo de la cola de eventos continua, para observar el siguiente evento y atenderlo.
Con los Intrínsecos de Xt, en los objetos de la interfaz se encapsulan las funciones, en llamadas a función por evento, callbacks en inglés. Cada objeto de interfaz tiene asociado un número de eventos que se pueden activar por la llamada a función por evento.
El manejo del ciclo de eventos con los Intrínsecos se encapsula en las funciones XtMainLoop y en XtAppMainLoop.

Los Xtoolkits permiten la construcción de nuevas clases a partir de alguna de la clases pertenecientes a su jerarquía. El programador debe seleccionar la clase que se asemeje más a la clase que planea construir, y especializarla. A los programadores que derivan nuevas clases de la jerarquía presentada con los Xtoolkits, se les denomina programadores de clases de interfaz o programadores de widgets.

De los conjuntos de clases de objetos construidos sobre los Xtoolkits, Athena se suministra sin costo al adquirir los Xtoolkits, y Open Look Intrinsic Toolkits (OLIT) y Motif se adquieren por separado.

2.5 Motif

De las interfaces gráficas de usuario: Athena, OLIT y Motif, la tercera es la más popular. Construye clases de objetos gráficos, sobre los proporcionado por los Xtoolkits.
Proporciona clases de objeto primitivos: etiquetas, botones de dibujo, botones en cascada, botones con opresión, botones con flecha, botones de verificación, listas, barras de deslizamiento, separadores.
Clases de apariencia heredada o contenedores: clase de objeto forma, de objeto armazón, de apariencia filacolumna, de pizarra, de ventana principal.
Clases de objetos compuestos: de selección de archivos, de ejecución de comandos, de ventanas de mensajes o de diálogo, de área de dibujo, de ventana de texto, de ventana de texto con barra deslizante.
Los botones en cascada, las etiquetas y los separadores son los elementos para la construcción de menúes en cascada (pulldowns) y menúes ocultos (pop-up).

La Figura 2.2 a) b) y c) representa la jerarquía básica de clases de Motif.

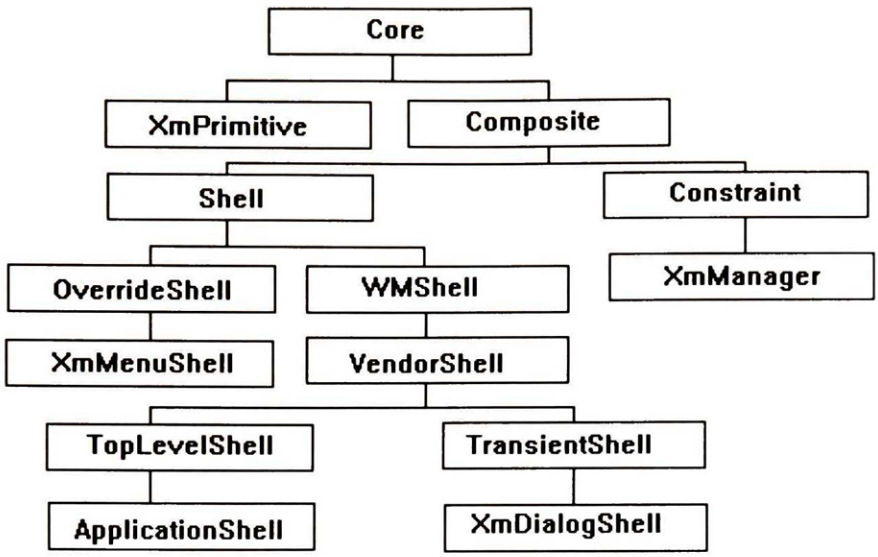


Figura 2.2 a) Jerarquía de clases a partir de la clase Core.

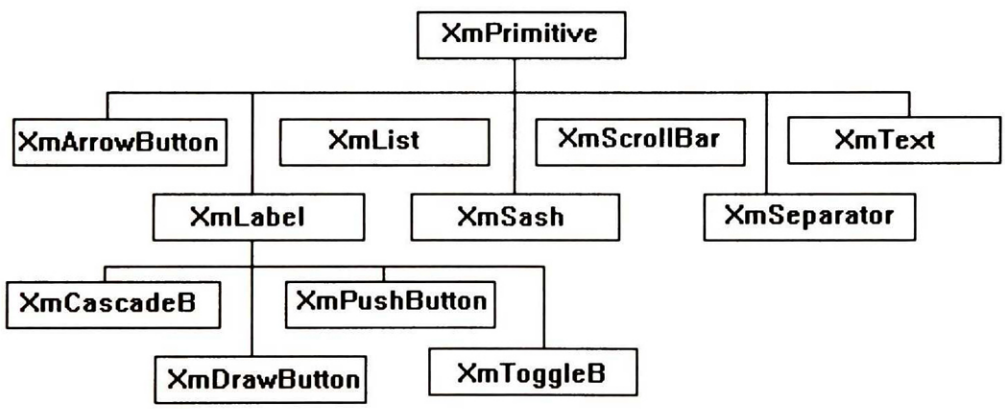


Figura 2.2 b) Jerarquía de clases a partir de la clase XmPrimitive

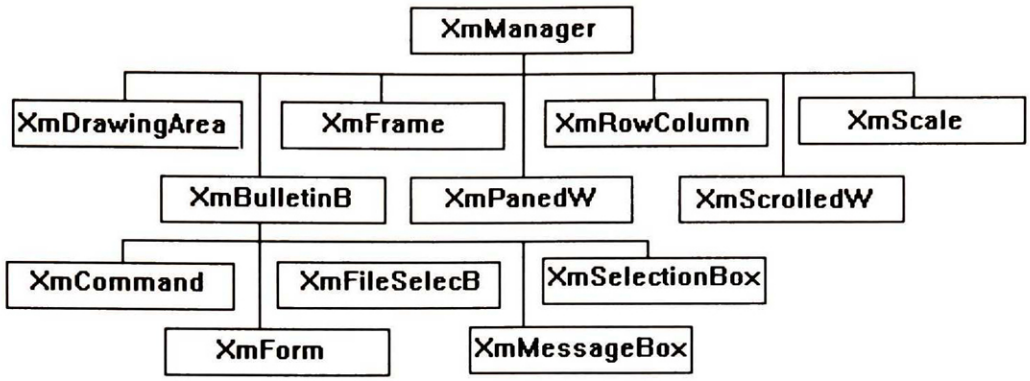


Figura 2.2. c) Jerarquía de clases a partir de la clase XmManager

Motif también añade a los tipos de datos de los Xtoolkits, un nuevo tipo: el XmString, este nuevo tipo, permite mezclar tipografías o fuentes diferentes en una cadena de caracteres.

La apariencia y comportamiento de una interfaz visual desarrollada en Motif, mejora la calidad de una desarrollada con los Xtoolkits, y esta a su vez a una desarrollada con Xlib.
Un botón con los Xtoolkits visualmente se representa por un doble rectángulo. Con las librerías un botón con el manejo del color tiene apariencia tridimensional, la cual se aprovecha para representar la opresión o liberación del botón al efectuar un clic de ratón sobre el objeto.
Motif ofrece un conjunto de recomendaciones y reglas a los programadores de interfaces visuales, que lo guían hacia la obtención de interfaces visuales consistentes.

2.6 Los archivos de recursos para los Xtoolkits y Motif.

Con los Xtoolkits y Motif, las aplicaciones pueden cambiar su configuración y comportamiento a tiempo de ejecución. Las aplicaciones con los Xtoolkits o Motif leen los archivos de configuración, denominados archivos de recursos, de directorios especificados por la variable de ambiente XAPPLRESDIR.
En un archivo de recursos, se pueden definir las propiedades de cada objeto de interfaz de la aplicación, dichas propiedades pueden aplicarse a un objeto o a un conjunto de objetos, e.g. definimos una propiedad para un objeto contenedor, y puede ser tan simple, como el tamaño o el color de un objeto, o más complejo, como el contenido de un objeto contenedor.
Generalmente, en los programas de aplicación se definen los recursos necesarios para su ejecución y en un archivo de recursos, los recursos que el usuario puede personalizar.

2.7 Sintaxis para el pseudocódigo de interfaces con Motif.

Para describir un programa con Motif emplearemos la siguiente sintaxis:

Crea Forma <Tipo de forma> <Nombre>
Crea Menú <Tipo de Menú> <Posición en Forma>
Crea Opción <Nombre>
Crea Letrero <Nombre>
Crea Texto <Tipo de Texto> <Nombre>
Crea Lista <Tipo de Lista> <Nombre>
Crea Botón <Etiqueta>
Crea Area de Dibujo
Crea Barra deslizante <Orientación>

Llamado a una función por evento. <Nombre función>
Llamado al Manejador de eventos. <Evento> <Invocador de Evento>
Activa Ciclo de Eventos Principal.

La elección del sistema Xwindow como plataforma para el desarrollo de evex se basa en las diferentes configuraciones cliente/servidor que ofrece, pero sobre todo en el grado de estandarización alcanzado por el sistema gráfico en red.
En la implementación de la herramienta CASE empleamos las librerías Xlib para la construcción de las primitivas de dibujo, de copiado y visualización de iconos y de trazado de líneas.
Los Xtoolkits y Motif son las librerías con una implementación fundamentada en clases y objetos que utilizamos en la construcción de las interfaces de usuario final.
Concluimos el capítulo con la sintaxis empleada para describir las interfaces de usuario de los diversos módulos de evex.

DISEÑO CONCEPTUAL CON EVE

El objetivo del diseño conceptual es capturar los requerimientos del problema en un modelo expresivo y simple, de tal manera que guarde el mismo significado para diseñadores y usuarios.
Un modelo que cubre con esas propiedades es el de entidad vínculo extendido.
El presente capítulo lo enfocamos al desarrollo del diseño conceptual de base de datos con el modelo entidad vínculo extendido apoyado en la herramienta automática EVE-Xwindows para las tareas de edición del diccionario de atributos y de los diagramas.
El modelo originalmente propuesto por Chen obtiene mayor expresividad semántica con la incorporación de interrelaciones jerárquicas y de grado mayores a dos. Las extensiones al modelo fueron representadas de manera diferente por autores como Teorey, Chen, Everest, con lo cual se agruparon los modelos en familias, la herramienta automática del trabajo, se basa en la familia de Teorey.
La explotación de la herramienta evex se enriquece cuando el usuario conoce los conceptos fundamentales del modelo original, las extensiones sustanciales y la metodología para diagramar con eve, todos ellos temas tratados en el capítulo.
Para quién desee introducirse más en la herramienta evex, en el capítulo se describe la implementación del editor de atributos y de diagramas.

3.1 El modelo original Entidad Vínculo.

El modelo entidad vínculo fué propuesto para modelar conceptualmente BD basándose en el enfoque más natural del mundo real que consiste en entidades e interrelaciones (Chen 1976).
EV se define sintácticamente mediante iconos con significado semántico al conectarse.
La semántica de un diagrama eve se transfiere a un modelo lógico de BD siguiendo un conjunto de reglas de transformación. Para iniciar el diseño conceptual con ev necesitamos conocer las definiciones de los elementos básicos: entidad, atributo, vínculo o interrelación.

3.1.1 Entidades

Las entidades denotan a las personas, lugares, conceptos, actividades u objetos de interés del sistema.
A las ocurrencias particulares de una entidad, se le denomina instancia de entidad.
Usualmente a partir del agrupamiento de objetos individuales llegamos a establecer el conjunto o entidad.
De una entidad **Empleado**, Juan Ramírez Luna es una instancia. De una entidad **Departamento**, el departamento de Ventas es una instancia.

3.1.2 Atributos

En término de entidades un atributo puede definirse como una propiedad o característica de las entidades.
Cuando el atributo describe una característica atómica se le denomina atributo simple o sin agregados. Un atributo compuesto describe más de una característica de la entidad. Cuando hablemos de atributo, a partir de este momento, nos estaremos refiriendo al atributo sin agregados o atributos que toman un solo valor.

A una ocurrencia particular de un atributo se le denomina valor de atributo. Los valores de atributos pertenecen a dominios específicos de datos. Una entidad **Empleado**, tiene como atributos un Nombre_Empleado, Fecha_Ingreso, Edad, Sexo. Los valores de atributo de la instancia Juan Ramírez Luna son "Juan Ramírez Luna", 04/01/80, 34, "M". Los valores de atributos de una instancia pertenecen a un dominio de datos, por ej. para la instancia Juan Ramírez sus valores son del dominio de arreglo de caracteres, fecha, número entero y caracter.

Al atributo o conjunto de atributos que distinguen de manera única una instancia de entidad y se le denomina identificador.
Cuando el identificador se compone de más de un atributo es posible que en nuestro modelado, estemos omitiendo alguna interrelación con la entidad del identificador. Si esto no es así, y si el identificador está compuesto de tres o más atributos se suele crear un atributo artificial numérico, simple y corto.
En la entidad **Empleado,** se crea el identificador artificial Número_Empleado, con el fin de evitar el identificador compuesto Nombre_Empleado, Fecha_Ingreso.

A los atributos que no forman parte del identificador, y por tanto no distinguen de manera única una instancia de entidad, se les denomina descriptores.
Del empleado Juan Ramírez, los atributos descriptores originales eran Edad y Sexo, con la introducción del identificador artificial Número_Empleado, el Nombre_Empleado y la Fecha_Ingreso se agregan como atributos descriptores.

En un diagrama EV, cuando un atributo descriptor toma múltiples valores para un valor del atributo identificador, conviene convertir el atributo de múltiples valores en una nueva entidad, asociando esa nueva entidad con una interrelación a la entidad original.
Con esto estaremos preservando en el modelado la normalización en primera forma normal, como veremos más adelante.

3.1.3 Vínculos o Interrelaciones

Las interrelaciones son las declaraciones de las interacciones o vínculos que se tienen entre las entidades.
Por ej. **Asignado_a**, entre **Empleado** y **Proyecto**.
Una instancia de una interrelación es una declaración de los vínculos entre instancias de entidad.

Por ej. La instancia de interrelación Asignado_a vincula las instancias Juan Ramírez Luna y el proyecto de Capacitación en Redes Locales.
Una mejor forma de declarar a las interrelaciones, consiste en enunciar los papeles o roles de las entidades.

La interrelación Asignado_a entre Empleado y Proyecto. Se enuncia como Un **Empleado** es **Asignado_a** un **Proyecto**.
O enunciando las instancias de roles como:
Juan Ramírez Luna, es asignado al proyecto de Capacitación en Redes Locales.

Las interrelaciones pueden contener atributos descriptivos inherentes. Estos describen características de la interrelación y no de alguna de las entidades participantes de la interrelación.
La interrelación **Asignado_a** entre las entidades **Empleado** y **Proyecto,** pudiera tener como atributo descriptivo Fecha_Asignación y Actividad_Desempeñada. Donde la Fecha_Asignación, es diferente a la Fecha_Ingreso del empleado en la compañía y a la Fecha_Inicio del proyecto, y la Actividad_Desempeñada refiere a la labor del empleado en el proyecto.
La instancia del rol de Juan Ramírez la completaríamos como:
Juan Ramírez Luna, con fecha 03/22/94 fue asignado al proyecto de Capacitación en Redes Locales como Expositor.

3.1.4 Grado de los Vínculos.

Al número de entidades participantes en una interrelación, se le conoce como grado de interrelación.
Una interrelación de grado uno es una interrelación recursiva, donde solo participa una entidad.
Una interrelación de grado uno es la de **Supervisor_de** entre **Empleados**. Con instancias: Juan Ramírez Luna supervisa a Raúl Flores Pérez.
Otros ejemplos de interrelaciones unarias son las interrelaciones **Esposo_de** entre **Empleados**, **Padre_de** entre **Empleados**.

Una interrelación de grado dos o binaria, es una interrelación entre dos entidades. Por ej. Una interrelación **Supervisa** entre **Empleado** y **Proyecto**. O mejor descrita con roles como : Un **Empleado supervisa** un **Proyecto**.

El modelo original ev, no incluía interrelaciones de grado mayor al binario. Sin embargo, resultó necesario incluir interrelaciones de grado tres, en las extensiones realizadas a EV

Una interrelación de grado tres o ternaria, es aquella en la cual participan tres entidades. Por ej. una interrelación **Asignado_a** entre **Empleado**, **Proyecto** y **Actividad**. Descrita con el rol Los **Empleados** son **Asignados_a Proyectos**, realizando una **Actividad**.

3.1.5 Conectividad de los Vínculos.

El comportamiento de las instancias de entidades, dentro de una interrelación, se describe con la conectividad de las interrelaciones.
Los valores de conectividad de las instancias es uno (1) o muchos (M).
Las interrelaciones de grado uno y dos tienen conectividad una a una (1:1), una a muchas (1:M) o muchas a muchas (M:N).

Una interrelación de grado uno **Supervisor_de,** tiene conectividad 1:M entre **Empleados**. Cuando se tienen instacias como Juan Ramírez Luna supervisando a más de un empleado: Raúl Flores Pérez, Amalia García Solís, Luis Méndez Ayala.
Una interrelación de grado uno **Esposo_de** tiene conectividad 1:1 entre **Empleados**, cuando se describe una interrelación de los esposos actuales entre empleados.
Un **Empleado** es **Esposo_de** otro **Empleado**. O la instacia del rol : Juan Ramírez Luna es esposo_de Diana Campos.

Una interrelación de grado dos, **Dirige** entre **Empleado** y **Proyecto,** dependiendo de la conectividad, interpretará de manera diferente a la interrelación.

Si Juan Ramírez dirige la Capacitación en Redes Locales.
Luis Méndez dirige la Venta de Accesorios en Redes Locales.
Diana Campos dirige la Venta de Equipo de Cómputo.

La conectividad es uno a uno (1:1) donde Un **Empleado Dirige** un **Proyecto** y el Proyecto es dirigido por un Empleado.

Si Juan Ramírez dirige la Capacitación en Redes Locales y la Venta de Accesorios en Redes Locales.
Diana Campos dirige la Venta de Equipo de Cómputo.
La conectividad es de uno a muchos (1:M) donde Un **Empleado Dirige** más de un **Proyecto** y un Proyecto es dirigido por un Empleado.

Si Juan Ramírez dirige la Capacitación en Redes Locales y la Venta de Accesorios en Redes Locales.
Diana Campos dirige la Venta de Equipo de Cómputo y la Venta de Accesorios en Redes Locales.
La conectividad es de muchos a muchos (N:M). Un **Empleado Dirige** más de un **Proyecto**, y un Proyecto es dirigido por más de un Empleado.

3.2 Extensiones al Modelo Entidad Vínculo.

El modelo original entidad vínculo, fué enriquecido por el mismo Chen y otros autores, entre los que destaca Teorey [Teorey 86].

Las propuestas de extensión surgieron de la problemática habitualmente encontrada en el diseño de bases de datos complejas y de tamaño grande, y consistieron en las interrelaciones de dependencia entre entidades, el aumento en la cardinalidad de las interrelaciones y la conceptualización de las jerarquías de generalización y especialización. En el presente apartado examinamos las últimas dos extensiones.

3.2.1 Interrelaciones de Grado Tres.

Una interrelación de grado tres , tiene conectividad de uno a uno a uno (1:1:1), de uno a muchos a uno (1:M:1), de uno a muchos a muchos (1:M:N) o de muchos a muchos a muchos (M:N:M).
Una interrelación de grado tres, **Tiene** entre **Empleado**, **Vivienda** y **Ciudad** es de conectividad (1:1:1) cuando describe un rol de:
Un **Empleado Tiene** una **Vivienda** en una **Ciudad.**
Un **Empleado** en una **Ciudad Tiene** una **Vivienda**.
Una **Vivienda** en una **Ciudad Tiene** un **Empleado** como dueño.
Las siguientes instancias respetarían los roles.
El empleado Juan Ramírez tiene la vivienda 1200 en Mazatlán. El empleado Juan Ramírez tiene la vivienda 1300 en Guadalajara. En Mazatlán Juan Ramírez es dueño de la vivienda 1200. En Guadalajara Juan Ramírez es dueño de la vivienda 1300. La vivienda 1200 de Mazatlán tiene como único dueño a Juan Ramírez. La vivienda 1300 de Mazatlán tiene como único dueño a Juan Ramírez.

Una interrelación de grado tres **Asignado_a** entre **Empleado**, **Proyecto** y **Actividad** con conectividad 1:M:1 describe una interrelación en donde:
Una **Actividad** es **Asignada_a** un **Empleado** participando en un **Proyecto**.
Un **Proyecto** es **Asignado_a** un **Empleado** efectuando una **Actividad**.
Pero una **Actividad Asignada_a** un **Empleado** puede desempeñarla en más de un **Proyecto**.

El Análisis del Sistema es Asignado a Juan Ramírez en el proyecto de Sistema de Normatividad.
Solo Juan Ramírez está Asignado al proyecto de Sistema de Normatividad para el Análisis del Sistema.
Pero Juan Ramírez es Asignado para Análisis del Sistema en el proyecto de Sistema de Normatividad, y en el proyecto de Precios Unitarios.

Una interrelación de grado tres **Percibe** entre **Empleado**, **Salario** y **Trabajo** con conectividad M:N:1 describe una interrelación en donde
Un **Empleado Percibe** un **Salario** por efectuar un **Trabajo**.
Un **Salario** de un **Trabajo** se **Percibe** por más de un Empleado.
Pero un **Empleado** por un Trabajo Percibe más de un Salario.

Una instancia con estas características sería que Juan Ramirez percibe el salario M-100 y el salario A-102 por el trabajo de Capacitación en Análisis OO.
Más el Salario M-100 de Capacitación en Análisis OO lo percibe Juan Ramírez y Luis Méndez.

Una interrelación de grado tres Utiliza entre **Empleado**, **Actividad** y **Técnica** con conectividad M:N:M describe una interrelación en donde
Más de un **Empleado Utiliza** más de una **Técnica** para una **Actividad**.

Juan Ramírez Utiliza la normalización como Técnica de Diseño de BD relacionales.
Juan Ramírez Utiliza el modelado con EVE como Técnica para Diseño de BD relacionales.
Diana Campos Utiliza la normalización como Técnica de Diseño de BD relacionales.
Juan Ramírez Utiliza el modelado con EVE como Técnica para Diseño de BDOO.

3.2.2 Interrelaciones de Jerarquías: Generalización y Especialización

Los primeros modelados de Bases de Datos, orientados hacia problemas específicos, fueron resueltos con el modelado EV propuesto por Chen. El advenimiento del modelado de la empresa o del sistema integral, dejó al modelo de EV sin elementos para plantear las jerarquías entre entidades.
El modelado de la empresa, propone un modelo de datos global diseñado de diferentes modelos conceptualizados de vistas de los integrantes de la empresa. En el proceso de integración del modelo global, se descubrían interrelaciones jerárquicas entre entidades.
De ahí la extensión al modelo de las jerarquías: de generalización y de especialización.

Una entidad E es una generalización de las entidades E1, E2, ... En, si obtenemos a la entidad E de la unión de los atributos comunes a E1, E2, ..., En.
La generalización E, resalta la semejanza entre las entidades E1, E2, ... En, pues es el resultado de su agrupamiento.
Una instancia de E es la generalización de una y solo una instancia de alguna de las entidades E1, E2, ..., En.

Un **Empleado** es la generalización de los empleados **Confianza** y **Sindicalizado**.
Numero_Empleado, Nombre_Empleado, Rfc, Edad, Sexo
son atributos generalizados de **Empleado**, comunes a **Confianza** y **Sindicalizado**.
A **Empleado** se le agrega el atributo artificial Tipo_Empleado, que nos distinguirá la clasificación.
Un **Empleado** no podrá a la vez ser **Sindicalizado** y de **Confianza**.
El empleado **Sindicalizado** tiene atributo especializado Nivel_Sindicalizado, Cuota_Sindical. Y el empleado de **Confianza** tiene atributo especializado Nivel_Confianza.

En la especialización, partiendo de una entidad E tomamos subconjuntos de atributos para formar con ellos entidades especializadas E1, E2, ... En. Existiendo la posibilidad de que alguna instancia de E no forme algun subconjunto, y de que genere más de un subconjunto.

Una **Cuenta** se especializa en **Cuenta_de_Cheques**, **Cuenta_de_Ahorro** y **Cuenta_de_Crédito**.
La **Cuenta** tiene como atributos generalizados el número de Cuenta, Num_Cuenta, tipo de cuenta, Tipo_Cuenta, fecha de apertura de la Cuenta, Fecha_Cuenta.

La **Cuenta_de_Cheques** tiene atributos especializados: Balance_de_Cheques, Disponible_en_Balance, Cargo_por_cheque.
La **Cuenta_de_Ahorro** tiene atributos especializados Balance_de_Ahorros, Tasa_de_interés.
La **Cuenta_de_Crédito** tiene atributos especializados: Crédito_original, Tasa_interés_crédito, Balance_cuenta_corriente.
La especialización permite la posibilidad de una **Cuenta_de_Ahorro** que sea **Cuenta_de_Cheques** y de una **Cuenta_de_Crédito,** que sea **Cuenta_de_Cheques**.

A la entidad de tipo más alta E de la generalización y de la especialización se le llama entidad genérica o entidad de supertipo. A las entidades E1, E2, ..., En, de niveles mas bajos se le denomina entidades de subtipo.

La diferencia fundamental entre la especialización y la generalización es que la primera especifica conjuntos traslapables de entidades de subtipo, mientras que la segunda especifica conjuntos mutuamente excluyentes de entidades de subtipo.

La generalización y la especialización, también pueden describirse en términos de herencia. La generalización es la interrelación jerárquica, en las cuales los atributos de un supertipo son propagadas o heredados a una entidad de subtipo y adicionalmente ese subtipo, puede tener atributos específicos diferentes a los encontrados en el supertipo. En la especialización los atributos de un supertipo, son heredados a las entidades de subtipo, y adicionalmente cada uno de esos subtipos contiene atributos específicos.

Una entidad supertipo en una interrelación puede ser una entidad subtipo en otra.
Cuando se encuentran estructuras con combinaciones de supertipo/subtipo, la estructura es llamada de jerarquía supertipo/subtipo.

3.3 Modelo Conceptual con Entidad Vínculo Extendido.

Las extensiones al modelo original hicieron dificil la aprobación de un modelo entidad vínculo estandar, en vez de ello, los modelos se han venido agrupando en familias.

3.3.1 Familias de modelos con EVE.

Para la construcción de modelados EV y EVE no se ha establecido una notación y una metodología como estándar.
Se puede hablar de familias de modelos EVE.
De las familias de modelos EVE, los modelos más utilizados son los de Chen, de Teorey y Reiner, de Everest y los de IDEF1X (desarrollado por la Fuerza Aérea de los EU).
Las principal diferencia en las metodologías consiste en que algunas extienden EV el grado de las relaciones a tres mientras otros solo permiten dos.
Aunque difieren en su notación, los objetos visuales del diagrama de un modelo puede encontrar su equivalente en los objetos visuales de otra.

La notación del modelo de Chen y del modelo de Teorey resultan parecidos. Lo mismo acontece con la notación del modelo de Everest y de IDEF1X

En los diagramas de Chen se tienen las siguientes representaciones:

Entidades.- Mediante cajas o rectángulos.
Interrelaciones.- Admite solo interrelaciones unarias y binarias representadas con rombos.
Conectividad de Interrelaciones.- Se representan con etiquetas 1 (uno) o M (Muchos) en las líneas de conexión que van de las relaciones a las entidades.
Especialización y Generalización.- La especialización se representa con un triángulo del cual se conecta por arriba con la entidad supertipo y por abajo con las entidades subtipos. El grosor de la línea de conexión, proporciona la diferencia entre especialización y generalización. Las líneas con mayor grosor, representan a la generalización y la de menor grosor a la especialización.

En Teorey y Reiner se diagraman:

Entidades.- Mediante cajas o rectángulos.
Interrelaciones.- Con rombos para las interrelaciones unaria y binaria, y triángulos para las interrelaciones ternarias.
Conectividad de las interrelaciones. Se expresa visualmente en las aristas del polígono de la relación, la conectividad muchos se representan, con una arista sombreada, y la conectividad de uno, con una arista sin sombrear.
Especialización y Generalización.- La especialización y la generalización se identifican con objetos visuales diferentes. Una flecha con su punta hacia arriba representa a la especialización, una flecha señalando hacia arriba pero con base representa a la generalización. Las conexiones de supertipo salen de la punta de flecha, y las de subtipo llegan por abajo.

En Everest se representan:

Entidades.- Mediante cajas o rectángulos.
Interrelaciones.- No se representan con un objeto visual o icono como en las metodologías descritas anteriormente. Las entidades se conectan con líneas y en ambos extremos de la conexión se establece el grado mínimo y máximo de conectividad de la relación. Con una línea vertical, se representan la conectividad de uno, y con tres líneas que parten de un extremo de la relación y llegan al extremo de la entidad, semejando las líneas unas patas de gallo. Esta metodología excluye a las relaciones ternarias.
Especialización y Generalización.- Se representa con un triángulo diminuto colocado debajo de la entidad supertipo.De la base del triángulo diminuto, parte una línea de conexión hacia los subtipos.

En IDEF1X se diagraman:

Entidades. - Mediante cajas o rectángulos.

Interrelaciones.- No cuenta con icono propio. Las líneas de conexión entre entidades colocan un círculo en los extremos de la línea que une a las entidades. La conectividad uno, se representa con un icono sin rellenar y la de muchos con un círculo relleno.
Especialización y Generalización.- La generalización se representa como un círculo sin rellenar con una línea horizontal como base, de la parte de arriba del círculo parte la conexión al supertipo y de la línea horizontal parte la línea hacia el subtipo.

El editor y el traductor del modelo de Entidad Vínculo Extendido del proyecto, emplea la representación de Teorey y Reiner.

El criterio para la selección de este tipo de representación se basó en la mayor claridad aportada al modelo EVE, al tratar visualmente el grado y la conectividad de las relaciones. Con la representación de Teorey se hace más distinguible la conectividad de las relaciones unarias, binarias y ternarias.
A continuación presentamos el diccionario de iconos de Teorey :

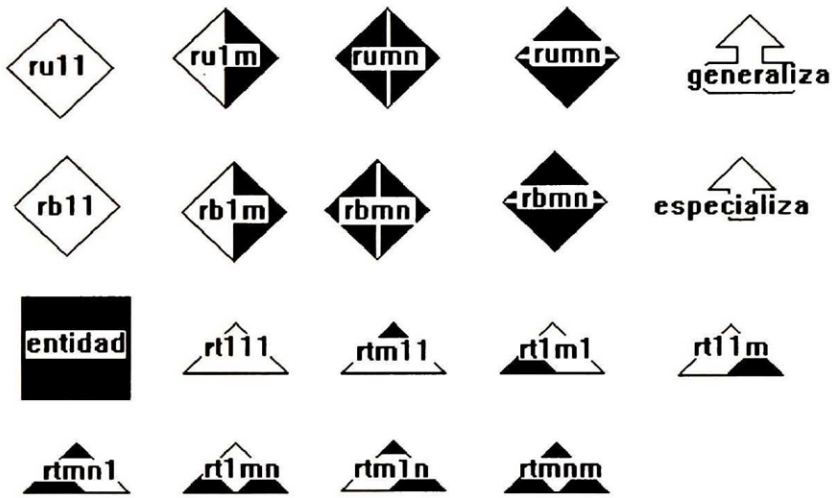


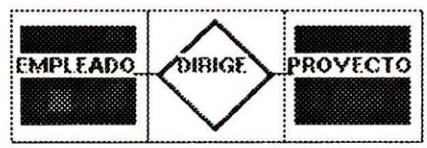
Figura.3.1 Diccionario de iconos de Teorey

3.3.2. Diagramación de roles con EVE de Teorey.

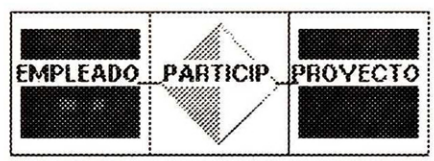
Presentaremos a continuación, la semántica de los iconos de Teorey, con algunos ejemplos de roles.

La interrelación de grado 1 conectividad 1:1 de un **Empleado** es **Esposo_de** otro **Empleado** se representa.

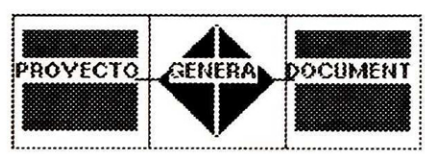
La interrelación un **Empleado Dirige** un **Proyecto**, y un Proyecto es dirigido por un Empleado se representa.



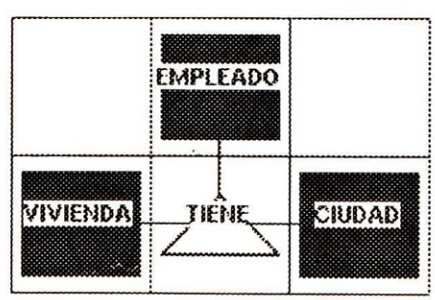
La interrelación un Empleado Participa en un **Proyecto** y en un Proyecto Participa más de un Empleado se representa.



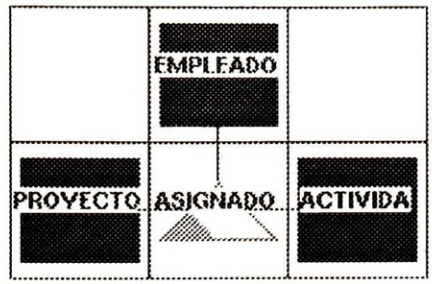
La interrelación un **Proyecto Genera** más de un **Documento** y un Documento puede Generarse con información de más de un Proyecto se representa.



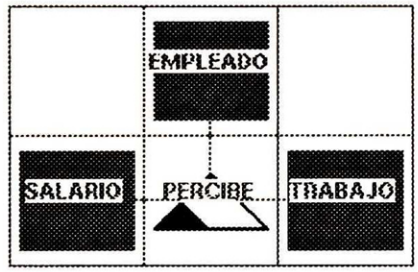
La interrelación un **Empleado Tiene** más de una **Vivienda** en más de una **Ciudad**. En donde para cada Empleado y Ciudad se tiene una Vivienda, y en donde por el identificador de Vivienda y Ciudad podemos reconocer al Empleado. Y con el par Empleado Vivienda podemos reconocer la Ciudad donde se localiza.
La interrelación se representa.



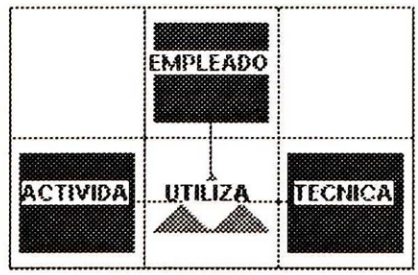
La interrelación un **Empleado** es **Asignado_a** a más de un **Proyecto** desempeñando alguna **Actividad**. Pero en donde en cada Proyecto el Empleado es Asignado_a una actividad y cada Actividad del Proyecto la efectúa un Empleado se representa.



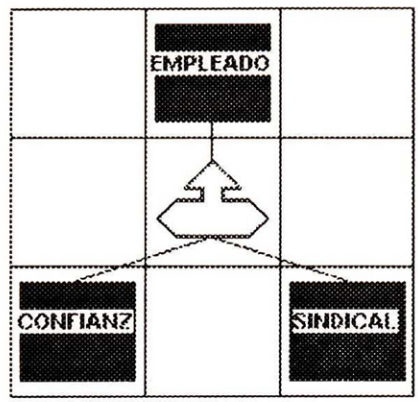
La interrelación un **Empleado Percibe** más de un **Salario** efectuando más de un **Trabajo**. Pero en donde
podemos reconocer el Trabajo desempeñado por un Empleado por el Salario percibido, se representa.



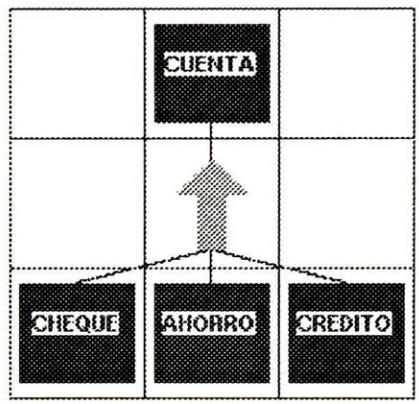
La interrelación un **Empleado Utiliza** en más de una **Actividad** más de una **Técnica**. Y una Técnica es empleada por más de un Empleado en más de una Actividad. Y una Actividad más de un Empleado Utiliza más de una Técnica se representa.



La interrelación jerárquica de generalización **Es_Un** entre **Empleado** y empleado de **Confianza** y **Sindical** se representa.



La interrelación jerárquica de especialización **Es_una** entre **Cuenta** y **Cuenta_de_Cheques, Cuenta_de_Ahorro, Cuenta_de_Crédito** se representa.



3.4 Módulos de EVEX de diseño conceptual.

Habiendo explicado las bases del modelo entidad vínculo extendido estamos en posibilidad de realizar diagramas eve.
Para diagramar hemos sugerido en la metodología con evex emplear el editor de atributos y el diagramador de evex.
A continuación describiremos los módulos de diseño conceptual, los algoritmos de interfaz con el usuario y los procedimientos para su uso.
De la figura 1.1 del primer capítulo explicaremos los módulos que a continuación esquematizamos:

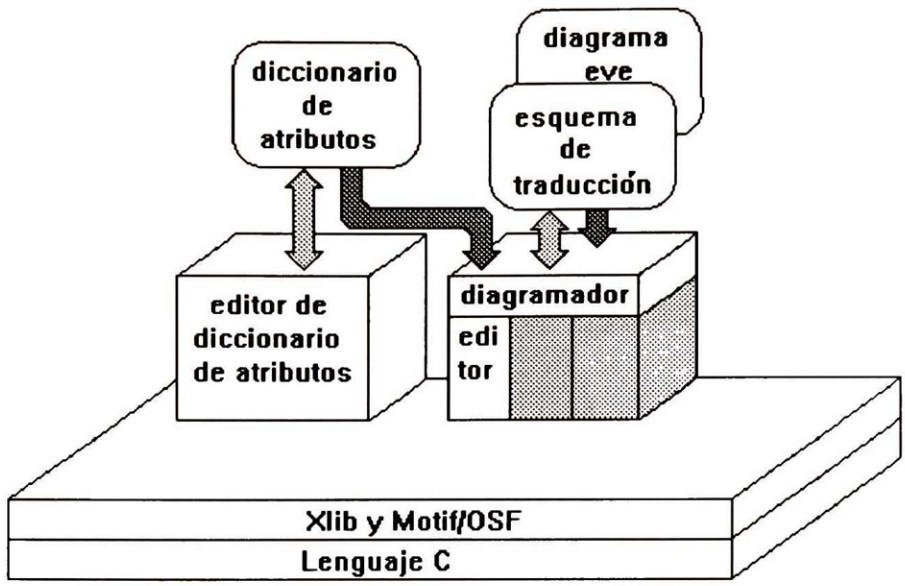


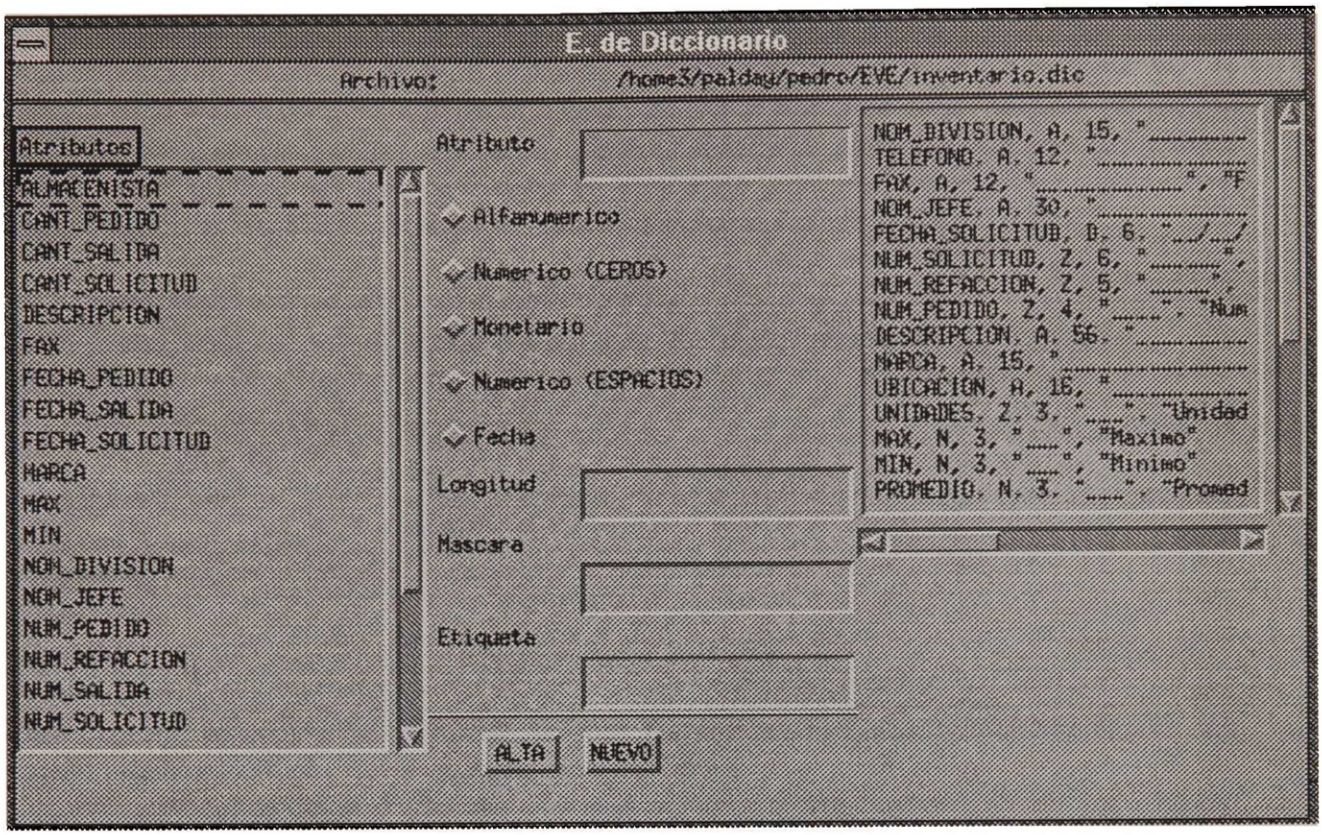
Figura 3.2 Módulos de EVEX para el diseño conceptual.

3.4.1 Editor de Diccionario de Atributos.

El programa Editor del Diccionario de Atributos permite crear un diccionario de datos en una notación entendible por el modelador evex, utilizando despliegues visuales.

3.4.1.1 Descripción.

El Diccionario consta: de una ventana con cinco áreas, un área superior, un área izquierda, una central, una derecha y un área inferior. En el área superior, se escribe el nombre del archivo de trabajo.En el área izquierda, se presenta una lista ordenada de nombres de atributos del diccionario actual. El área central, es el área de edición donde se modifican y dan de alta los atributos seleccionados. El área derecha, despliega el diccionario en la notación entendible por el modelador. El área inferior es para Mensajes generales del programa.
El área central contiene un menú oculto, que se activa al presionar el botón derecho del ratón. El menú consta de opciones para realizar un Nuevo diccionario, Cargar un diccionario y Salir. El área derecha de la ventana principal contiene un menú oculto, con opciones de Graba y Graba como.



3.4.1.2 Procedimiento.

Existen dos formas de ejecutar el diccionario de atributos, en una el programa se ejecuta de manera aislada y en la otra como un submenú del proyecto integrado **evex**.
De forma aislada, el programa se ejecuta tecleando desde el sistema operativo **dic**. De forma integrada ejecute **evex**. Seleccione del menú principal la opción Diccionario y de ahí la subopción Edita.

Si ejecuta el programa de manera aislada, se asume la creación de un nuevo diccionario, por lo cual el área de edición no tendrá datos. Si lo ejecuta como proyecto integrado, arma el nombre del diccionario de trabajo utilizando el nombre sin extensión del nombre de diagrama EVE activo y le añade la terminación dic, si encuentra un archivo con tal nombre, lo activa para su edición.
El diccionario despliega la ventana de edición de diccionario con sus áreas previamente descritas.

Para capturar los datos del atributo (nombre, tipo de datos, máscara de datos,etiqueta) , se teclean en el área central de la ventana. Dando un clic al botón de Alta del área central, se incorpora en el orden correspondiente, el nombre del atributo en la lista del área izquierda y como último renglón del área de la derecha, el nombre junto con los atributos adicionales. Cada renglón del área derecha sigue la gramática descrita a continuación:

diccionario_atributos →(descripción_atributo NL)$^{+}$
descripción_atributo → nom_atributo, tipo_dato, longitud, máscara, letrero
nom_atributo →letra (dígito | letra | _)*
tipo_dato→A | N | D | C
longitud→digito(digito)*
máscara→"(_)*"
letrero→"(digito | letra | _ | | ? | :)*"
letra→a-z | A-Z
dígito→0-9
NL→' \n'

Para capturar los datos de un nuevo atributo se presiona el botón etiquetado Nuevo. La acción limpiará los datos que se encuentren en la forma.
Los menúes ocultos del área derecha y central, se activan al presionar el botón derecho del ratón.
El menú oculto del área central con la opción Recupera permite editar un diccionario previamente grabado con el programa.
La opción Nuevo del menú, permite iniciar un nuevo diccionario.
El menú oculto del área derecha tiene opciones para grabar con el nombre del diccionario activo y para grabar con otro nuevo nombre con Graba como.
El diccionario graba el contenido del área derecha en un archivo con extensión dic.

3.4.1.3. Implementación.

3.4.1.3.1 Estructuras de datos.

La estructura base para sustentar el diccionario es un arreglo de elementos de diccionario de longitud MXELE. Donde el elemento de diccionario se compone de:

3.4.1.3.2 Algoritmos.

El algoritmo del Diccionario y su interfaz visual se presenta a continuación:

Algoritmo **Diccionario.**
Crea forma principal.
    Crea forma 1
        Crea Letrero "Archivo:"
        Crea Texto Nombre de Diccionario
    Crea forma 2
        Crea forma 2.1
            Crea Letrero "Atributos"
            Crea Lista Lista de Atributos Ordenados
            Llamado a función por evento. Selección Simple, Selecciona un atributo
        Crea forma 2.2
            Crea Letrero "Atributo:"
            Crea Texto Nombre Atributo
            Crea Letrero "Longitud:"
            Crea Texto Longitud
            Arreglo de Cuadros de Selección de Tipo de Datos
            Arreglo de Llamado a funciones por evento. Selección
                de Cuadro del tipo de Dato.
            Crea Letrero "Máscara"
            Crea Texto "-----"
            Crea Letrero "Etiqueta"
            Crea Texto Letrero de Atributo
            Crea Botón Alta
            Llamado a función por evento: Alta a elemento
            Crea Botón Nuevo
            Llamado a función por evento. Limpia captura de elemento.
        Crea forma 2.3
            Crea Texto con barra deslizante Diccionario de Atributos.
    Crea forma 3
        Crea Letrero de Error
Crea Menú Oculto. Central.
Llamado al Manejador de Eventos. Presión de Botón, Menú

CreaOpción Nuevo diccionario
Llamado a función por evento. Nuevo diccionario.
Crea Opción Recupera diccionario
Llamado a función por evento: Recupera diccionario.
Crea Opción Salir
Crea Menú Oculto. Derecho.
Llamado al Manejador de Eventos Presión de Botón, Menú de Diccionario
Crea Opción Graba.
Llamado a función por evento. Graba diccionario
Crea Opción Graba como.
Llamado a función por evento. Graba como
Activa Ciclo de Eventos Principal.

3.4.2 Editor de Diagramas EVE.

3.4.2.1 Descripción.

El Editor de diagramas eve se despliega en una ventana compuesta de un área de Menú Principal, un área de Archivos, un área de Dibujo, un área de Traducción, un área de Posición de Dibujo y un área de Mensajes.
El Menú Principal se localiza en la parte superior de la ventana. Inmediatamente abajo, se localiza el área de Archivos, abajo de esta área ocupando la porción izquierda del Editor y Traductor, se localiza el área de Dibujo, ocupando la porción derecha se localiza el área de Traducción y un área inferior a ambas, dividida en un área de Posición de Dibujo, en su parte izquierda y en un área de Mensajes en su parte derecha.
El área de Traducción se encuentra subdividida en un área superior para Errores de Traducción y un área inferior para Despliegue de Traducción.
El área de Archivos permite identificar el archivo de diagramas eve activo y el archivo de Diccionario activo.
El área de dibujo se distingue por contener una rejilla de dibujo delimitada con barras deslizantes horizontal y vertical.
Una malla de la rejilla delimita a una celda de dibujo. Cada icono del diagrama eve debe dibujarse en una celda de dibujo. El área real de dibujo es mayor al área de dibujo visible, de ahí la utilidad de las barras deslizantes. La posición x, y en el área de Posición de Dibujo indica la posición de la primera celda (celda superior izquierda) del área de dibujo visible.
Conforme se deslice horizontalmente o verticalmente las barras deslizantes, se mueve el área visible de dibujo y los valores de ubicación x, y de la primera celda.

Menú Principal.
El Menú Principal tiene opciones para Diagramas del Editor, opciones para el Editor del Diccionario de Atributos y opciones de Traducción.

Las opciones para los diagramas son de Nuevo para creación de un nuevo diagrama, Recupera para leer un diagrama eve, Graba y Graba como para guardar un diagrama con el nombre actual o con un nuevo nombre.
Las opciones del submenú de Diccionario son de Nuevo para sustituir el diccionario activo, de Recupera para leer un diccionario construido con el Diccionario de Atributos.
Las opciones Compilación son de Sintaxis para el Análisis sintáctico del modelado y de Traducción completa y Traducción por pasadas.

Ventana de Edición de Entidades.
Esta ventana se compone de cuatro áreas. Un área para la edición del nombre de la entidad, localizada en la parte superior, un área debajo de la anterior pero ocupando la porción izquierda, para edición de identificadores. Un área ubicada a la derecha de la anterior, para edición de atributos. Ambas áreas de edición contienen opciones de OK y Cancela.
Un área localizada en la parte inferior de la ventana, empleada para los botones de opción OK y Cancela.

Panel de Iconos.
El panel de iconos está compuesto por los iconos del modelado eve de Teorey.
El panel es un menú oculto, que se activa al presionar el botón derecho del ratón en el área de Dibujo.
El panel consta en su primer menú de la entidad, la interrelación unaria, la interrelación binaria, la interrelación ternaria, la generalización y la especialización, el icono de conexión, de desconexión y de rotación.
Los iconos de interrelación se despliegan ordenados según su grado. Al posicionarse en un grado, el submenú despliega su conectividad. En el menú los grados de las relaciones se encuentran etiquetados. Facilitando la diferenciación entre las relaciones unarias y binarias.

3.4.2.2. Procedimiento.

Al iniciar la ejecución del programa editor de diagramas evex, el área de dibujo se encuentra vacío, para la construcción de un nuevo diagrama.

Selección de un icono del Panel.
Para activar el panel de iconos presione el botón derecho del ratón. Arrastre el ratón hasta el icono a seleccionar, haga clic en la celda donde desee incorporar el símbolo.
El nombre sobreescrito en el icono se pierde al insertarlo en una celda del diagrama.
En el diagrama eve se requiere que los iconos contengan identificadores referentes al universo del problema.
Los iconos también se asocian a un diccionario de atributos referente al universo del problema, en un procedimiento que hemos denominado de edición de entidades e interrelaciones.

Conexión y Desconexión de interrelaciones y entidades.

Después de seleccionar a entidades e interrelaciones, debemos unir unas con otras.

Para efectuar una conexión, seleccione el icono de conexión del panel, presione el botón izquierdo del ratón sobre la primer celda participante de la conexión y presione el botón izquierdo sobre la segunda celda.

Para desconectar los iconos de una interrelación seleccione el icono de desconexión y haga clics sobre las celdas participantes de la conexión que desea desligar.

Las conexiones entre entidades e interrelaciones es mediante líneas rectas. Parten de puntos específicos de una entidad y se dirigen hacia puntos específicos de una interrelación.

Un icono de interrelación unaria, con un rombo como símbolo, tiene dos puntos de conexión. Si se desea una conexión horizontal con una entidad, los vértices superior e inferior del rombo son los puntos de unión de las líneas; si su conexión es vertical los puntos de unión son los vértices izquierdo y derecho del rombo.

Un icono de interrelación binaria, con un rombo como símbolo, tiene dos puntos de conexión que pueden ser izquierdo y derecho o superior e inferior.

Un icono de interrelación ternaria, con un triángulo como símbolo, tiene tres puntos de conexión uno en cada punto medio de cada lado del triángulo .

Una entidad, rectángulo, tiene cuatro puntos de conexión: uno izquierdo, uno inferior, uno derecho y otro superior:

Pudiera pensarse que cada entidad estaría limitada a ligarse a lo más con cuatro interrelaciones diferentes. Para terminar con esa limitación, el editor de evex permite la repetición de entidades en otra parte del diagrama, dando a esa repetición otros cuatro posibles puntos de conexión.

Al implementarse el editor evex se planeó adelantar la verificación de ciertos errores durante el proceso de dibujo de diagramas. Cuando se construye un diagrama no se permite conectar más entidades que las especificadas en el grado de la relación. Tampoco es permitido conectar entidades a interrelaciones trazando la línea de conexión sobre otras entidades o interrelaciones.

Una conexión entre una interrelación y una entidad se representa en el diagrama con el trazado de una línea de enlace. Cada enlace fisico genera dos enlaces lógicos, los enlaces de salida del objeto y los enlaces de entrada al objeto.

Rotación de Iconos.

El icono de rotación del panel permite cambiar la orientación de la conectividad muchos (M) de una interrelación.

El sentido de rotación es inverso al de las manecillas del reloj, y lo que rota es el sombreado de muchos de la reprensentación de Teorey.

Por ejemplo, si se selecciona un icono de interrelación binaria uno a muchos (1:M) de izq.-der., y después selecciona el icono de rotar, el icono cambia a uno de interrelación binaria muchos a uno (M:1) superior-inferior, al seleccionar otra vez el icono de rotar, el icono binario cambia a uno con cardinalidad muchos a uno (M:1) de izq.-der., si vuelve a seleccionar el icono de rotar cambia el icono binario a uno con cardinalidad uno a muchos (1:M) superior-inferior, una nueva selección del icono de rotar, regresa al primer icono de relación binaria.

Borrado de Iconos.
Para borrar un icono debe presionar el botón central del ratón. Borrar un icono elimina al icono del diagrama, junto con su información lógica, llaves y descriptores y sus conexiones lógicas.

Edición de Entidades e Interrelaciones.
Con los procesos anteriormente descritos se puede dibujar un diagrama eve. Aunque falta editar las entidades e interrelaciones para nombrar y ligar sus atributos a un diccionario de datos.
Para asociar un diccionario a un diagrama seleccione en el Menú Principal la opción de Diccionario de Atributos, subopción Recupera.
En el área de Archivos debe desplegarse el nombre del diccionario.
Para editar una entidad o interrelación, haga clic en la entidad o interrelación de interés. Al momento surge la lista atributos ordenados relativos al Diccionario activo. Y una ventana de edición para la interrelación o entidad. Oprimiendo el botón izquierdo del ratón sobre algún atributo de la lista, se selecciona el atributo, y se introduce en el área de edición de identificadores o atributos.
En un principio el área activa es la de los identificadores, pero al presionar el botón OK del área de identificadores, el área activa se cambia al área de atributos.
Al presionar el botón OK en el área de de atributos se restablece el área activa a los identificadores.
El botón de Cancela presionado en el área de indentificadores o en el área de atributos borra el último atributo seleccionado de la lista.
El diagrama puede almacenarse en cualquier momento de su construcción seleccionando del Menú Principal la opción del editor de Graba o Graba Como.

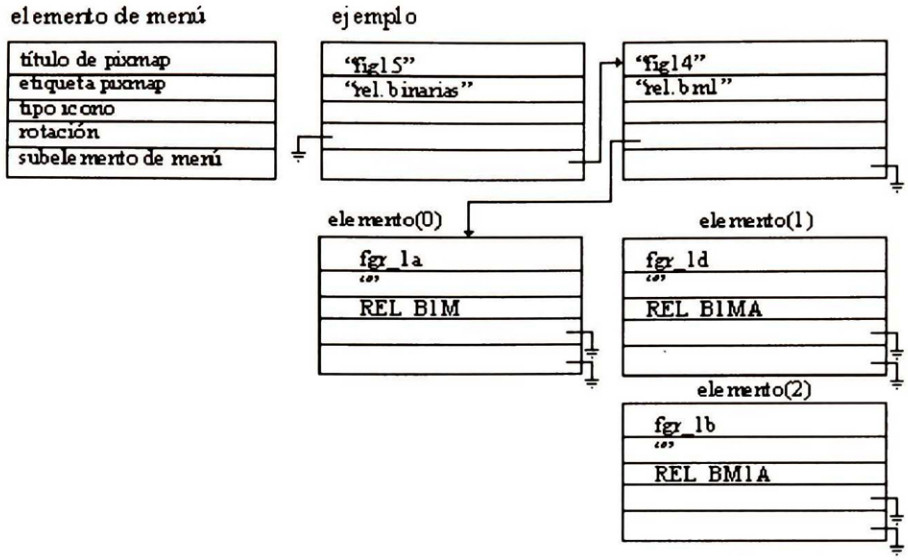
El formato de lectura y escritura de los diagramas eve es un formato con extensión chn, en el cual se describen las posiciones donde se encuentran los diferentes objetos de interés de eve: entidades, interrelaciones y las ligas entre ellos.

Los diagramas completos -dibujo con su asociación- pueden traducirse con las opciones de Compilación del Menú Principal.

3.4.2.3 Implementación.

3.4.2.3.1 Estructuras de datos.

El panel de iconos del Editor de diagramas eve se construye con elementos de menú con la siguiente estructura.



Donde el título de pixmap es el nombre del archivo de mapas de pixeles. La etiqueta de pixmap es el nombre que se sobreescribe al mapa de pixeles. El tipo de icono es una constante numérica que identifica la conectividad y grado de una relación.

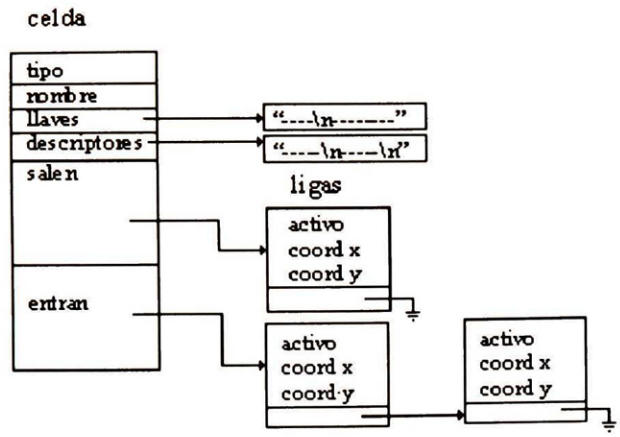
icono

número icono	
vértice arriba	x
	y
vértice abajo	x
	y
vértice izquierda	x
	y
vértice derecho	x
	y

ejemplo

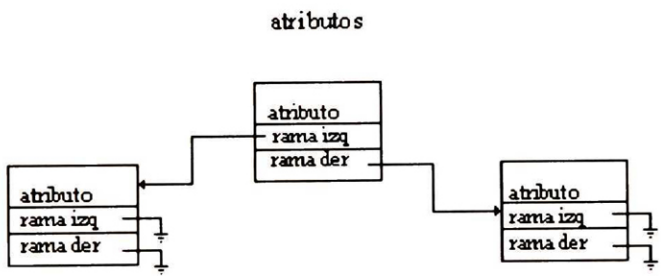
Rel Binaria 1a1	
vértice arriba	32
	4
vértice abajo	32
	59
vértice izquierda	4
	32
vértice derecho	59
	32

Un diagrama se define como una matriz MAX_WIDTH, MAX_HEIGHTde celdas, en donde cada celda tiene un tipo de icono Un valor negativo en el tipo de icono, indica ausencia de elemento de eve en la celda.



El tipo de icono de una celda representa el grado y conectividad de una interrelaciones, un tipo de jerarquía o una entidad.
El nombre es el identificador dado al icono en el universo del problema.

Los atributos de la lista de atributos para la edición de entidades e interrelaciones usan para su ordenamiento un árbol binario con la estructura siguiente:



3.4.2.3.2 Algoritmos.

Algoritmo. **Editor de diagramas EVEX.**
Crea forma principal.
Crea forma 1
Crea Letrero "Archivo:"
Crea Letrero Nombre de Archivo
Crea Letrero "Diccionario"
Crea Letrero Nombre de Diccionario
Crea Forma con barra deslizante
Crea Area de Dibujo.
Llamado al Manejador de Eventos. Presión de Botón, Selección de celda de dibujo.
Crea Barra deslizante Horizontal
Crea Barra deslizante Vertical
Asocia Area de Dibujo con Barras deslizantes

Llamado a función por evento. Barra Horizontal.
Llamado a función por evento. Barra Vertical.
Crea Letrero "Sintaxis"
Crea Texto con barra deslizante Resultado de Compilación.
Crea Letrero "Traducción"
Crea Texto con barra deslizante Resultado de Traducción.
Crea Menú Oculto. Traducción.
Llamado al Manejador de Eventos. Presión de Botón, Menú de Traducción.

Crea Opción Graba.
Llamado a función por evento. Graba diccionario
Crea Opción Graba como.
Llamado a función por evento. Graba como

Crea forma 2
Crea Letrero "Posición"
Crea Letrero x
Crea Letrero y
Crea Letrero de Errores Generales
Crea Menú de Archivo
Crea Opción. Nuevo.
Llamado a función por evento. Nuevo Diagrama de Chen.
Crea Opción. Recupera.
Llamado a función por evento. Recupera Diagrama de Chen.
Crea Opción. Graba.
Llamado a función por evento. Graba Diagrama de Chen.
Crea Opción. Graba como.
Llamado a función por evento. Graba Diagrama de Chen seleccionando nombre.
Crea Opción. Salir.
Llamado a función por evento. Salir.
Crea Menú Diccionario.
Crea Opción. Nuevo.
Llamado a función por evento. Nuevo Diccionario.
Crea Opción. Recupera.
Llamado a función por evento. Recupera Diccionario.
Crea Menú Compilación.
Crea Opción. Analiza Sintaxis.
Llamado a función por evento. Análisis Sintáctico de Diagrama Activo.
Crea Menú. Traducción.
Crea Opción. Traducción por pasadas.
Llamado a función por evento. Traducción por pasos.
Crea Opción. Traducción completa.
Llamado a función por evento. Traducción completa.
Activa Ciclo de Eventos Principal.

Crea Forma de Diálogo. Lista de Atributos.
    Crea Lista con barra deslizante. Lista de Atributos.
    Llamado a función por evento. Selecciona un atributo.

Crea Forma de Diálogo. Actualización de Entidad.
    Crea forma 1.
        Crea Letrero. "Identificador"
        Crea Texto. Identificador de Entidad.
    Crea forma 2.
        Crea Letrero. "Llave".
        Crea Texto. Lista de atributos llaves.
        Crea Botón. Ok.
        Llamado a función por evento. Acepta Lista de Atributos.
        Crea Botón. Cancela.
        Llamada a función por evento. Cancela Lista de Atributos.
        Crea Letrero. "Atributos"
        Crea Texto. Lista de Atributos.
        Crea Botón. Ok.
        Llamado a función por evento. Acepta Lista de Atributos.
        Crea Botón. Cancela.
        Llamado a función por evento. Cancela Lista de Atributos.
    Crea forma 3.
        Crea Botón. Ok.
        Llamado a función por evento. Acepta Datos de Entidad.
        Crea Botón. Cancela.
        Llamado a función por evento. Cancela datos de Entidad.

Algoritmo. **Conexión física y lógica de iconos.**

```
Conecta.
Parámetros:   x1, y1 . Posiciones de primer icono.
              x2, y2. Posiciones del segundo icono.
Selecciona Tipo de icono x1, y1
Caso ENTIDAD
    Selecciona Tipo de icono x2, y2
        Caso. ESPECIALIZACION
        Caso.GENERALIZACION
            Si no existe enlaces
                Inserta enlaces de entrada en x2, y2
                Inserta enlaces de salida  en  x1, y1
            fin si
```

```
                    Dibuja línea de conexión.


                Caso. INTERRELACIONES UNARIAS
                    Si enlaces de entrada x2, y2 < 1
                        Inserta enlaces de entrada en x2,y2
                        Inserta enlaces de salida en x1, y1
                    fin si
                    Dibuja línea de conexión.
                Caso.INTERRELACIONES BINARIAS
                    Si enlaces de entrada x2, y2 < 2
                        Inserta enlaces de entrada x2, y2
                        Inserta enlaces de salida en x1, y1
                    fin si
                    Dibuja línea de conexión
                Caso.INTERRELACIONES TERNARIAS
                    Si enlaces de entrada x2, y2 < 3
                        Inserta enlaces de entrada x2, y2
                        Inserta enlaces de salida x1, y1
                    fin si
                     Dibuja línea de conexión.
        fin Caso.
Caso ESPECIALIZACION.
Caso GENERALIZACION.
        Selecciona Tipo de Icono x2, y2
        Caso ENTIDAD.
        Si enlaces de salida x1, y1 < 1
                Inserta enlace de entrada x2, y2
                Inserta enlace de salida x1, y1
          fin si
        Dibuja línea de conexión.
        fin Caso.
Caso INTERRELACIONES UNARIAS.
        Selecciona Tipo de Icono x2, y2
        Caso ENTIDAD.
        Si enlaces de entrada x1, y1 < 1
                Inserta enlaces de entrada x1, y1
                Inserta enlaces de salida x2, y2
        fin si
        Dibuja línea de conexión.
        fin Caso.
Caso INTERRELACIONES BINARIAS.
           Selecciona Tipo de icono x2, y2
        Caso ENTIDAD
        Si enlaces de entrada x1, y1 < 2
```

```
                Inserta enlaces de entrada x1, y1
                Inserta enlaces de salida x2, y2
        fin si
        Dibuja línea de conexión.
        fin Caso.
Caso INTERRELACIONES TERNARIAS.
            Selecciona Tipo de icono x2, y2
        Caso ENTIDAD
        Si enlaces de entrada x1, y1 < 3
                Inserta enlaces de entrada x1, y1
                Inserta enlaces de salida x2, y2
        fin si
        Dibuja línea de conexión.
        fin Caso.
fin Caso.
```

El presente capítulo proporciona los recursos necesarios para realizar el diseño conceptual de un problema con el modelo entidad vínculo extendido empleando la herramienta automática.

Los conceptos básicos del modelo de Chen de entidad, atributo, grado y cardinalidad de las interrelaciones, así como los conceptos añadidos de grado tres en las interrelaciones y de jerarquías de supertipo y subtipo fueron explicados en el inicio del capítulo.

Posteriormente describimos una metodología para el modelo en cuestión y los módulos de edición del diccionario de atributos y de los diagramas que la herramienta automática posee para facilitar la representación conceptual del problema.

Cerramos el capítulo con algunos algoritmos y estructura de datos utilizados en la implementación de la herramienta.

DISEÑO LOGICO CON EVE

Como primera etapa del diseño lógico, el esquema conceptual de BD modelado con eve se traduce a esquemas de relación en 3FN, aplicando las reglas de transformación para las entidades e interrelaciones.
Inicialmente cada entidad del diagrama produce un esquema de relación y posteriormente dependiendo del grado y cardinalidad de las interrelaciones se trasladan los identificadores como atributos ajenos o se crean nuevas relaciones.
Cuando un diagrama se realiza en forma inadecuada sea por un análisis defectuoso o falta de dominio del modelo, las relaciones generadas con las reglas de transformación resultan redundantes, ambiguas o con pérdida semántica. Esto justifica la agregación de una segunda etapa en el diseño lógico, que consiste en la normalización por síntesis de los esquemas relacionales, con lo cual aseguramos la obtención de esquemas en 3FN.
En este capítulo explicamos las reglas de transformación para diagramas eve, presentamos algunos casos en los que estos violan la primera, segunda y tercera forma normal.
Confirmamos que para tales casos es necesario adicionar un proceso de normalización al diseño lógico. El algoritmo de normalización por síntesis se integra en la herramienta evex, por lo cual describimos sus fundamentos y su implementación.

4. 1 Transformación del diagrama EVE a esquemas de relación.

La metodología con eve propone traducir el modelo conceptual entidad vínculo extendido a un modelo lógico de los conocidos en base de datos: al modelo de red, relacional o jerárquico.
La metodología propuesta en el proyecto evex transforma un modelado eve a un modelo lógico relacional.
Definiremos por consiguiente, los conceptos básicos referentes a las relaciones.

Una relación es cualquier subconjunto tomado del producto cartesiano de n-dominios. Donde un dominio es un conjunto de valores, semejante a los tipos de datos de un lenguaje de programación.
A los miembros de una relación se les denomina tupla.
Una tupla tiene componentes v_1 ,v_2 , ..., v_n donde el componente v_1 pertenece a un dominio D_1, el componente v_2 pertenece al dominio D_2 y así sucesivamente.
Cuando a un componente v_1, v_2 , ... v_n se le denomina con un nombre, se le llama atributo.
Una relación compuesta de un conjunto de atributos con nombre se le denomina esquema de relación. Un esquema de relación se suele escribir como $R(A_1, A_2, ..., A_n)$
De una manera práctica, se identifica a un esquema de relación como una tabla, donde las filas son las tuplas y las columnas los componentes de la tupla.
Una colección de esquemas de relación forma un esquema de base de datos relacional.
Una llave es un conjunto de atributos S de una relación que identifican como única una tupla dentro de una relación R y no tienen un subconjunto propio que a su vez sea llave.

Una llave se dice compuesta cuando se compone de más de un atributo.
Una llave primaria es una llave diseñada por el modelador del esquema como el identificador principal de una tupla.
Una llave candidata es cualquier posible composición de atributos que satisface las condiciones de identificar de modo único una tupla y no contiene un subconjunto propio como llave.
Una llave alterna es una llave candidata que no es llave primaria de una tabla.
Una llave ajena es un conjunto de atributos que forman la llave primaria de otra tabla.
Un nombre de atributo que hace referencia a otro nombre de atributo previamente definido se denomina atributo sinónimo o alias de atributo.

El proceso de transformación de diagramas eve a esquemas de relación, traduce los diagramas en el siguiente orden: entidades, interrelaciones de grado uno, dos, tres, interrelaciones jerárquicas.
Las reglas de tranformación de las interrelaciones de eve dependen de su grado y conectividad.

4.1.1 Transformación de Entidades.

En un diagrama, una entidad se transforma en un esquema de relación. Los atributos de la entidad se convierten en los atributos del esquema de relación, la llave primaria de la entidad es el identificador único o llave primaria del esquema de relación.
La entidad Empleado transforma en un esquema de relación:

Empleado(Num_empleado, Nom_empleado, Rfc, Edad, Sexo)

En término de dependencias funcionales, tenemos que Nom_empleado, Rfc, Edad, Sexo dependen funcionalmente de Num_Empleado. Esto mismo se expresa como:

Num_Empleado $\rightarrow$ Nombre_Empleado, Rfc, Edad, Sexo

4.1.2 Transformación de Interrelaciones Unarias.

Una interrelación uno a uno (1 :1) o uno a muchos (1:M) unaria transforma el esquema de relación de la entidad participante, al añadirle a los atributos propios del esquema, los sinónimos de la llave primaria y los descriptores de la interrelación unaria.
Ejem. La interrelación unaria una a una Esposos entre Empleados transforma el esquema de relación de :

Empleado(Num_empleado, Nom_empleado, Rfc, Edad, Sexo)

en uno con:

Empleado(Num_empleado, Nom_empleado, Rfc, Edad, Sexo, Num_conyuge)

donde Num_conyuge es un sinónimo o alias de Num_empleado.

El esquema de relación Empleado ahora tiene como dependencia funcional:

Num_empleado $\rightarrow$ Nom_empleado, Rfc, Edad, Sexo, Num_conyuge

Una interrelación unaria muchos a muchos (M:N) crea un nuevo esquema de relación que toma por nombre el de la interrelación y como llave primaria la concatenación de llave primaria de la entidad y los sinónimos de la llave primaria, y como atributos descriptores, los descriptores de la interrelación.
Ejem. Una interrelación unaria Esposos entre Empleados donde interese los diversos matrimonios entre empleados.

Empleado(Num_empleado, Nom_empleado, Rfc, Edad, Sexo)

en uno con:

Empleado(Num_empleado, Nom_empleado, Rfc, Edad, Sexo, Num_conyuge)
Esposos(Num_empleado, Num_conyuge)

donde Num_conyuge es un sinónimo o alias de Num_empleado.

El nuevo esquema de relación Esposos tiene como dependencia funcional:

Num_empleado Num_conyuge $\rightarrow \varnothing$

4.1.3. Transformación de Interrelaciones Binarias.

Una interrelación binaria una a una (1:1) transforma el esquema de relación de una de las entidades al añadirle como descriptor, la llave primaria de la otra entidad participante en la interrelación más los descriptores de la interrelación.
Ejem. La interrelación Supervisa entre Empleado y Proyecto.

Empleado(Num_empleado, Nom_empleado, Rfc, Edad, Sexo)
Supervisa(Fecha_supervision)
Proyecto(Num_proyecto, Nom_proyecto, Fecha_inicio, Tiempo_estimado)

transforma el esquema de relación de Empleado en

Empleado(Num_empleado, Nom_empleado, Rfc, Edad, Sexo, Num_proyecto, Fecha_supervision)

El nuevo esquema de relación tiene como dependencia funcional:

Num_empleado $\rightarrow$ Nom_empleado, Rfc, Edad, Sexo, Num_proyecto, Fecha_supervision

Una interrelación binaria una a muchas (1:M) transforma el esquema de la relación de la entidad con conectividad M a uno al cual le añade la llave primaria de la entidad con conectividad 1, más los atributos descriptores de la interrelación.
Ejem. Un rol de un Proyecto Emplea a muchos Empleados.

Proyecto(Num_proyecto, Nom_proyecto, Fecha_inicio, Tiempo_estimado)
Emplea
Empleado(Num_empleado, Nom_empleado, Rfc, Edad, Sexo)

transforma el esquema Empleado a:

Empleado(Num_empleado, Nom_empleado, Rfc, Edad, Sexo, Num_proyecto)

El esquema de relación Proyecto tiene como nueva dependencia funcional:

Num_empleado $\rightarrow$ Nom_empleado, Rfc, Edad, Sexo, Num_proyecto, Fecha_supervisión

Una interrelación binaria con conectividad muchos a muchos (M:N) crea un nuevo esquema de relación que toma por nombre el de la interrelación y toma como llave primaria la concatenación de las llaves primarias de las entidades participantes y como atributos descriptores los atributos descriptores de la interrelación.
Ejem. Un rol Empleado Labora en muchos Proyectos y un Proyecto emplea muchos Empleados.

Proyecto(Num_proyecto, Nom_proyecto, Fecha_inicio, Tiempo_estimado)
Labora
Empleado(Num_empleado, Nom_empleado, Rfc, Edad, Sexo)

crea un nuevo esquema de relación Labora.

Labora(Num_proyecto, Num_empleado)

El esquema de relación Labora tiene como nueva dependencia funcional:

Num_proyecto Num_empleado $\rightarrow \varnothing$

4.1.4. Transformación de Interrelaciones Ternarias.

Una interrelación ternaria una a una a una (1:1:1) crea un nuevo esquema de relación cuyo nombre es el de la interrelación. El número de llaves de la nueva interrelación se forma del número de posibles combinaciones entre dos de las llaves primarias de las entidades participantes, con lo que tendremos una llave primaria y dos alternas.
Los descriptores de la nueva relación serán los descriptores de la interrelación

Ejem.

Empleado(Num_empleado, Nom_empleado)
Proyecto(Num_proyecto, Nom_proyecto, Fecha_inicio, Tiempo_estimado)
Actividad(Actividad, Hrs_actividad)
Asignado_a(Fecha_asignacion)

crea un nuevo esquema de relación

Asignado_a(Num_empleado, Num_proyecto, Actividad, Fecha_asignacion)
Asignado_a(Num_empleado, Num_proyecto, Actividad, Fecha_asignacion)
Asignado_a(Num_empleado, Num_proyecto, Actividad, Fecha_asignacion)

Las dependencias funcionales de la nueva relación son:

Num_empleado, Num_proyecto $\rightarrow$ Actividad Fecha_asignacion
Num_empleado, Actividad $\rightarrow$ Num_proyecto Fecha_asignacion
Num_proyecto, Actividad $\rightarrow$ Num_empleado Fecha_asignacion

Una interrelación ternaria con conectividad muchas a una a una (M:1:1) crea una nuevo esquema de relación que toma por nombre el de la interrelación y como llave primaria la concatenación de la llave primaria de una de las entidades con conectividad uno y la llave primaria de la entidad con

conectividad de muchos y como descriptores los descriptores de la interrelación La llave primaria con conectividad uno excluida y la llave primaria de la entidad conectividad de muchos forman la llave alterna del nuevo esquema de relación.

Ejem.

Empleado(Num_empleado, Nom_empleado)
Proyecto(Num_proyecto, Nom_proyecto, Fecha_inicio, Tiempo_estimado)
Actividad(Actividad, Hrs_actividad)
Asignado_a(Fecha_asignacion)

Crea un nuevo esquema de relación

Asignado_a(Num_empleado, Num_proyecto, Actividad, Fecha_asignacion)
Asignado_a(Num_empleado, Num_proyecto, Actividad, Fecha_asignacion)

Las dependencias funcionales para la nueva relación son:

Num_empleado, Num_proyecto → Actividad Fecha_asignacion
Num_proyecto, Actividad → Num_empleado Fecha_asignacion

Una interrelación ternaria con conectividad muchas a muchas a una (M:N:1) crea un nuevo esquema de relación que toma por nombre el de la interrelación, y como llave primaria la concatenación de llaves primarias de las entidades con conectividad de muchos, y como descriptores, la llave primaria de la conectividad uno más los descriptores de la interrelación.
Ejem.

Empleado(Num_empleado, Nom_empleado)
Proyecto(Num_proyecto, Nom_proyecto, Fecha_inicio, Tiempo_estimado)
Actividad(Actividad, Hrs_actividad)
Asignado_a(Fecha_asignacion)

crea un nuevo esquema de relación

Asignado_a(Num_empleado, Num_proyecto, Actividad, Fecha_asignacion)

La dependencia funcional para la nueva relación es:

Num_empleado, Num_proyecto → Actividad Fecha_asignacion

Una interrelación ternaria con conectividad muchos a muchos a muchos (M:N:M) crea un nuevo esquema de relación que toma por nombre el de la interrelación y como llave primaria la concatenación de llaves primas de las tres entidades participantes de la interrelación y como descriptores los descriptores de la interrelación.
Ejem.

Empleado(Num_empleado, Nom_empleado)
Proyecto(Num_proyecto, Nom_proyecto, Fecha_inicio, Tiempo_estimado)
Actividad(Actividad, Hrs_actividad)
Asignado_a(Fecha_asignación)

crea un nuevo esquema de relación

Asignado_a(Num_empleado, Num_proyecto, Actividad, Fecha_asignación)

La dependencia funcional para la nueva relación es:

Num_empleado, Num_proyecto Actividad → Fecha_asignación

4.1.5 Transformación de Interrelaciones de Jerarquías.

Las interrelaciones jerárquicas de generalización y especialización se transforman a esquemas de relación de manera semejante.
El esquema de relación del supertipo después de la transformación contiene como descriptores, los descriptores comunes a todos los subtipos, y como llave primaria la especificada. Los subtipos heredan como llave primaria la llave del supertipo y como descriptores sus atributos especificados.
Ejemplo.- La generalización de **Empleado** en **Sindicalizado** y **Confianza** transforma los esquemas

Empleado(Num_empleado, Nom_empleado, Rfc, Sexo)
Sindicalizado(Nivel_Sindicalizado, Cuota, Nom_empleado)
Confianza(Nivel_Confianza, Nom_empleado)

a

Empleado(Num_empleado, Nom_empleado, Rfc, Sexo)
Sindicalizado(Num_empleado, Nivel_sindicalizado,Cuota)
Confianza(Num_empleado, Nivel_confianza)

Un aspecto importante en la especialización y generalización es que el comportamiento de las llaves y los atributos descriptores es diferente durante los procesos de modificación e inserción de datos.

4.2 Transformación de diagramas eve con EVEX.

Habiendo explicado las reglas de transformación de diagramas eve a esquemas relacionales, pasaremos a describir la implementación de los módulos de análisis y traducción que forman parte del diagramador de acuerdo a como lo señalamos en el primer capítulo.

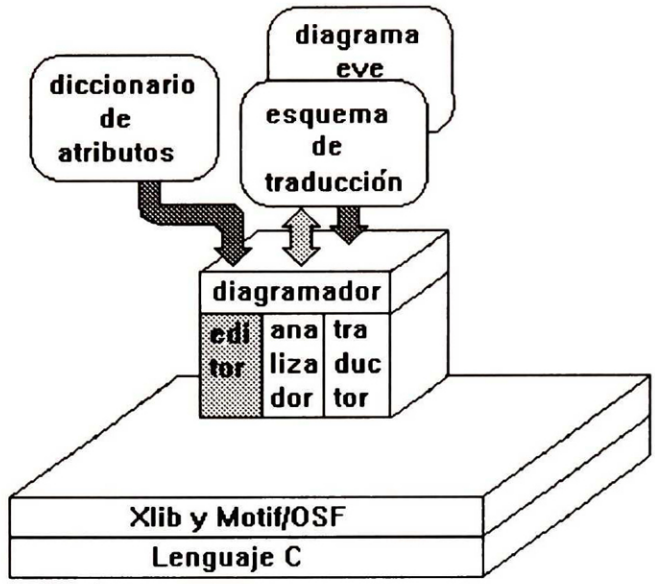


Figura 4.1 Módulo de Análisis y Traducción de EVEX.

4.2.1 Descripción.

La traducción del modelado con eve se ejecuta en los módulos Analizador y Traductor de evex.
Conforme se va efectuando el dibujo de su modelo eve, se puede seleccionar la subopción Sintaxis dentro de la opción Compila del Menú Principal del diagramador de evex, para encontrar los errores sintácticos.
Los errores y advertencias detectados van acompañados de la posición de la celda donde aconteció. Cuando existe algún error no se permite continuar con la traducción, las advertencias en cambio ejecutarán alguna acción correctiva que permita la traducción.
Los procesos de sintaxis y traducción se aplican de la celda superior izquierda con posición (0,0) a la celda inferior derecha del editor.

El analizador sintáctico realiza dos tareas de verificación: la de comprobación de sintaxis y la de integridad y consistencia.
En la comprobación de sintaxis el analizador asegura que las interrelaciones tengan enlaces de entrada y que coincidan estos en número con el grado de interrelación del icono, así como que las entidades tengan al menos un enlace de salida.
La comprobación de integridad y consistencia asegura que los datos se encuentren completos.
Comprueba la existencia de nombres para aquellas interrelaciones que al transformarse producen un nuevo esquema de relación, como son las interrelaciones ternarias con cualquier cardinalidad o las binarias y unarias muchas a muchas.

Con el modelador eve se permite la repetición de entidades, mediante la repetición de nombres.
En este caso el analizador verifica que las ocurrencias repetidas conserven los mismos atributos llave y descriptores.

Cuando no se conservan los mismos atributos, el analizador manda un mensaje de advertencia y continua creando una entidad con el nombre común y con el atributo llave formado de la unión de todos los atributos llaves de las entidades y con atributo descriptor formado de la unión de todos los atributos descriptores de la entidad.

En el caso de las jerarquías de generalización y especialización, la comprobación se lleva a cabo verificando primero que coincidan el número de enlaces de entrada de cada interrelación eve con el grado de interrelación del icono. Verifica la existencia de nombre para aquellas interrelaciones que al transformarse producen un nuevo esquema de relación, como son las interrelaciones binarias y unarias muchos a muchos y las interrelaciones ternarias.
Por un lado, el modelador eve permite la repetición de entidades para un diagrama, y por el otro, el analizador verifica que las ocurrencias repetidas de una entidad conserven los mismos atributos, como son la llave primaria y los descriptores. Si no coinciden el modelador manda una advertencia y uniformiza las entidades, uniendo las llaves primarias especificadas en las distintas ocurrencias en una llave primaria y uniendo todos los atributos descriptores en un descriptor.
En las jerarquías de generalización y especialización, el analizador resuelve como llave primaria de las entidades subtipo a la llave primaria del supertipo, los atributos comunes entre los subtipos se agregan como atributos descriptores al supertipo. A un mismo nivel las entidades subtipo (hermanas) no deben coincidir en todos sus atributos descriptores. Cuando existe anidamiento jerárquico se procede de la misma manera, la llave primaria del supertipo se hereda al subtipo y los atributos comunes de los subtipos suben al supertipo.
Las opciones de traducción completa y la de traducción por pasadas incluyen el análisis sintáctico, y se ejecutan si no se tienen errores en el análisis.

4.2.2. Procedimiento.

El proceso de traducción se aplica en el siguiente orden: traducción de jerarquías de generalización y especialización, de interrelaciones unarias con conectividad creciente uno a uno, uno a muchos, muchos a muchos, de interrelaciones binarias con conectividad uno a uno, uno a muchos, muchos a muchos y de interrelaciones ternarias con conectividad creciente.
El algoritmo de traducción para conectividad binaria uno a uno ha tomado como criterio escoger la entidad a la cual se le van agregar las llaves ajenas en base a su ubicación en la interrelación.
La entidad izquierda de una interrelación binaria horizontal uno a uno (izquierdo-derecho) añade como descriptores la llave primaria de la entidad derecha.
La entidad superior de una interrelación binaria vertical uno a uno (superior-inferior) añade como descriptores la llave primaria de la entidad inferior.
La traducción completa despliega los esquemas de relación resultantes de la traducción del diagrama, mientras que la traducción por pasadas va dando los esquemas de traducción después de aplicar cada paso de traducción de entidades y de interrelaciones.
En el área de Traducción existe un menú oculto con opciones de Graba y Graba como.
La traducción por pasadas y la completa se graban tal como salen en su ventana de despliegue con extensiones lst y rel, respectivamente.

El archivo de esquemas de relación con extensión rel, sigue la gramática:

traducción→ (relación nom_rel NL(llave(nom_llave NL)+ finllave)*
descriptor (nom_descriptor NL)* findescriptor)$^{+}$
relación → #name rel NL
llave → #keys NL
finllave → #end keys NL
descriptor → #desc NL
findescriptor → #end desc NL
nom_rel →id
nom_llave→ id
nom_descriptor → id (Ref_G nom_rel | Ref_E nom_rel)
id →letra (dígito | letra | _)*
NL→' \n'

4.2.3 Implementación.

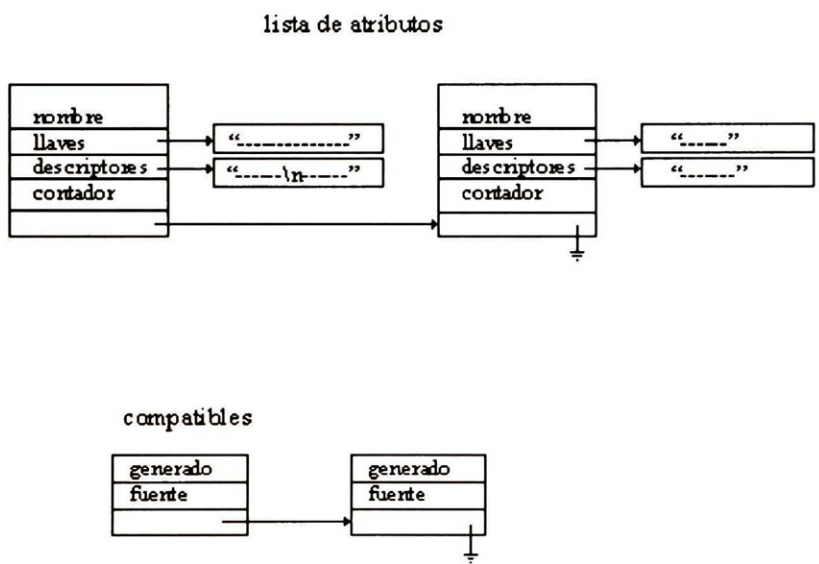


Figura 4.2. Estructuras de datos empleadas en la traducción.

4.3 Justificación de la Normalización.

Aun cuando un buen diagrama eve se traduce a relaciones en 3FN Teorey [Teorey86], es posible que al diagramar se excluyan interrelaciones entre entidades, esto provocará que en la traducción se obtengan relaciones no en 3FN.

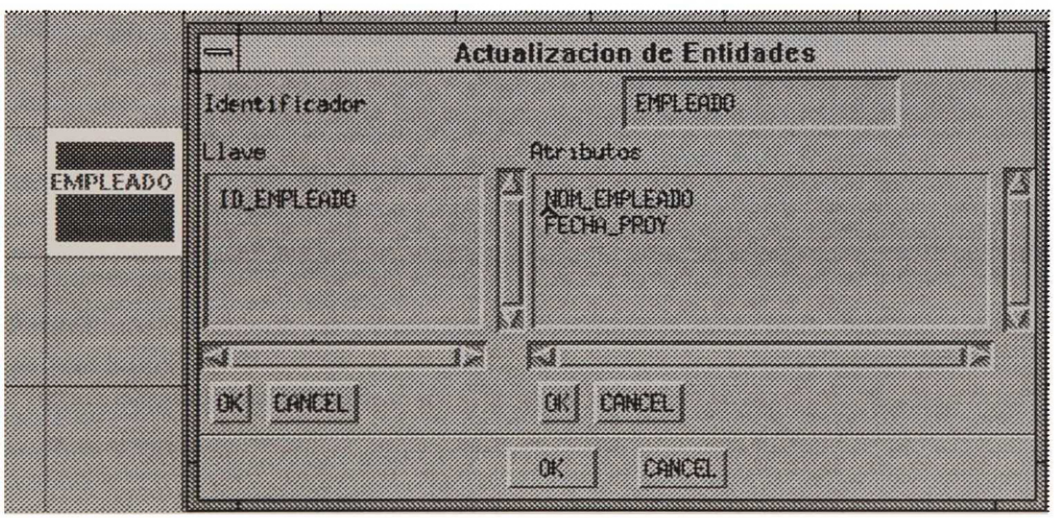
En la metodología que proponemos se plantea aplicar el algoritmo de normalización de Bernstein a las relaciones traducidas.
Así si excluimos interrelaciones en nuestro diagrama, pero agregamos las dependencias funcionales en las relaciones, como entrada al algoritmo de normalización, y aplicamos el algoritmo, obtendremos las relaciones en 3FN.

4.3.1 Diagramas que violan las Formas Normales.

4.3.1.1 Primera Forma Normal.

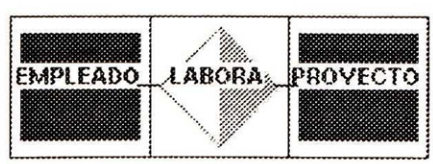
Recordando las reglas de normalización. Se dice que una relación se encuentra en primera forma normal (1FN) si todos los atributos de la relación tienen un solo valor para cada instancia, o lo que es lo mismo una relación está en 1FN cuando no contiene grupos repetitivos como atributos.

Si en nuestro diagrama eve modelamos una entidad Empleado con un identificador id_empleado y con atributos un nombre de empleado y muchas fechas de proyectos.



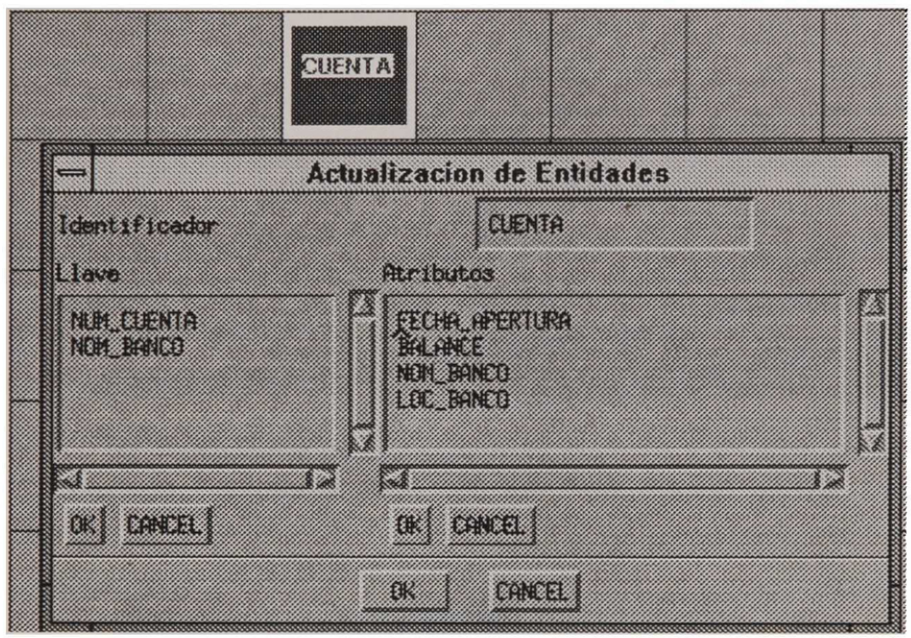
Estaríamos omitiendo las dependencias funcionales entre:
id_proyecto → fecha_proyecto e id_proyecto → id_empleado.

El modelado correcto con eve sería dos entidades Empleado y Proyecto con una interrelación binaria (1:M) entre Empleado y Proyecto.

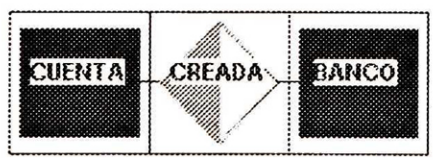


4.1.3.2. Segunda Forma Normal.

Una relación se dice que se encuentra en 2FN si además de estar en 1FN todos los atributos que no forman parte de la llave dependen del identificador único completo.
Si en nuestro modelado con eve diagramamos una entidad CuentaBanco con identificador único compuesto de num_cuenta, num_banco y atributos balance, fecha_apertura, nom_banco y localización del banco: loc_banco.



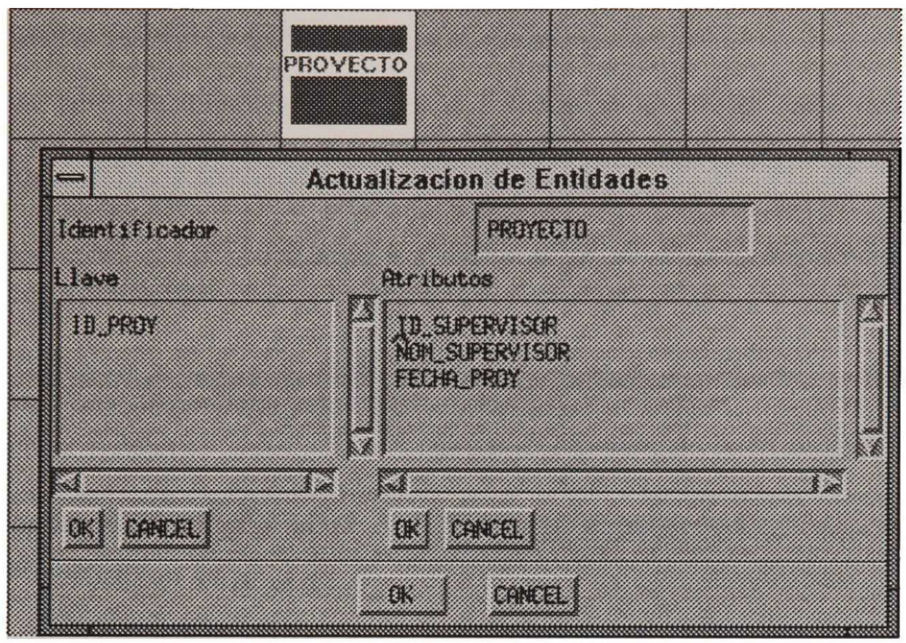
Estaríamos omitiendo las dependencias funcionales entre num_banco → nom_banco y num_banco → loc_banco.
El modelado correcto con eve estaría dado con una interrelación binaria (1:M) entre dos entidades Banco, Cuenta.



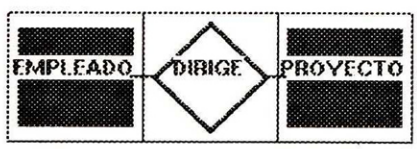
4.3.1.3. Tercera Forma Normal.

Una relación está en 3FN si además de estar en 2FN, ningún atributo no identificador único puede ser dependiente de otro atributo no identificador único. Otra definición dice que una relación está en 3FN si todos los atributos que no forman el identificador único dependen completamente del identificador único.

Si en un modelo con eve se diagrama una entidad Proyecto con identificador único id_proyecto y con atributos fecha_proy, id_supervisor, nom_supervisor.



Estaríamos omitiendo las dependencia funcional entre id_supervisor → nom_supervisor
El modelado correcto con eve estaría dado con una interrelación binaria Dirige (1:1) entre Empleado y Proyecto.



Los anteriores casos de modelado incorrecto, nos llevaron a la inclusión del proceso de normalización dentro de la metodología con eve.

4.4 Algoritmo de Bernstein.

4.4.1 Conceptos fundamentales.

En el juego de herramientas evex incluimos la automatización del proceso de normalización. Para entender dicho proceso, necesitamos conocer los algoritmos de normalización y conceptos de la teoría computacional que fundamentan los algoritmos.

Sean X y Y subconjuntos del conjunto de atributos A de un esquema de relación r.
Una dependencia funcional $X \rightarrow Y$ es una relación en donde el descriptor X determina o implica al descriptor Y.
Los atributos X del lado izquierdo se denominan implicantes o determinantes, y los atributos Y del lado derecho se denominan implicados o determinados.
Una dependencia funcional trivial $X \rightarrow Y$ es aquella dependencia en donde Y es un subconjunto de X.
Una dependencia funcional elemental $X \rightarrow Y$ es una dependencia no trivial en donde Y es un atributo único.

Los algoritmos de normalización requieren como entrada un conjunto de dependencias funcionales DF. Los atributos componentes de las dependencias funcionales forman el esquema relacional universal R.
El algoritmo de síntesis añade a las dependencias originales nuevas dependencias inferidas de la aplicación de los axiomas de Armstrong de reflexividad, aumentatividad, transitividad, proyectividad, aditividad y pseudotransitividad.

Los axiomas de Armstrong básicos son: reflexividad, aumentatividad, transitividad, etc.

Reflexividad.
Si $Y \subseteq X$ entonces $X \rightarrow Y$
Aumentatividad.
Si $X \rightarrow Y$ y $Z \subseteq W$, entonces $XW \rightarrow YZ$
Transitividad.
Si $X \rightarrow Y$ e $Y \rightarrow Z$, entonces $X \rightarrow Z$

y los agregados de

Proyectividad.
Si $X \rightarrow Y$, entonces $X \rightarrow Y'$ si $Y' \subset Y$
Aditividad.
Si $X \rightarrow Y$ y $X \rightarrow Z$ entonces $X \rightarrow YZ$
Pseudotransitividad.
Si $X \rightarrow Y$ e $YW \rightarrow Z$ entonces $XW \rightarrow Z$

Sea DF un conjunto de dependencias funcionales para el esquema de relación universal R y sea $X \rightarrow Y$ una dependencia funcional, decimos que DF lógicamente implica $X \rightarrow Y$, DF $X \rightarrow Y$, si para toda relación r de R que satisface la dependencia en DF también satisface $X \rightarrow Y$.
La cerradura DF^+ de un conjunto de dependencias funcionales DF, es el conjunto de dependencias funcionales lógicamente implicadas de DF.

$DF^+ = \{ X \rightarrow Y \mid DF \quad X \rightarrow Y \}$

Una dependencia funcional de se dice que es redundante del conjunto de dependencias funcionales DF si al eliminarla, del conjunto { DF - d } de dependencias funcionales, mediante la aplicación de los axiomas de Armstrong se puede derivar d.

Un atributo B perteneciente a un implicante X de la dependencia funcional $X \rightarrow Y$ se dice atributo extraño si la dependencia $(X-B) \rightarrow Y$ se encuentra en DF^+

La cubierta minimal de un conjunto de dependencias funcionales DF asociadas a un conjunto de atributos A es el subconjunto H de dependencias elementales del conjunto inicial DF, en donde ninguna de las dependencias funcionales contiene atributos extraños y los implicantes se componen de atributos simples.

Si al conjunto original DF de dependencias funcionales le aplicamos los axiomas de Armstrong obtenemos un DF^+. Si al nuevo conjunto le eliminamos las dependencias redundantes, los atributos extraños de los implicantes y reducimos los implicantes a atributos simples, obtenemos la cubierta minimal H del conjunto original DF.
El cálculo de la cubierta minimal H de DF es el primer proceso realizado en el algoritmo de Normalización por síntesis de Bernstein.

Para el cálculo de la cubierta minimal H, requerimos del cálculo del conjunto de todos los atributos derivados de X al aplicar los axiomas de Armstrong, denominado cálculo de la cerradura de un descriptor.

4.4.2 Cerradura de un descriptor de un conjunto de dependencias.

La cerradura de un descriptor X respecto a un conjunto de dependencias funcionales DF, se representa como X^+_{DF}.

Como una lectura complementaria a los conceptos por definidos, recomendamos la lectura de Ullman [Ullman88] y Chapa [Chapa90].

El algoritmo de la cerradura de un descriptor se presenta a continuación.

Algoritmo **Cerradura de un descriptor de un conjunto de dependencias.**

Entrada: **Un conjunto de dependencias DF.**
Un descriptor X.

Salida: **X^+_{DF}, cierre de X respecto a DF.**

Proceso:

$X^+ = X$
Repetir hasta que no se agreguen más atributos a X^+
Por cada dependencia $Y \rightarrow A$ en DF.
Si $Y \subseteq X$ y $A \notin X^+$
entonces $X^+ = X^+ \cup A$

4.4.3 Cálculo de la cubierta minimal de un conjunto de dependencias.

Con el cálculo del cierre de X respecto a DF, podemos entonces detallar el algoritmo de la cubierta minimal:

Algoritmo **Cálculo de Cubierta Minimal de un conjunto de Dependencias Funcionales**.

Entrada: **Un conjunto de dependencias funcionales DF.**
Salida: **H recubrimiento minimal de DF**.

Proceso Resumido:

1. Eliminación de atributos extraños
2. Eliminación de dependencias redundantes.

Proceso:
/* Eliminación de atributos extraños */
Repetir por cada dependencia $X \rightarrow B$ de DF
$Z = X$
Repetir por cada atributo A de X
Si $B \in (Z - A)^+$
entonces $Z = Z - A$
Reemplazar $X \rightarrow B$ por $Z \rightarrow B$

/* Eliminación de dependencias redundantes */
$H = DF$
Repetir por cada dependencia $X \rightarrow A$ de DF
$G = H - \{ X \rightarrow A \}$

Si A $\in X^{+}_{G}$ entonces H = G

El algoritmo de síntesis de Bernstein calcula como primer paso, la cubierta minimal H de las dependencias funcionales DF de entrada.
De ellas se agrupan las dependencias con implicantes iguales. Se unen los grupos con claves equivalentes Se eliminan las dependencias transitivas de cada grupo y construye de cada grupo de dependencias, un esquema de relación con claves candidatas. El conjunto de esquemas de relación cumplen con la tercera forma normal.

4.4.4 Normalización de Bernstein.

Algoritmo. **Normalización de Bernstein.**
Entrada: **DF conjunto de dependencias funcionales.**
Salida: **{Ai, Hi} conjunto de esquemas de relación en 3FN**.

Proceso Resumido:
1. Calcular la cubierta minimal H de DF.
2. Realizar una partición de H en grupos con el mismo implicante.
3. Mezclar claves equivalentes.
4. Eliminar dependencias transitivas.
5. Construir relaciones.

Proceso:
/* Cálculo de cubierta minimal H de DF */
/* Partición de H en grupos con el mismo implicante */
/* Mezcla de claves equivalentes */
Hacer J = $\varnothing$
Por cada par de grupos Hi, Hj con implicantes Xi, Xj
Si Xi está incluido en X^{+}j y Xj está incluido en X^{+} i
entonces se mezcla Hi y Hj.
añade a J las dependencias Xj $\rightarrow$ Xi y Xi $\rightarrow$ Xj
borrar de H todas las dependencias que cumplan
Xi $\rightarrow$ A si A $\in$ Xj
Xj $\rightarrow$ B si B $\in$ Xi.
/* Eliminación de dependencias transitivas */
Encontrar un conjunto minimal H' incluido en H tal que
$(H' \cup J)^{+} = (H \cup J)^{+}$
Incluyendo cada dependencia de J a su correspondiente grupo de dependencias del conjunto H
/* Construcción de Relaciones */
Por cada grupo construir un esquema de relación
Ri = <Ai, Hi>
donde Ai es el conjunto de todos los atributos que aparecen en Hi.

4.5. Algoritmo de Bernstein en EVEX.

4.5.1 Descripción.

El programa normalizador se visualiza en una forma o ventana de cuatro áreas, un área superior para Identificación del Archivo a normalizar, un área izquierda de Edición de Dependencias Funcionales, un área derecha de Despliegue de Resultado de la normalización y un área inferior de Mensajes.
El área de Edición de Dependencias consta de arriba hacia abajo de un área para Identificación de la relación actual, un área de Despliegue de Dependencias Funcionales en la cual se visualizan todas las dependencias funcionales de la relación actual, un área para Edición de Lado Izquierdo de una nueva dependencia funcional, un área para Edición del Lado Derecho de una nueva dependencia funcional, un área de botones de opciones etiquetados con Alta, Sig, Ant de las dependencias funcionales.
En el área de Lado Izquierdo y en el Lado Derecho de la dependencia funcional se escriben los nombres de atributos de la relación actual.
Las opciones Sig y Ant de las dependencias funcionales, sirven para recorrer las relaciones hacia la siguiente o anterior relación del archivo. La opción Alta añade al conjunto de dependencias de la relación, la dependencia funcional o dependencias funcionales escritas en el área de Edición del Lado izquierdo y Lado Derecho. Las dependencias funcionales dadas de alta se agregan al final de la lista del área de Despliegue de Dependencias Funcionales como Lado Izquierdo $\rightarrow$ Lado Derecho.
El área de Edición de Dependencias Funcionales oculta un menú que resurge con opciones de Recupera, Normalización Completa y Por pasadas y Salir.
El área de Despliegue del Resultado de la normalización oculta un menú que resurge con opciones de Graba, Graba Como y Cambia Nombre.

4.5.2. Procedimiento.

Existen dos formas de ejecutar el programa de normalización, en una su ejecución se realiza desde el sistema operativo unix tecleando **normaliza**, en la otra, su ejecución es con **evex**, y la opción Normaliza del menú principal
Si se ejecuta desde el sistema operativo debe abrir un archivo resultado de la traducción de un diagrama eve (extensión rel). Si lo ejecuta con evex, trata de abrir un archivo con el nombre del diagrama activo del diagramador eve con la extensión rel, si lo encuentra se visualizará en el área de Edición de Dependencias Funcionales, las dependencias funcionales de su primer esquema de relación.
Si no halla el archivo, con la opción de Recupera del menú oculto de Edición de Dependencias funcionales, lea un nuevo archivo para normalización. Las opciones de Normalización Completa y Por pasadas del mismo menú, aplican el algoritmo de 5 pasos de Bernstein. La aplicación del algoritmo Normalización Por pasadas va describiendo las acciones tomadas en el procedimiento para cada uno de los pasos del algoritmo y finalmente presenta los esquemas normalizados resultante. Cuando se aplica la Normalización Completa, solo muestra los esquemas resultantes.

La opción de Graba y Graba Como del menú oculto de Despliegue de Resultado, permite el almacena acenamiento de la información desplegada en el área de Despliegue de Resultado de la normalización. La normalización por pasadas se guarda con extensión nor, mientras que la normalización completa se guarda con extensión 3fn.
El proceso de normalización cuando detecta alguna violación a las formas normales para un esquema de relación, crea nuevas relaciones del esquema de relación original. Dándole a esos nuevos esquemas el nombre del esquema original concatenado a un guión bajo y la letra a, b, c, etc. dependiendo del número de esquema de relación creado.
La opción Cambia Nombre del menú oculto de Despliegue de Resultado, permite cambiar esos nombres a unos descriptivos.
El formato de los esquemas normalizados guardados con extensión 3fn es idéntico al formato de los esquemas relacionales con extensión rel.

En la metodología propuesta con **evex** el algoritmo de normalización se aplica al producto de la traducción de un esquema eve. De cada esquema de relación, se toma como implicante su identificador y los atributos descriptores como implicados.

4.5.3. Implementación.

4.5.3.1 Estructuras de Datos.

Los nombres de los esquemas de relación de entrada se guardan en una lista ligada de relaciones denominada Predicate_. La estructura mantiene los nombres de las nuevas relaciones sintetizadas.

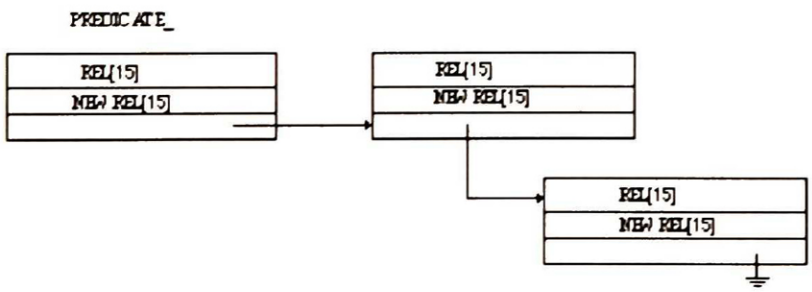


Figura. 4.2. Estructura Predicate_

La estructura schema va a guardar los esquemas de relación de salida, en una lista de esquemas relacionales en donde cada esquema tiene una lista de atributos, una lista de llaves, en donde cada llave se compone de una lista de atributos y una lista de descriptores.

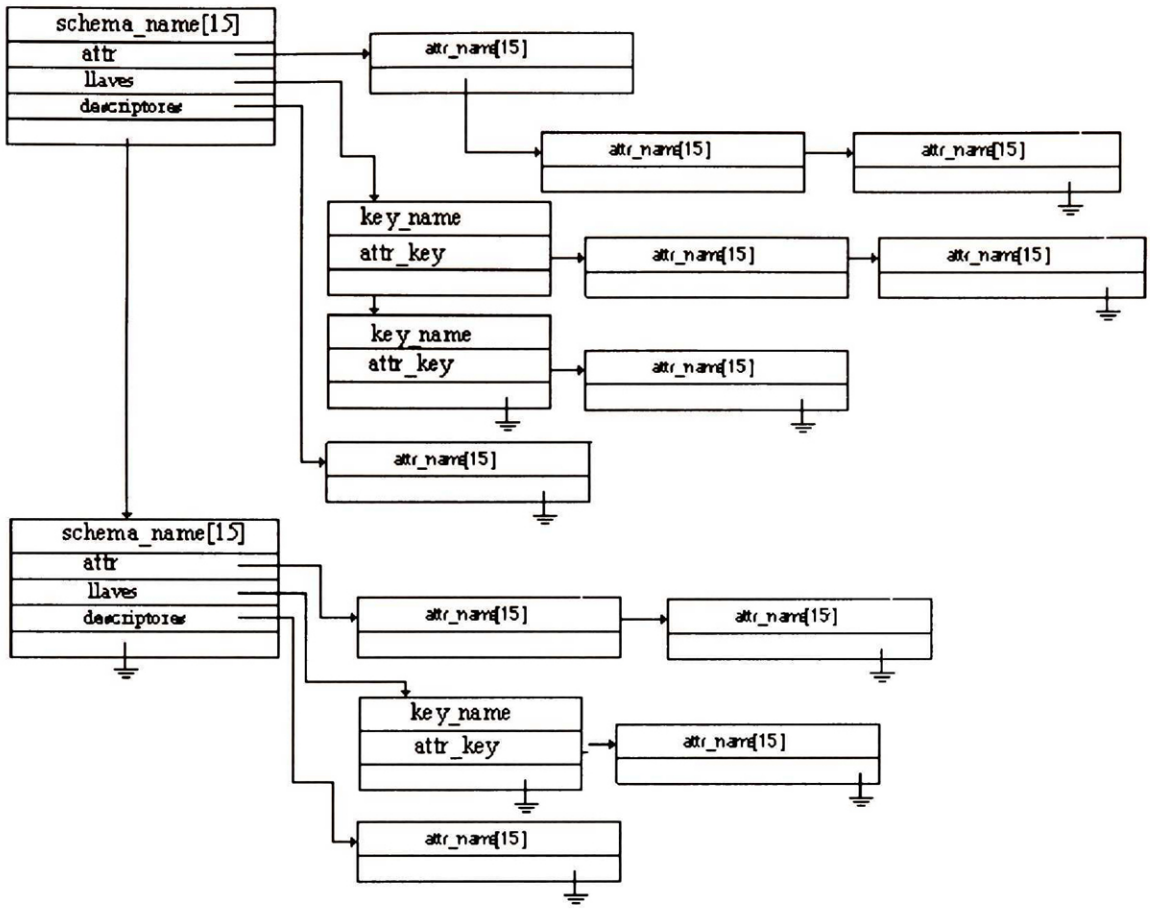


Figura 4.3 Estructura schema

La estructura EF_ se emplea para mantener un conjunto de dependencias funcionales.

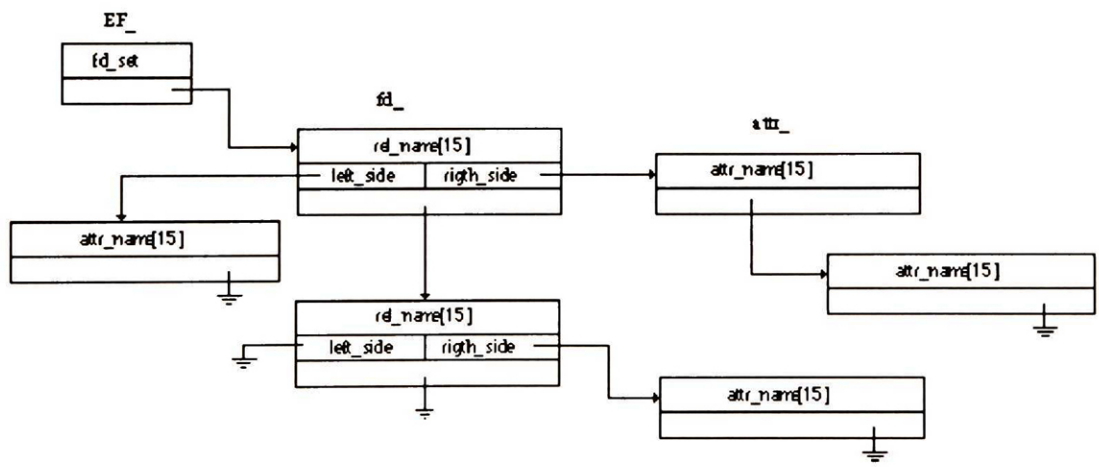


Figura 4.4. Estructura EF_

La estructura group_ se emplea para mantener una lista de grupos de dependencias del conjunto H.

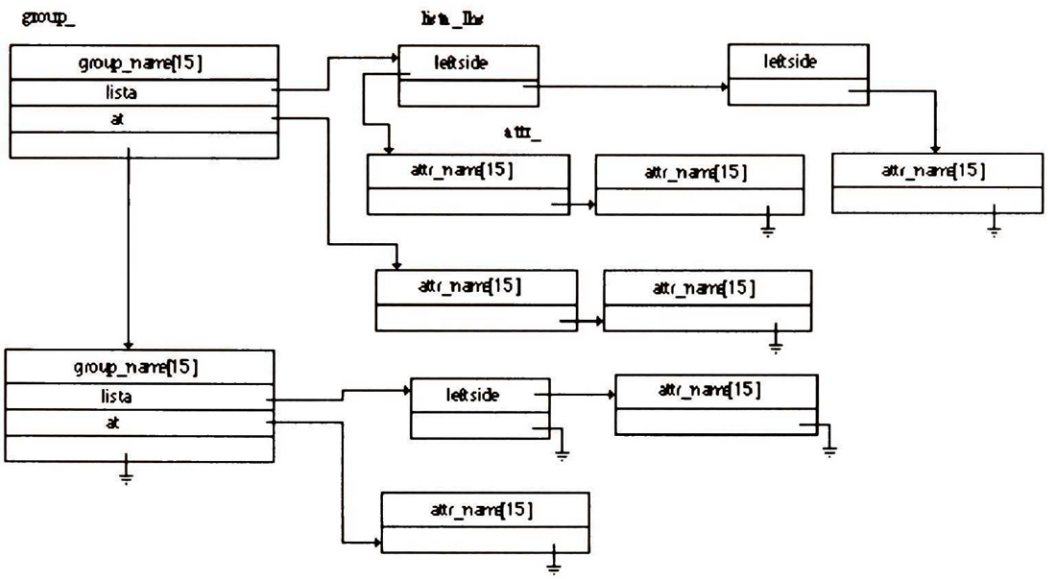


Figura 4.5 Estructura group_

La estructura closure_ se utiliza para mantener la cerradura de un conjunto de atributos X con respecto a los lados izq. de un conjunto de dependencias funcionales.

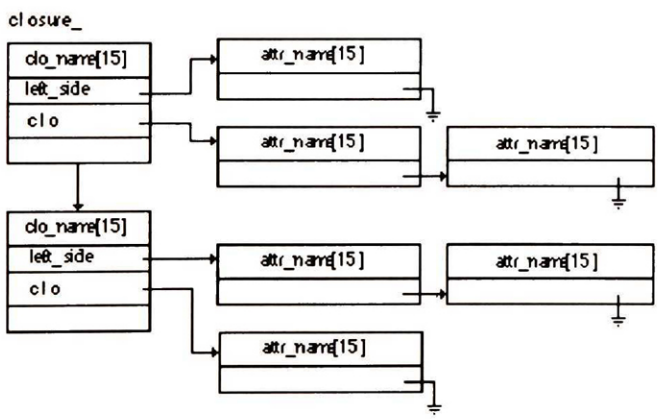


Figura 4.6 Estructura closure_

4.5.3.2 Algoritmo de la Interfaz.

Algoritmo. **Interfaz de Normalización.**

Crea forma principal.
    Crea forma 1.
        Crea Letrero "Archivo:"
        Crea Letrero Nombre Archivo con extension rel.
    Crea forma 2
        Crea Letrero "Relación"
        Crea Texto Nombre de relación
        Crea Texto con barra deslizante. Dependencias funcionales
        Crea Letrero "Lado izq."
        Crea Texto Lado izq. de Dependencia funcional.
        Crea Letrero "Lado der."
        Crea Texto Lado der. de Dependencia funcional.
        Crea Botón Alta.
        Llamado a función por evento. Agrega_fds.
        Crea Botón Siguiente.
        Llamado a función por evento. Esquema_siguiente
        Crea Botón Anterior.
        Llamado a función por evento. Esquema_previo.
    Crea forma 3.
        Crea Letrero "Normalización"
        Crea Texto con barra deslizante. Traduccion.
Crea Menú Oculto Dependencias Funcionales.
Llamado al Manejador de Eventos. Presión de Botón, Menú
    Crea Opción. Recupera.
    Llamado a función por evento. Recupera Relación.
    Crea Menú. Normalización.
        Crea Opción. Por pasadas.
        Llamado a función por evento. Normaliza, TRUE
        Crea Opción. Completa.
        Llamado a función por evento. Normaliza, FALSE.
    Crea Opción. Salir.
    Llamado a función por evento. Salir.
Crea Menú Oculto. Traducción.
Llamado al Manejador de Eventos.Presión de Botón, Menú.
    Crea Opción. Graba.
    Llamado a función por evento. Graba 3FN.
    Crea Opción. Graba como.
    Llamado a función por evento. Graba como 3FN.
    Crea Opción. Cambia_nombres.
    Llamado a función por evento. Cambia_nombres
Lee_relación. Archivo de Relación.

Activa ciclo de Eventos Principal.

La riqueza semántica y sencillez del modelo eve debe transferirse a un modelo lógico de base de datos relacional. La transferencia se logra con la conversión de los diagramas a esquemas relacionales mediante la aplicación de reglas de transformación sencillas.
Descubrimos en el capítulo, ejemplos en donde se hace necesaria la normalización para eliminar la redundancia de las relaciones resultantes, producto de diagramas mal elaborados. Revisamos por tanto el algoritmo de normalización por síntesis y su implementación.

DISEÑO FISICO DE LA BD

El proceso de diseño fisico de base datos se clasifica en dos, en el primero, a partir de un esquema lógico se construye una especificación del diseño fisico en un lenguaje de definición de datos de un sistema manejador en particular, en el segundo, el esquema fisico del lenguaje es compilado y traducido en estructuras de datos internas del SMBD particular.
Siguiendo la cronología de los procesos, este capítulo lo iniciamos explicando el módulo de construcción del lenguaje de definición de datos de evex. Continuamos con la descripción del compilador del lenguaje de definición de datos para el SMBD del proyecto CdataX.
El compilador del lenguaje de definición llamado compila_ldd, se encarga de crear una instancia de la base de datos particular con el nombre cap_nombrebase, instancia que a su vez emplearemos para verificar el comportamiento de la base datos diseñada, pues permite las operaciones de alta, baja, consulta para una tabla de la base de datos.
En el capítulo se detalla la arquitectura del CdataX, dado que su comprensión nos servirá para conocer las funciones aplicables a la instancia de la base de datos y para reconocer las estructuras de datos internas en las que se transforma el esquema lógico.

5.1 El Constructor del Lenguaje de Definición de Datos.

5.1.1. Descripción.

El Constructor del lenguaje de definición de datos tiene como propósito describir la definición de nuestra base de datos en un lenguaje entendible para el SMBD implementado. Para ello necesita unir los esquemas de relación normalizados de la base de datos con el o los diccionarios de atributos.
El Constructor consta de cuatro áreas un área superior para el nombre de archivo, un área izquierda para despliegue de los conjuntos de esquemas de relación, un área derecha para despliegue de la definición de la base de datos en el LDD y un área de mensajes.
El área izquierda tiene un menú oculto que hemos denominado de Recupera y Salir. El área derecha tiene un menú oculto de Graba y Graba como.
El programa constructor despliega también una ventana de diálogo compuesta de una etiqueta Diccionario, un área de texto con barra deslizante en donde se visualizan las rutas y nombres de los diccionarios seleccionados, y un arreglo de botones para opciones de busca, cancela, ok.

5.1.2 Procedimiento.

El programa constructor puede ejecutarse como una opción de **evex** o desde el sistema operativo con **constructor_ldd**.
En **evex** se ejecuta seleccionando la opción de Diseño Físico del menú de Normalización.

Cuando se ejecuta como **constructor_ldd** primero debe de seleccionarse el nombre del archivo que guarda el esquema de relación normalizado, con la opción Recupera del menú oculto Recupera y Salir.
Si se escogió un archivo normalizado correcto, se despliega el contenido del archivo en el área izquierda de la ventana principal. Y sobre ella se visualiza una ventana de diálogo para selección de los diccionarios de atributos.
Originalmente el programa busca por un archivo de diccionario con nombre igual al del archivo de esquemas normalizados pero con extensión dic. Si lo halla, escribe la ruta y el nombre en el área de texto. El usuario puede escoger más nombres de diccionario si oprime el botón busca de la ventana de diálogo.
El botón etiquetado como cancela permite borrar la selección de un diccionario. Cuando se presiona el botón ok, se leen los diccionarios y junto con el esquema normalizado se despliega el lenguaje de definición de datos en el área derecha del constructor.
En el lenguaje se toma como nombre de la base de datos el nombre del archivo de esquemas normalizados sin la extensión. El LDD se guarda tal como se visualiza en el área derecha con un nombre de archivo con extensión ddl -siglas en inglés de data definition language-, después de seleccionar la opción Graba o Graba como del menú oculto del área derecha.

Los archivos con extensión ddl siguen la gramática:

esquema_ddl→ esquema nom_bd NL diccionario (diccionario_atributos)+ findiccionario
(archivo nom_archivo NL (nom_atributo NL)+ finarchivo)+
(llave nom_archivo nom_atributo (, nom_atributo)*)*
(alias nom_alias, nom_atributo)* finesquema
esquema→ #esquema
diccionario→ #diccionario NL
findiccionario→ #fin diccionario
archivo→#archivo
finarchivo→ #fin archivo NL
llave→#llave
alias→#alias
finesquema →#fin esquema
nom_bd→id
nom_archivo→ id
nom_atributo→ id (Ref_G nom_archivo | Ref_E nom_archivo)
nom_alias → id
id → letra (dígito | letra | _) *
NL→' \n'

donde diccionario_atributos tiene la misma definición que diccionario_atributos de la gramática de salida del Editor del Diccionario de Atributos.

5.1.3 Implementación

Algoritmo. **Interfaz del Constructor del Lenguaje de Definición de Datos.**

```
Crea forma principal.
    Crea forma 1
        Letrero              "Archivo:"
        Letrero              Nombre de Archivo
    Crea forma 2.
        Crea forma 2.1
            Letrero          "3FN"
            Texto con barra deslizante        Relaciones en 3FN
        Crea forma 2.2
            Letrero "LDD"
            Texto con barra deslizante        Relaciones en LDD.
    Crea forma 3.
        Letrero              Errores
Crea Menú Oculto Izquierdo
Llamado al Manejador de Eventos. Presión de Botón, Menú
    Crea Opción  Recupera Relación
    Llamado a función por evento. Recupera Relación en 3FN
    Crea Opción  Salir
    Llamado a función por evento. Salir.
Crea Menú Oculto Derecho.
Llamado al Manejador de Eventos. Presión de Botón, Menú de Traducción.
    Crea Opción  Graba.
    Llamado a función por evento. Graba diccionario
    Crea Opción  Graba como.
    Llamado a función por evento. Graba como
Activa Ciclo de Eventos Principal.

Crea Forma de Diálogo. Diccionario.
    Crea  Forma 1.
        Letrero          "Diccionario"
        Texto con barra deslizante        Nombres de Diccionarios
    Crea Forma 2.
        Botón Busca.
        Llamado a función por evento. Recupera Diccionario.
        Botón Cancela.
        Llamado a función por evento. Cancela Nombre de Diccionario.
        Botón OK.
        Llamado a función por evento. Traduce Diccionario.
```

5.2 El Compilador del Lenguaje de Definición de Datos.

5.2.1 Descripción.

El diseño lógico de base de datos concluye en nuesto proyecto de tesis con la definición del esquema lógico de BD, generado automáticamente por el constructor del lenguaje de definición de datos.
Sin embargo, en el proyecto se decidió continuar con el diseño fisico de la base de datos. Las especificaciones de registros lógicos del LDD se transforman en estructuras de datos entendibles para un SMBD que las utiliza para el mapeo, resguardo y recuperación de registros fisicos.
El SMBD seleccionado para nuestro proyecto, le denominamos CdataX, y es un pequeño SMBD con funciones de resguardo y recuperación de datos interrelacionados usando un método de acceso de arboles B+.

Como un primer paso al diseño fisico se creó el programa **compila_ldd.** El cual permite la recuperación de un archivo con extensión ddl, y su visualización en una interfaz de usuario con una ventana, forma 1, para el diccionario de datos; una ventana, forma 2, con nombres y componentes de tablas; una ventana, forma 3, con nombres y componentes de índices y una ventana, forma 4, para describir los alias de los atributos.

El algoritmo para la interfaz de usuario se presenta a continuación.

Algoritmo. **Interfaz del Compilador de LDD**.

```
Crea forma principal.
        Crea forma 1.
                Crea Letrero "Diccionario"
                Crea forma con barra deslizante. 1.1
                        Crea Letrero "Atributo"
                        Crea Letrero "Tipo"
                        Crea Letrero "Longitud"
                        Crea Letrero "Mascara"
        Crea forma con barra deslizante. 2.
                Crea Letrero "Archivos"
        Crea forma con barra deslizante. 3.
                Crea Letrero "Llaves"
        Crea forma con barra deslizante. 4.
                Crea Letrero "Alias"
        Crea Menú Principal.
                Crea Menú Esquema.
                Crea Opción Recupera.
```

```
                Llamado a función por evento.Recupera_arc_LDD
                Crea Opción Compila
                Llamado  a función por evento. Compila.
                Crea Opción Salir.
                Llamado a función por evento. Salir.
Activa ciclo de Eventos Principal.

Función. Recupera_arc_LDD
Abre archivo arc_LDD
Lee linea_arc_LDD
Mientras linea_arc_LDD <> "#fin esquema"
        Lee linea_arc_LDD
fin Mientras
Crea forma FilaColumna de forma 1.1.
Desde i = 0 a ultimo_atributo_dic
        Crea forma
                Crea Letrero nom_atrib
                Crea Letrero tipo
                Crea Letrero longitud
                Crea Letrero mascara
                Crea Letrero etiqueta
fin Desde
Crea forma FilaColumna de forma 2.
Desde i=0 a ultimo_arc_dic
        Crea forma
                Crea Letrero nom_arch i
                Crea forma FilaColumna
                Mientras existan atributos_archivo
                        Crea Texto nom_atrib i
                fin Mientras
fin Desde

Crea forma FilaColumna de forma 3.
Desde i=0 a ultimo_arc_dic
        Crea Letrero nom_arch i
        Desde j=0 a ultimo_indice
                Crea forma FilaColumna
                Mientras existan atrib_en_indice
                        Crea Letrero nom_atrib de indice j  nom_arch i
                fin Mientras
        fin Desde
fin Desde
Crea Texto con barra deslizante de forma 4.
```

5.2.2 Procedimiento.

El programa **compila_ldd** se ejecuta desde el sistema operativo unix. Al ejecutarlo, se despliega la forma vacía, para llenar las diversas áreas de la forma, seleccione un archivo con extensión ddl, con la opción Recupera del menú Esquema. Después de ver las estructuras del ddl. Puede compilarlo con la opción Compila del menú Esquema. Al término de la compilación se despliega un mensaje indicando la finalización con éxito o errónea.
La compilación genera las estructuras de datos de la Base de Datos y las enlaza con las funciones y estructuras del CdataX para producir una instancia de actualización y recuperación de información de la base de datos.

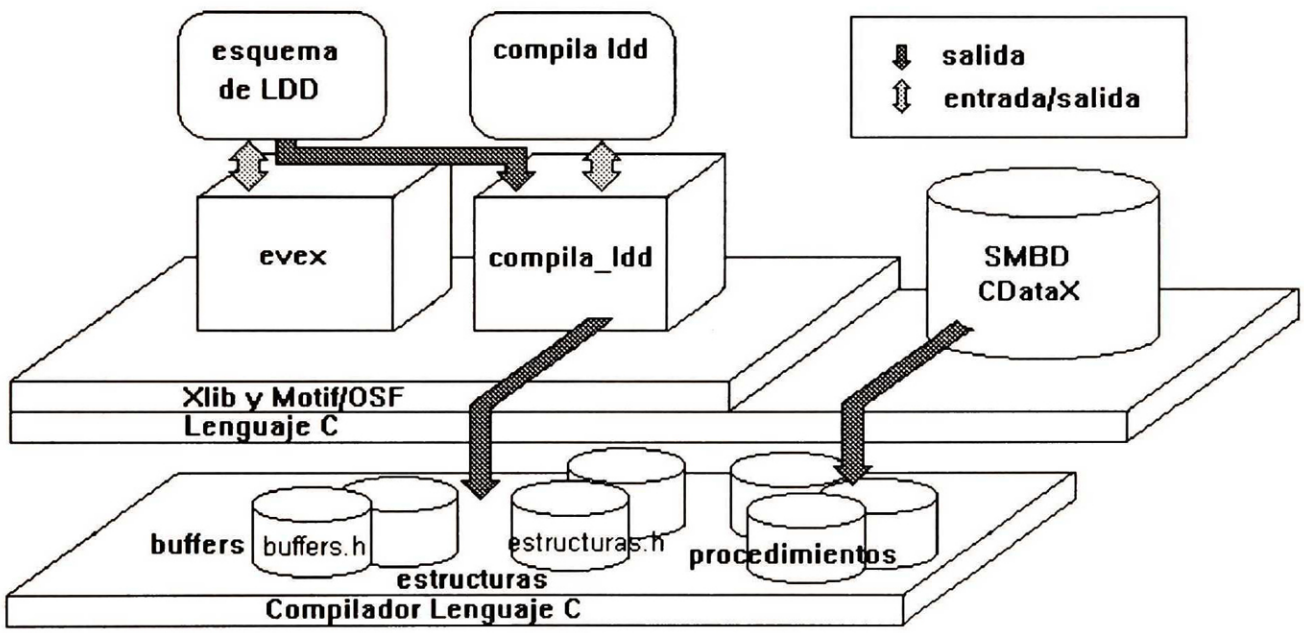


Figura 5.1 Compilador del Lenguaje de Definición de Datos.

5.2.3 Implementación.

5.2.3.1 Estructura de Datos para la Instancia de la BD.

La estructura de datos de la instancia se define en dos archivos include de C: buffers.h y estructuras.h, localizados en el mismo directorio que los programas del SMBD CdataX.

Los archivos buffers.h y estructuras.h se componen de directivas include en donde se proporciona al preprocesador de C, la ruta y nombre de los archivos de definición de estructuras de datos y buffers de la instancia de la BD particular.

Así estructuras.h contiene siempre una línea con la directiva: #include "/ruta/nombrebd.h".Y buffers.h contiene dos líneas con la directivas: #include "/ruta/nombrebd.1" y #include "/ruta/nombrebd.buff.h".

En buffers.h se describe la ruta y nombre del archivo del diccionario de datos (nombrebd.1) y de los buffers de memoria utilizados por las tablas de la BD (nombrebd.buff.h).

En estructuras.h se proporciona el nombre y ruta de la estructura de datos completa de la instancia contenida en el archivo nombrebd.h. El archivo nombrebd.h a su vez se describe con 3 líneas con las directivas: #include "/ruta/nombrebd.1", #include "/ruta/nombrebd.2" y #include "/ruta/nombrebd.3"
En donde el primer archivo include, nombrebd.1, define el diccionario de datos y elementos de las tablas, el segundo, nombrebd.2, define las estructuras de datos asociadas a los atributos de las tablas y el tercero, nombrebd.3 define las estructuras de datos asociadas a los índices de las tablas

El archivo nombrebd.1 contiene una serie de directivas #define, en donde cada atributo de la instancia de la BD se identifica en el define con una constante numérica, que va en orden creciente. También contiene declaraciones de estructura para cada tabla, con sus elementos.

```
#define ATRIBUTO_1              1
#define ATRIBUTO_2              2
#define ATRIBUTO_3              3
#define ATRIBUTO_4              4

struct file_1 {
                char atributo_1[30];
                char atributo_2[2];
        };

struct file_2 {
                char atributo_3[15];
                char atributo_4[20];
};
```

El archivo nombrebd.2 se compone de definiciones de estructuras de datos asociadas a las tablas. Y son un arreglo de apuntadores a caracteres para los nombres de atributos, un arreglo de caracteres para guardar el tipo de datos de cada atributo, un arreglo de apuntadores a caracteres para guardar la máscara de captura de cada atributo, un arreglo de apuntadores a caracter para almacenar las etiquetas de captura asociadas a los atributos, un arreglo de apuntadores a caracteres para almacenar los nombres de los archivos.

```
char *denames []= {
      "Atributo_1",
      "Atributo_2",
```

```
        "Atributo_3",
        "Atributo_4",
        (char *) 0
};

char eltype[]="ANAA";

char *elmask[]= {
    "----------------------------",
    "--",
    "---------------",
    "-------------------",
   (char *)  0
};

char *ellabel[] = {
     "Atributo No.1 : ",
     "Atributo No.2 : ",
     "Atributo No.3 : ",
     "Atributo No.4 : ",
    (char *)  0
};

char *dbfiles[] = {
        "FILE_1",
        "FILE_2",
       (char *)   0
};
```

El archivo nombrebd.3 se compone de un arreglo de enteros, en donde se almacena la longitud de cada atributo de la BD; un arreglo de enteros por tabla de la BD, en donde se almacena las constantes numéricas que identifican a los atributos pertenecientes a cada tabla; un arreglo de apuntadores a enteros para guardar las tablas que son de tipo apuntadores a entero; n-arreglos de enteros por tabla para almacenar las constantes numéricas que identifican a un índice, en donde n es el número de índices por tabla; un arreglo de apuntadores a entero por cada tabla, para identificar los índices por tabla; un arreglo de apuntadores a apuntadores de entero para almacenar las tablas con índice.

```
int ellen[] = { 30, 2 ,15, 20};

int f_file_1 [] = { ATRIBUTO_1, ATRIBUTO_2, 0};

int f_file_2 []  = { ATRIBUTO_3, ATRIBUTO_4, 0};
```

```
int *file_ele[] = { f_file_1, f_file_2 (int *) 0 };

int x1_file_1 [] = {
    ATRIBUTO_1,
        0
};

int x2_file_1[] = {
        ATRIBUTO_2,
    0
};

int * x_file_1 [] = {
    x1_file_1,
    x2_file_1,
     (int *) 0
};

int x1_file_2 [] = {
     ATRIBUTO_3,
     ATRIBUTO_4,
     0
};

int * x_file_2[] = {
      x1_file_2,
      (int *) 0
};

int ** index_ele [] = {
     x_file_1,
     x_file_2,
     (int **) 0
};
```

El archivo nombrebd.buff.h contiene como principales estructuras de datos, las áreas de memoria para cada tabla de la base de datos, y un arreglo de apuntadores a caracter conteniendo las direcciones de inicio de las áreas de memoria por tabla.

```
struct  file_1  buff_file_1;
struct  file_2  buff_file_2;

char * pstring [MXFILS] {
        (char *)        &buff_file_1;
        (char *)        &buff_file_2;
```

5.3. El SMBD CdataX.

Como dijimos en la introducción el SMBD construido se basa en el trabajo de tesis de maestría de Fernando Fiorentino "Modelo físico de un sistema de información geográfica para la exploración petrolera", en el cual se implementó un SMBD con método de acceso de arboles B en lenguaje C. Este proyecto elaborado para equipos personales, adopta ideas del SMBD Cdata de Al Stevens [Steven87].
El código del SMBD de Fiorentino se transportó de DOS a Unix, y al transportase tuvieron que modificarse las funciones de interfaz de modo gráfico de DOS a Xwindows. De este último sistema tomamos la X, para formar el nuevo nombre del SMBD, CdataX.

Las bases de datos implantadas con el CdataX usan para el resguardo de un registro lógico, dos registros físicos, uno para almacenar los datos en un archivo y otro para mejorar el método de acceso mediante un archivo de índices.en un archivo de índice.
Los archivos de datos se reconocen se almacenan en archivos con extensión DAT y los índices, en archivos con extensión X01, X02, X03, etc., en donde el índice primario tiene la numeración menor (01), el índice secundario el 02, y así sucesivamente [Fioren96].

Las funciones de administración de las estructuras del sistema manejador se encuentran implementadas en el archivo datafile.c, mientras que las de administración de los índices se implementan en el archivo btree.c
A un nivel de programación más alto, invocando las funciones de datafile.c y btree.c se encuentran las funciones de los archivos browse.c y database.c
El archivo browse.c define las funciones para solicitud de memoria de recuperación y estructura de datos, de administración de esa memoria y de preparación de las operaciones de recuperación y resguardo de datos.

5.3.1 Funciones de alto nivel.

A continuación presentamos el nombre y objetivo de las funciones iniciando con browse.c:

limpia_buffer.	Limpia el área de memoria de datos.
reg_de_bd.	Guarda en memoria un registro de un archivo.
reg_a_bd.	Escribe de la memoria a un registro de un archivo.
inicia_buffers.	Inicia dos áreas de memoria independientes, para contener registros de un archivo.
limpia.	Limpia las dos áreas de memoria de datos asignadas por inicia_buffers.
encuentra.	Encuentra un registro en una tabla de la base de datos.
browse.	Aplica la acción seleccionada (Alta, Baja...) a un archivo, empleando un área de memoria.

El archivo database.c implementa funciones relacionadas con los registros lógicos de una tabla, como:

db_open.	Abre la base de datos.
add_rcd.	Agrega un registro (datos e índice) a una tabla.
find_rcd.	Encuentra un registro (datos e índice) a una tabla.
verify_rcd.	Verifica que un registro este en un archivo.
first_rcd.	Encuentra el primer registro indexado de una tabla.
last_rcd.	Encuentra el último registro indexado de una tabla.
next_rcd.	Encuentra el siguiente registro indexado de una tabla.
prev_rcd.	Encuentra el registro previo indexado de una tabla.
rtn_rcd.	Regresa en un área de memoria el registro actual.
del_rcd.	Borra un registro (datos e índice) de una tabla.
curr_rcd	Encuentra el registro actual
seqrcd.	Regresa el siguiente registro en secuencia de una tabla.
db_cls.	Cierra la base de datos.
init_rcd.	Inicia un área de memoria para registro.
dberror.	Manejo de errores de la base de datos.
rcd_fill.	Mueve los datos de un registro a otro.
init_index	Inicia los índices para una tabla.

5.3.2. Funciones de bajo nivel.

El archivo datafile.c contiene funciones relacionadas con los datos de las tablas, como:

file_create.	Creación de archivo de datos.
file_open.	Apertura de archivo de datos.
new_record.	Crea un registro nuevo de datos.
delete_record.	Borrado de registro de datos.
get_record.	Recuperación de un registro de datos.
put_record.	Reescribe un registro.

El archivo btree.c implementa funciones de manipulación de árboles B, para acceso de las tablas, como:

btree_init	Inicia un árbol B.
btree_close.	Cierra un árbol B.
build_b.	Construye un árbol B.
locate.	Encuentra la localización de la llave.
deletekey.	Borrado de la llave.
insertkey.	Inserción de la llave.
implode.	Combina dos nodos hermanos.

Junto con las funciones, el archivo btree.c , define las estructuras de datos requeridas en los árboles B:
nodo del árbol, BTREE y encabezado del árbol, HEADER.

Si realizamos el seguimiento de una operación con una tabla de la base de datos podemos distinguir la forma de invocar las funciones y por tanto los niveles alto de las funciones browse.c y database.c y bajo de las funciones database y btree.
Tomemos como ejemplo una operación de búsqueda según el índice de una tabla de la BD, desde la función browse del archivo browse.c. Veremos que dentro del switch para operaciones con la BD, en el case de BUSCA, se invoca a las funciones encuentra y reg_de_bd implementadas dentro del mismo browse.c. Si continuamos con la función encuentra, notaremos que esta invoca a las funciones verify_rcd y find_rcd implementadas en el archivo database.c. Si continuamos rastreando ahora a la función find_rcd, nos daremos cuenta que esta invoca a su vez a la función locate implementada en el archivo btree.c, y que también invoca a getrcd implementada en database.c, y esta a su vez invoca a get_record implementada en datafile.c

5.4 Instancia de resguardo y recuperación de la base de datos.

Lo que hemos dado en llamar instancia de actualización y recuperación de la base de datos es un programa ejecutable que toma el nombre de **cap_nombrebd**.

La compilación del archivo con extensión ddl, también crea el archivo de recursos XCapBD en el directorio $HOME/app-defaults definiendo el recurso *cruta: /ruta
Por ejemplo si los datos se encontraran en $HOME/datos, en XCapBD debe grabarse
*cruta: $HOME/datos.

El programa cap_nombrebd se encuentra listo para crear los archivos de índice y de datos de la base de datos, así como los buffers para la unidad de memoria para iniciar. Para más tarde mediante ellas utilizar las tablas para resguardo y recuperación de datos.

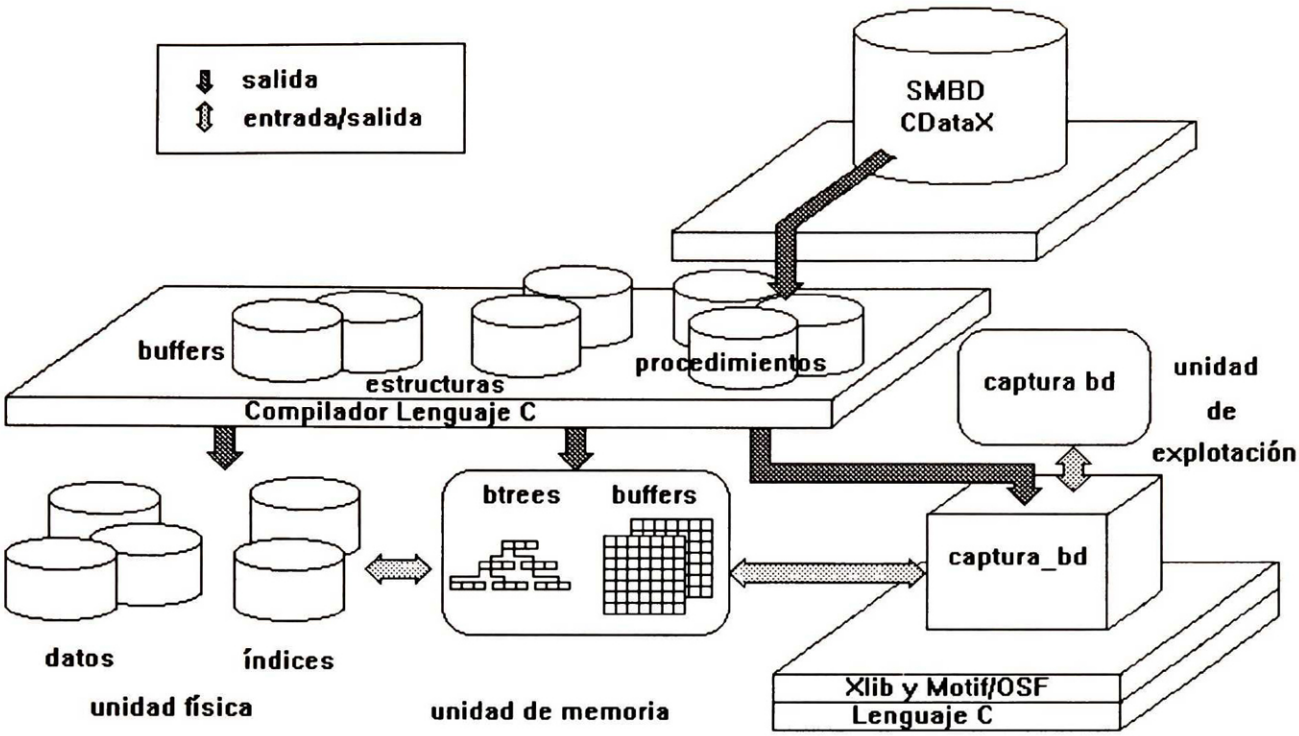


Figura. 5.2 Instancia de resguardo y recuperación.

5.4.1. Descripción.

La instancia de resguardo y recuperación despliega una forma que contiene todas las tablas de la base de datos ordenadas en 2 columnas. Cada tabla se representa con un símbolo de archivo y con el nombre de la tabla a su derecha.
La forma también tiene un menú de utilerías con opciones de Inicio, Indexa y Salir.
La instancia tiene dos formas adicionales, las cuales se activan por eventos.
Una de estas formas se emplea para reconstruir las tablas de índices. La forma se divide en dos áreas, superior e inferior. El área superior contiene un botón de opción bivaluada (activo/desactivo), con el nombre de la tabla asociada a su derecha, por cada tabla con índice. Los botones están dispuestos en 2 columnas. En su parte inferior, la forma tiene dos botones, uno etiquetado con OK y el otro con Cancela.
La otra forma relevante se utiliza para la actualización y recuperación de los datos de una tabla. Esta forma esta divida en tres áreas. En el área superior se localiza el botón de Salir. En su parte media se despliegan de arriba a abajo, las etiquetas de captura de cada atributo en la tabla y un área de captura de texto. En su parte inferior se despliegan 2 filas de botones de acciones aplicables a las tablas, etiquetados como, Alta, Baja, Busca, Limpia, Primero, Ultimo, Sigte, Previo.

5.4.2 Procedimiento.

Desde el sistema operativo mande ejecutar el programa **cap_nombrebd,** en donde el nombrebd depende de su instancia de base de datos. Si su instancia se llama inventario debe tener un programa ejecutable llamado **cap_inventario.**
El programa despliega la forma principal. Si es primera vez que ejecuta el programa, presione la opción Inicio del menú de Utilerías, para crear los archivos de datos y de índices de la instancia. Después de esta configuración, podrá efectuar cualquier operación de resguardo y recuperación con cualquier tabla.
Para seleccionar una tabla, presione el símbolo de archivo asociado a la tabla de su elección. Se despliega la forma de actualización y recuperación de la tabla.
Con ella podrá realizar las acciones de Alta, Baja, Busca, Primero, Ultimo, Sigte., Previo y Limpia.
Si desea efectuar la alta de un registro, debe teclear los datos del registro en el área de texto asociada a cada atributo, presione después el botón de Alta. Si la alta es aceptada se deben limpiar las áreas de texto de la forma y en la salida por omisión expresarse que el movimiento de alta ha sido correctamente ejecutado.
Si desea buscar un registro, debe introducir datos solamente en las áreas de texto asociadas a los atributos componentes del índice de la tabla, después presione el botón de Busca. Si se encuentra un registro con esos valores, en la forma se despliegan los valores de los demás atributos de la tabla. Si no se encontró un registro satisfaciendo la condición de búsqueda, en su salida por omisión se expresa que no se halló el registro.
Los botones de Primero y Ultimo también efectuan acciones relacionados con la recuperación de datos. Si desea buscar el primer registro de una tabla, siguiendo el orden de los índices, presione Primero. Si desea el último presione el botón de Ultimo. Su forma, debe rellenar las áreas de texto con el registro recuperado.
Las opciones de Baja, Sigte. y Previo se emplean sobre registros previamente desplegados.
Así si desea dar de baja un registro, primero debe de efectuar su búsqueda y cuando lo tenga desplegado en la forma utilice el botón etiquetado como Baja. Esta opción tiene una ventana de confirmación. Cuando una baja se ha dado correctamente, se limpia el área de texto de la forma y en la salida por omisión, se despliega el mensaje de que la baja se ha efectuado con éxito.
Si desea obtener un registro anterior en el orden de los índices al actuamente desplegado en la forma, presione el botón de Previo, el área de texto se rellena con los datos recuperados.

Si desea obtener el registro posterior al desplegado presione el botón de Sigte, el área de texto se rellena con los datos del registro siguiente. Si presiona Sigte. y el registro actual es el último en el orden de los índices, o si presiona Previo y el registro actual es el primero en el orden de los índices, se despliega el mensaje de que no exiten más registros en esa dirección.
El botón etiquetado como Limpia se emplea para limpiar todas las áreas de texto de la forma de una tabla.
Cabe hacer notar que durante una ejecución del programa de cap_nombrebd, puede abrir al mismo tiempo el número de tablas que desee, aunque no es recomendable abrir demasiadas.

5.4.3 Implementación.

El algoritmo para la Interfaz de Resguardo y Recuperación de la instancia y sus funciones se presenta a continuación.

Algoritmo. **Interfaz de Resguardo y Recuperación de la instancia.**
Crea forma principal.
Crea FilaColumna.
Mientras existan archivos
    Crea forma.
    Crea botón. Pixmap:archivo.
    Llamado a función por evento. Crea_dialogo_de_Captura_de_Archivos.
fin Mientras.
Crea Menú Principal.
    Crea Menú. Utilerias.
    Crea Opción. Inicia.
    Llamado a función por evento. Inicia.
    Crea Opción. Indexa.
    Llamado a función por evento. Archivos_a_indexar.
    Crea Opción.Salir.
    Llamado a función por evento. Salir.
Activa ciclo de Eventos Principal.

Función. **Crea_dialogo_de_Captura_de_Archivos**
Crea forma de diálogo.Crea_dialogo_de_Captura_de_Archivos
Define Arreglo de boton_accion { Alta, Baja, Busca, Limpia, Primero, Ultimo, Sigte, Previo }
Inicia_buffers
Abrir archivo $_k$
Crea forma FilaColumna 1
Crea botón Salir
Llamado a función por evento Salir, k
Crea forma FilaColumna 2
Desde i= 0 a ultimo_elemento
    Crea Letrero Etiqueta _de_elemento $_i$
    Crea Texto con longitud longitud $_i$
fin Desde
Crea forma FilaColumna 3
Desde i=0 a ultima_accion
    Crea botón accion $_i$
    Llamado a función por evento acciones, i
fin Desde

Función. **Archivos_a_indexar**.
Crea forma de Diálogo. Archivos_a_indexar.
Crea forma FilaColumna.
Crea Contenedor de Opciones bivaluadas.
Desde i=0 a num_arch
    Crea Opción bivaluada de nom_arch i
    Llamado a función por evento. Toggled, i
fin Desde

En la última fase del diseño de base de datos, conocida como de diseño fisico, el esquema lógico escrito en un lenguaje de definición de datos de un sistema manejador de base de datos se traduce a un esquema fisico del mismo manejador.
La traducción del esquema lógico al lenguaje de definición, se lleva a cabo en evex. El proceso de compilación del lenguaje se lleva a cabo en el compilador de CdataX, y en este último proceso se incorpora la instancia de la base de datos al SMBD. Con esta instancia podemos aplicar las operaciones fundamentales en las tablas para comprobar el funcionamiento de nuestro diseño de base de datos.

CASO DE ESTUDIO

En este capítulo se presenta la aplicación de la metodología de diseño de base de datos apoyada con la herramienta evex a un problema de administración de almacén.
Iniciamos el caso de estudio con una descripción del problema. La cual tomaremos como punto de partida para el diseño conceptual de base de datos. Primero, distinguimos los atributos, entidades e interrelaciones. Con el editor de diccionario de evex declaramos a todos los atributos reconocidos.
Volvemos a la descripción para identificar los roles y fusionarlos en un diagrama global.
Concluimos el diseño conceptual cuando realizamos el diagrama con el editor de entidad vínculo extendido.
Para el diseño lógico aplicamos el traductor de diagramas a esquemas de relación de evex y a los esquemas resultantes le efectuamos el proceso de normalización para transformarlos en esquemas relacionales en 3FN.
Finalmente construimos el lenguaje de definición de datos de la base de datos de almacén para el SMBD CdataX.

6.1 Descripción Preliminar.

En la fábrica X se producen 3 productos : A, B y C. Cada producto se elabora en una división, y cada una de estas tiene un jefe de división.
La fábrica cuenta con otros departamentos como son comercialización, compras y almacén general de refacciones.
El jefe del almacén general se encarga de despachar las solicitudes de refacciones de las divisiones y cuando no se tengan en existencia las refacciones, de solicitar los pedidos al departamento de compras.
Las refacciones tienen un identificador, una descripción, una marca, una ubicación por anaquel y un numero de unidades en existencia, un precio de compra, un valor mínimo de existencia.
Algunas refacciones, las de mayor uso, tienen refacciones sustitutas.
Los jefes de división al día levantan una solicitud de refacciones. En la solicitud se indica el número de solicitud, la fecha, el nombre de la división, el nombre del jefe, las refacciones y sus cantidades solicitadas.
Los empleados de almacén en el mismo día despachan las solicitudes de refacciones.
Con las refacciones despachadas entregan un reporte de despacho de la solicitud, en donde señalan si se ha despachado parcialmente o totalmente la solicitud, las piezas que se están despachando y la cantidad, el nombre del despachador, el nombre del que recoge las refacciones.
Como trabajo adicional deben descontar las existencias de las refacciones que despacharon.
Cuando el almacén no tenga una pieza original despacha una pieza sustituta. Cuando no exista la refacción sustituta, los almacenistas deben realizar una solicitud de pedido dirigido al departamento de compras. A este tipo de solicitud se le clasifica como urgente, existe otro tipo de solicitud de

pedido, clasificado como normal, en el que se solicitan refacciones que estén en o sobre el mínimo de existencias.
En las solicitudes de pedidos se indica el número de pedido, la fecha, el tipo de solicitud, las refacciones y cantidades solicitadas. Cuando la solicitud es urgente debe acompañarse con los nombres de las divisiones que solicitan la refacción.

Se plantea la automatización de un sistema de administración del almacén general, para lo cual requeriremos una base de datos que sirva de soporte.

6.2 Diseño Conceptual

6.2.1 Identificación de entidades, atributos e interrelaciones

Como primer paso del diseño conceptual, analizamos la descripción del problema y obtenemos las entidades, atributos e interrelaciones de la tabla

entidad	atributo	atributo	interrelación
división	nom_division	teléfono	administra
jefe_división	fax	nom_jefe	gestiona
solicitud	fecha_solicitud	num_solicitud	especifica
refacción	num_refacción	num_pedido	despacha
almacén	descripción	marca	surte
pedido	ubicación	unidades	requiere
urgente	max	min	hace
normal	promedio	fecha_pedido	
salida	almacenista	tipo_pedido	
	num_salida	fecha_salida	
	observaciones	cant_salida	
	cant_solicitud	cant_pedido	

6.2.2 Diccionario de atributos.

Como segundo paso, especificamos el tipo de datos y otras características de los atributos. Elaboramos el diccionario con el editor y lo nombramos como inventario.dic Presentamos a continuación su contenido.

```
NOM_DIVISION, A, 15, "_____", "Nom. Division"
TELEFONO, A, 12, "_____", "Telefono"
FAX, A, 12, "_____", "Fax"
NOM_JEFE, A, 30, "_____", "Jefe de Division"
FECHA_SOLICITUD, D, 6, "__/__/__", "Fecha Solicitud"
NUM_SOLICITUD, Z, 6, "_____", "Num. de Solicitud"
NUM_REFACCION, Z, 5, "_____", "Num. Refaccion"
```

```
NUM_PEDIDO, Z, 4, "____", "Num. Pedido"
DESCRIPCION, A, 56, "________________________________________________________", "Descripcion"
MARCA, A, 15, "_______________", "Marca"
UBICACION, A, 16, "________________", "Ubicacion"
UNIDADES, Z, 3, "___", "Unidades"
MAX, N, 3, "___", "Maximo"
MIN, N, 3, "___", "Minimo"
PROMEDIO, N, 3, "___", "Promedio"
FECHA_PEDIDO, D, 6, "__/__/__", "Fecha Pedido"
ALMACENISTA, A, 30, "______________________________", "Almacenista"
TIPO_PEDIDO, A, 1, "_", "Tipo de Pedido"
NUM_SALIDA, Z, 4, "____", "Num de Salida"
FECHA_SALIDA, D, 6, "__/__/__", "Fecha de Salida"
OBSERVACIONES, A, 256,
"
_____________________", "Observaciones"
CANT_SALIDA, N, 3, "___", "Cantidad"
CANT_SOLICITUD, N, 3, "___", "Cantidad"
CANT_PEDIDO, N, 3, "___", "Cantidad"
```

Cabe hacer notar que cada atributo en el archivo original se describe en una línea. En este documento se formatean los párrafos justificados a un tamaño determinado de ancho de hoja (105 columnas), por lo cual, atributos como el de OBSERVACIONES se describirá empleando más de una línea.

6.2.3 Diagrama EVE

Antecediendonos a la construcción del diagrama, asociamos a los atributos con las entidades e interrelaciones. Escogemos a los identificadores de las entidades e interrelaciones.
Con los atributos incluidos las entidades resultan:

División (nom_división, telefono, fax)
Jefe_división (nom_jefe)
Solicitud (num_solicitud, fecha)
Refaccion (num_refaccion, descripcion, marca, ubicacion, unidades, max, min)
Pedido (num_pedido, fecha, tipo_pedido)
Almacenista (nom_almacenista)
Salida (num_salida, fecha)
Urgente(nom_división)
Normal(observaciones)

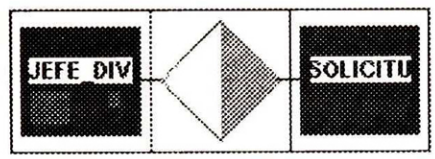
las interrelaciones:

Solic_Refaccion(cant_solicitud)
Refac_Pedido(cant_pedido)
Salida_Refaccion(cant_salida)

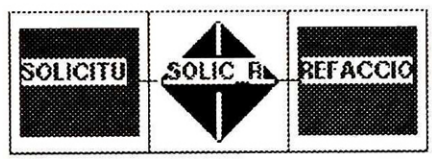
Procedemos a identificar los vínculos entre entidades e interrelaciones, expresados en roles:

Un Jefe de división Administra una División.

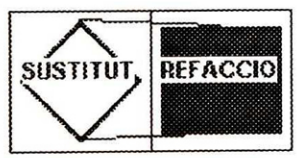
Un Jefe de división Gestiona más de una Solicitud.



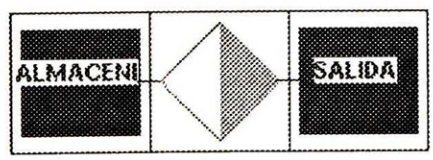
Una Solicitud Requiere más de una Refacción y una Refacción es requerida en más de una Solicitud.



Una Refacción Sustituye otra Refacción.

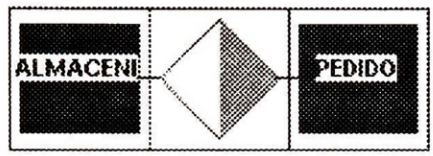


Un Almacenista Despacha más de una Salida.

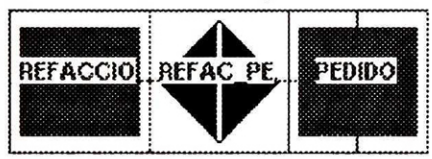


Una Salida Requiere más de una Refacción y una Refacción es requerida en más de una Salida.

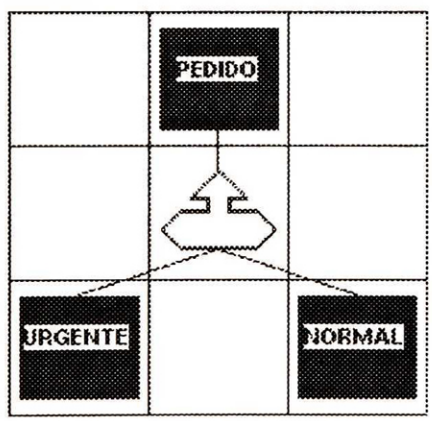
Un Almacenista Solicita más de un Pedido.



Un Pedido Requiere más de una Refacción, una Refacción es requerida en más de un Pedido.



Un Pedido es la generalización de un Pedido Urgente y un Pedido Normal.



Fusionamos los roles y elaboramos el diagrama:

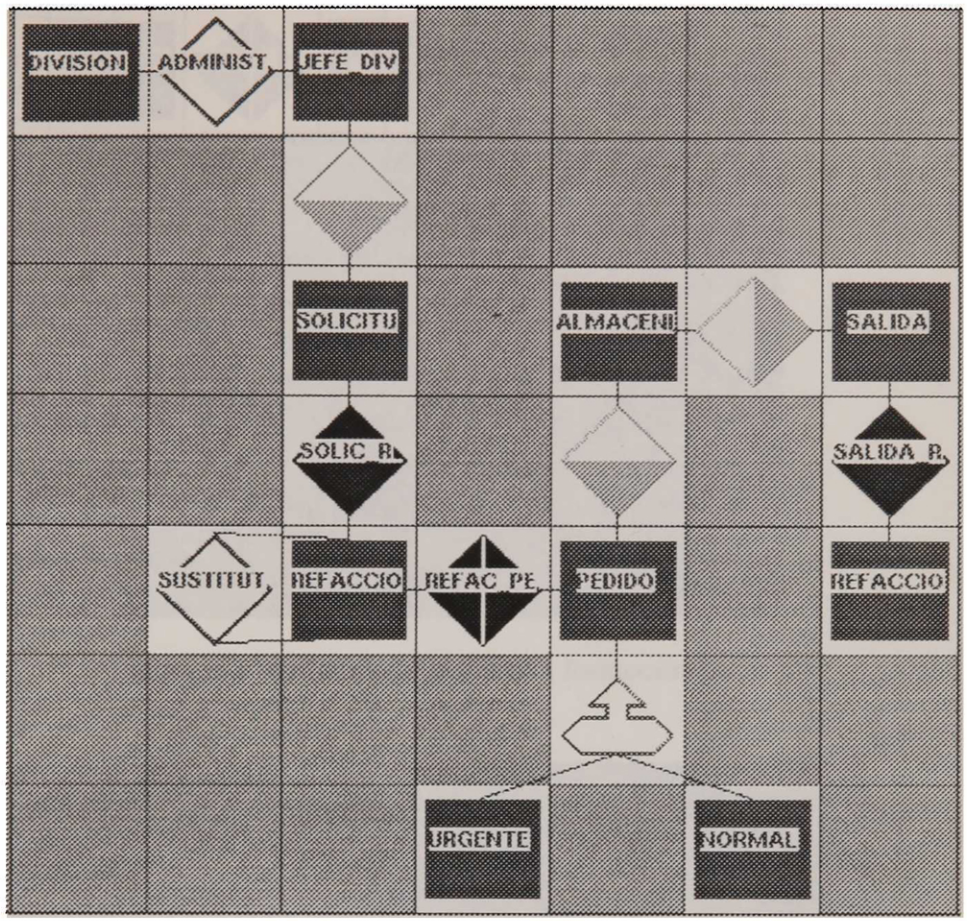


Figura 6.1 Diagrama eve para la administración del almacén

De cada entidad declaramos sus identificadores y atributos. Como ejemplo mostramos los identificadores y atributos de las entidades Refacción y Pedido.

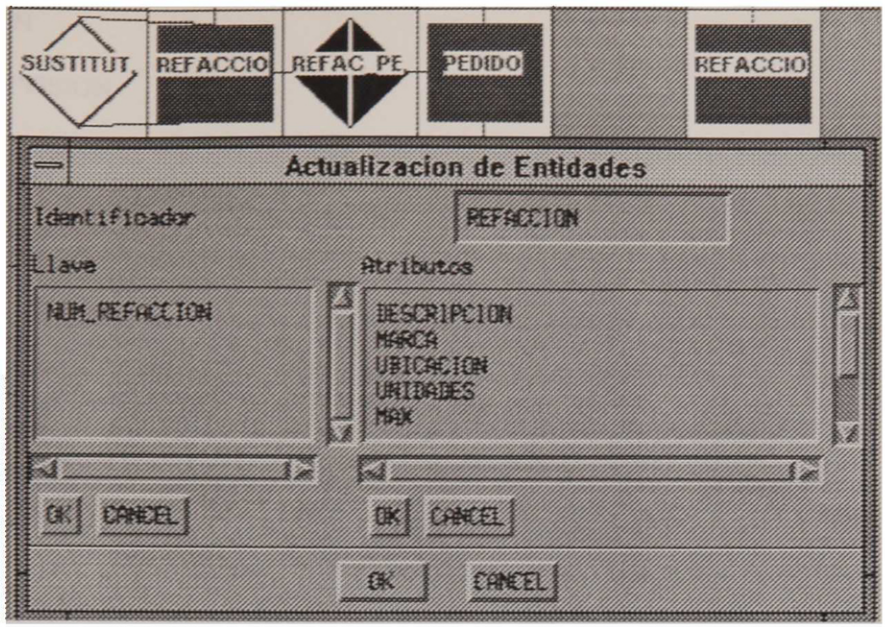


Figura 6.2 Atributos de la entidad Refacción.

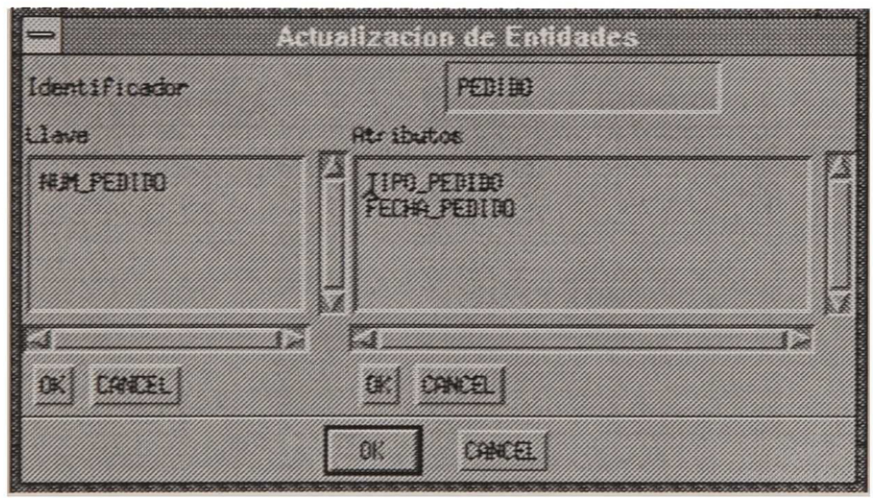


Figura 6.3 Atributos de la entidad Pedido.

6.3 Diseño lógico

6.3.1. Relaciones resultantes de la transformación del diagrama.

En la primera fase del diseño lógico el diagrama se traduce a esquemas de relación siguiendo las reglas de transformación para entidades e interrelaciones.
Las transformaciones de los esquemas originales se guardó en el archivo inventario.rel:
#name_rel

DIVISION
#keys
NOM_DIVISION
#endkeys
#desc
TELEFONO
FAX
NOM_JEFE
#enddesc

#name_rel
JEFE_DIV
#keys
NOM_JEFE
#endkeys
#desc
#enddesc

#name_rel
SOLICITUD
#keys
NUM_SOLICITUD
#endkeys
#desc
FECHA_SOLICITUD
NOM_JEFE
#enddesc

#name_rel
ALMACENISTA
#keys
ALMACENISTA
#endkeys
#desc
#enddesc

#name_rel
SALIDA
#keys
NUM_SALIDA
#endkeys
#desc
FECHA_SALIDA
ALMACENISTA
#enddesc

#name_rel
REFACCION
#keys
NUM_REFACCION
#endkeys
#desc
DESCRIPCION
MARCA

UBICACION
UNIDADES
MAX
MIN
PROMEDIO
NUM_REFACCION_01
#enddesc

#name_rel
PEDIDO
#keys
NUM_PEDIDO
#endkeys
#desc
TIPO_PEDIDO
FECHA_PEDIDO
ALMACENISTA
#enddesc

#name_rel
URGENTE
#keys
NUM_PEDIDO Ref_G PEDIDO
#endkeys
#desc
NOM_DIVISION
#enddesc

#name_rel
NORMAL
#keys
NUM_PEDIDO Ref_G PEDIDO
#endkeys
#desc
OBSERVACIONES
#enddesc

#name_rel
SOLIC_R
#keys
NUM_SOLICITUD
NUM_REFACCION
#endkeys
#desc
CANT_SOLICITUD
#enddesc

#name_rel
SALIDA_REF
#keys
NUM_SALIDA
NUM_REFACCION
#endkeys
#desc

CANT_SALIDA
#enddesc

#name_rel
REFAC_PEDIDO
#keys
NUM_REFACCION
NUM_PEDIDO
#endkeys
#desc
CANT_PEDIDO
#enddesc

#gene
NUM_REFACCION_01
#fuente
NUM_REFACCION

6.3.2 Relaciones resultantes de la normalización.

En la segunda fase del diseño conceptual se aplica la normalización por síntesis de Bernstein, para generar los esquemas en 3FN.
El algoritmo de normalización obtiene el conjunto de dependencias funcionales DF , de las asociaciones entre identificadores y atributos no identificadores de los esquemas de relación.
e.g. presentamos las dependencias vinculadas a los esquema de relación Refaccion y Pedido.

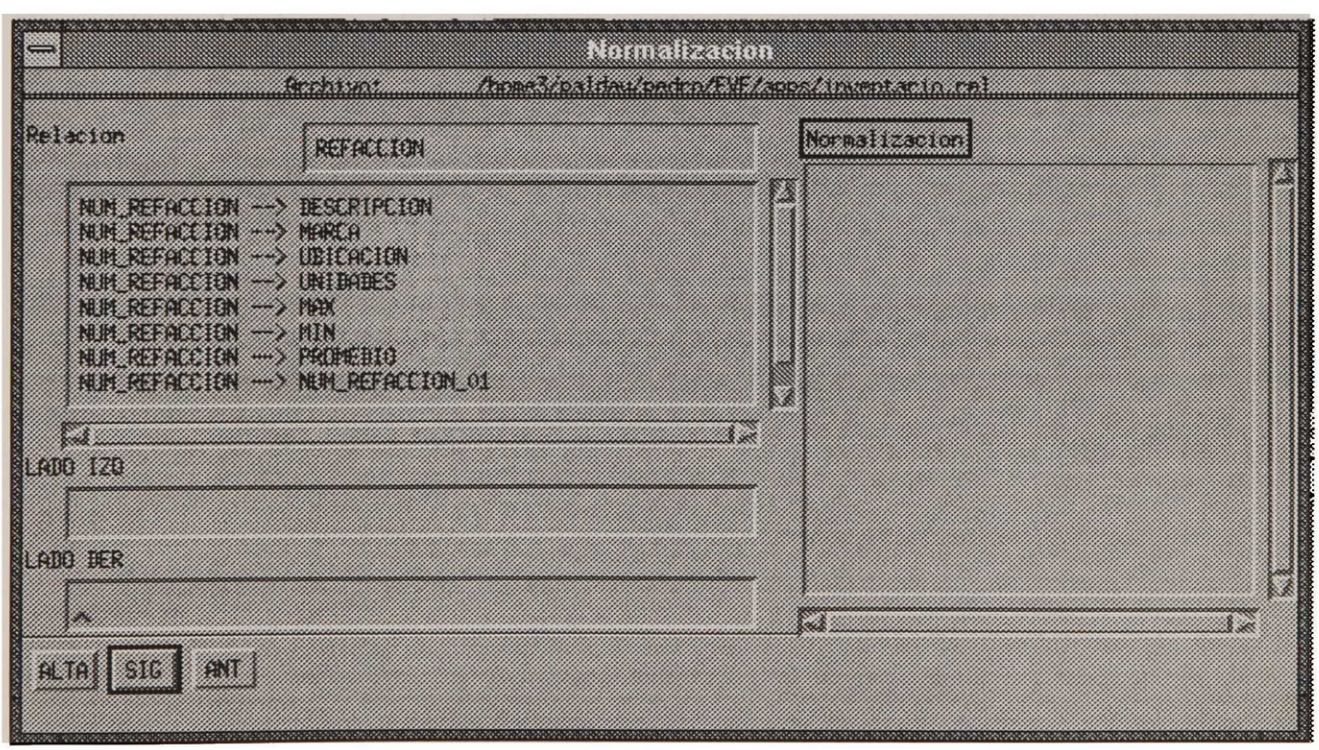


Figura 6.4 Dependencias funcionales de Refacción

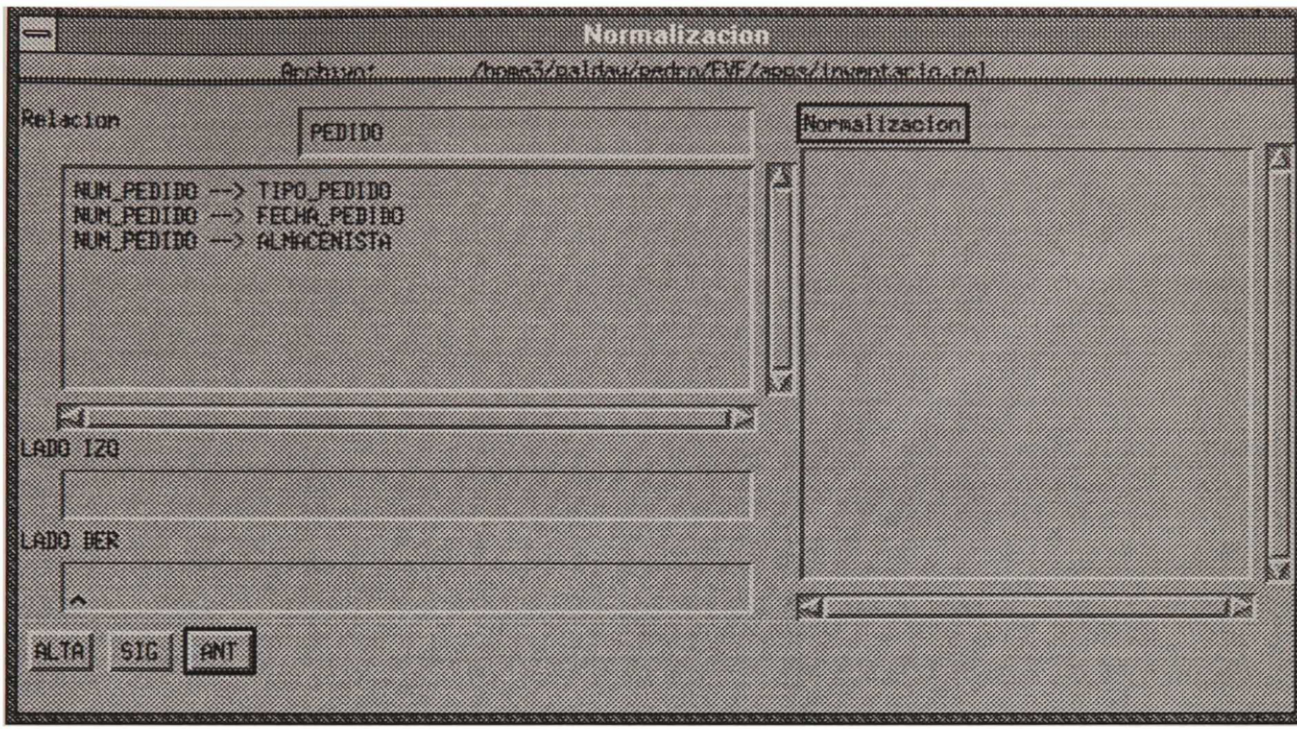


Figura 6.5 Dependencias funcionales de Pedido.

El modelo estaba diagramado en 3FN, como podemos apreciar si comparamos el siguiente listado con el contenido del archivo con extensión rel

#name_rel
DIVISION
#keys
NOM_DIVISION
#endkeys
#desc
TELEFONO
FAX
NOM_JEFE
#enddesc

#name_rel
JEFE_DIV
#keys
NOM_JEFE
#endkeys
#desc
#enddesc

#name_rel
SOLICITUD
#keys

NUM_SOLICITUD
#endkeys
#desc
FECHA_SOLICITUD
NOM_JEFE
#enddesc

#name_rel
ALMACENISTA
#keys
ALMACENISTA
#endkeys
#desc
#enddesc

#name_rel
SALIDA
#keys
NUM_SALIDA
#endkeys
#desc
FECHA_SALIDA
ALMACENISTA
#enddesc

#name_rel
REFACCION
#keys
NUM_REFACCION
#endkeys
#desc
DESCRIPCION
MARCA
UBICACION
UNIDADES
MAX
MIN
PROMEDIO
NUM_REFACCION_01
#enddesc

#name_rel
PEDIDO
#keys
NUM_PEDIDO
#endkeys
#desc
TIPO_PEDIDO
FECHA_PEDIDO
ALMACENISTA
#enddesc

#name_rel
URGENTE

#keys
NUM_PEDIDO Ref_G PEDIDO
#endkeys
#desc
NOM_DIVISION
#enddesc

#name_rel
NORMAL
#keys
NUM_PEDIDO Ref_G PEDIDO
#endkeys
#desc
OBSERVACIONES
#enddesc

#name_rel
SOLIC_R
#keys
NUM_SOLICITUD
NUM_REFACCION
#endkeys
#desc
CANT_SOLICITUD
#enddesc

#name_rel
SALIDA_REF
#keys
NUM_SALIDA
NUM_REFACCION
#endkeys
#desc
CANT_SALIDA
#enddesc

#name_rel
REFAC_PEDIDO
#keys
NUM_REFACCION
NUM_PEDIDO
#endkeys
#desc
CANT_PEDIDO
#enddesc

#gene
NUM_REFACCION_01
#fuente
NUM_REFACCION

6.4 Diseño físico

El ejemplo de administración de almacén en esta etapa produce una esquema entendible para CdataX, mediante el constructor del lenguaje de definición de datos.

6.4.1. Lenguaje de Definición de Datos resultante de aplicar el Constructor.

El contenido del archivo inventario.ldd , resultante de aplicar el constructor se presenta a continuación.

```
#esquema                INVENTARIO

#diccionario
        NOM_DIVISION, A, 15, "_______________", "Nom. Division"
        TELEFONO, A, 12, "____________", "Telefono"
        FAX, A, 12, "____________", "Fax"
        NOM_JEFE, A, 30, "______________________________", "Jefe de Division"
        FECHA_SOLICITUD, D, 6, "__/__/__", "Fecha Solicitud"
        NUM_SOLICITUD, Z, 6, "______", "Num. de Solicitud"
        NUM_REFACCION, Z, 5, "_____", "Num. Refaccion"
        NUM_PEDIDO, Z, 4, "____", "Num. Pedido"
        DESCRIPCION, A, 56, "_____________________________________________________" ,
"Descripcion"
        MARCA, A, 15, "_______________", "Marca"
        UBICACION, A, 16, "________________", "Ubicacion"
        UNIDADES, Z, 3, "___", "Unidades"
        MAX, N, 3, "___", "Maximo"
        MIN, N, 3, "___", "Minimo"
        PROMEDIO, N, 3, "___", "Promedio"
        FECHA_PEDIDO, D, 6, "__/__/__", "Fecha Pedido"
        ALMACENISTA, A, 30, "______________________________", "Almacenista"
        TIPO_PEDIDO, A, 1, "_", "Tipo de Pedido"
        NUM_SALIDA, Z, 4, "____", "Num de Salida"
        FECHA_SALIDA, D, 6, "__/__/__", "Fecha de Salida"
        OBSERVACIONES, A, 256,
"_____________________________________________________________________________
_____", "Observaciones"
        CANT_SALIDA, N, 3, "___", "Cantidad"
        CANT_SOLICITUD, N, 3, "___", "Cantidad"
        CANT_PEDIDO, N, 3, "___", "Cantidad"
#fin diccionario

#archivo        DIVISION
                NOM_DIVISION
                TELEFONO
                FAX
                NOM_JEFE
#fin archivo

#archivo        JEFE_DIV
```

```
                NOM_JEFE

#fin archivo

#archivo        SOLICITUD
                NUM_SOLICITUD
                FECHA_SOLICITUD
                NOM_JEFE
#fin archivo

#archivo        ALMACENISTA
                ALMACENISTA
#fin archivo

#archivo        SALIDA
                NUM_SALIDA
                FECHA_SALIDA
                ALMACENISTA
#fin archivo

#archivo        REFACCION
                NUM_REFACCION
                DESCRIPCION
                MARCA
                UBICACION
                UNIDADES
                MAX
                MIN
                PROMEDIO
                NUM_REFACCION_01
#fin archivo

#archivo        PEDIDO
                NUM_PEDIDO
                TIPO_PEDIDO
                FECHA_PEDIDO
                ALMACENISTA
#fin archivo

#archivo        URGENTE
                NUM_PEDIDO  Ref_G   PEDIDO
                NOM_DIVISION
#fin archivo

#archivo        NORMAL
                NUM_PEDIDO  Ref_G   PEDIDO
                OBSERVACIONES
#fin archivo

#archivo        SOLIC_R
                NUM_SOLICITUD
                NUM_REFACCION
                CANT_SOLICITUD
#fin archivo
```

```
#archivo        SALIDA_REF
                NUM_SALIDA
                NUM_REFACCION
                CANT_SALIDA
#fin archivo

#archivo        REFACCION_PEDIDO
                NUM_REFACCION
                NUM_PEDIDO
                CANT_PEDIDO
#fin archivo

#llave  DIVISION            NOM_DIVISION
#llave  JEFE_DIV            NOM_JEFE
#llave  SOLICITUD           NUM_SOLICITUD
#llave  ALMACENISTA                ALMACENISTA
#llave  SALIDA              NUM_SALIDA
#llave  REFACCION           NUM_REFACCION
#llave  PEDIDO              NUM_PEDIDO
#llave  URGENTE             NUM_PEDIDO Ref_G PEDIDO
#llave  NORMAL              NUM_PEDIDO Ref_G PEDIDO
#llave  SOLIC_R             NUM_SOLICITUD, NUM_REFACCION
#llave  SALIDA_REF          NUM_SALIDA, NUM_REFACCION
#llave  REFAC_PEDIDO        NUM_REFACCION, NUM_PEDIDO

#alias  NOM_REFACCION_01    NUM_REFACCION

#fin esquema  INVENTARIO
```

El diseño conceptual, lógico y fisico de base de datos con el modelo entidad vínculo extendido es ideal para cualquier problema de sistemas de información. En problemas de tamaño mediano, como el de administración de almacén es conveniente auxiliarse con evex.

Para el ejemplo, durante el diseño conceptual seguimos los pasos de identificación de entidades, atributos e interrelaciones, construcción del diccionario de atributos, identificación de roles y construcción del diagrama. De evex empleamos el editor de diccionarios y de diagramas.

En el diseño lógico transformamos el diagrama eve, en esquemas de relación, los cuales a su vez produjeron esquemas en 3FN, cuando le aplicamos el algoritmo de normalización. Finalmente obtenemos el esquema fisico para el manejador de base de datos CdataX, aplicando el constructor de lenguaje de definición de datos.

CONCLUSIONES

En el último capítulo examinamos las tendencias en herramientas automatizadas de diseño de base de datos basadas en el modelo eve. Las ventajas proporcionadas al modelo con la herramienta evex, sus deficiencias, los trabajos de desarrollo propuestos para aumentar su alcance y proyección de la herramienta automatizada. Estos puntos los hemos dividido según la tarea implicada en modelado de datos, traducción, normalización, diseño físico e implementación del SMBD CdataX.

EVEX es un editor y traductor de diagramas eve con un normalizador y un constructor de LDD. SMBD CdataX es un compilador del LDD y generador del programa manipulador de la BD particular cap_BD.
Ambos se desarrollaron para estaciones de trabajo DEC station 5000/25, DEC 5000/240 y DEC 5000/300, bajo XWindows y Motif para ULTRIX v. 4.2 1988-1992. Posteriormente la herramienta EVEX se dejó disponible para estaciones de trabajo SUN Sparc Classic 20.

En la representación de un modelo eve, generalmente sobrevienen dudas en el empleo de las conectividades de las interrelaciones, en el empleo de las interrelaciones jerárquicas y en la representación de roles aparentemente redundantes . Con el uso de la herramienta evex, y su capacidad de traducción al momento, los diseñadores pueden ir probando diferentes conceptualización del problema e identificar la representación más adecuada.

Por otra parte, con la ejecución repetida del Normalizador de evex, los diseñadores pueden reconocer que durante el proceso de normalización se considera no deseables para la implementación de BD a las dependencias transitivas e incompletas.

El resultado de su modelado lógico, puede compilarlo para más tarde probarlo en su instancia de resguardo y recuperación, operando con el SMBD CdataX.

El trabajo realizado requiere modificaciones en evex y en el SMBD CdataX para consolidarse como una herramienta robusta. Algunas de estas modificaciones abarcarían ambos sistemas.
Por ej. la herramienta no incluye elementos semánticos del modelo eve, como son el empleo de entidades débiles y de las interrelaciones opcionales, en buena parte porque el CdataX no posee rutinas para el control de las integridades referenciales.

Siguiendo una secuencia del modelo conceptual a la implementación física iremos comentando los puntos en los que se podría trabajar en evex, en el Compilador de LDD y en el CdataX.

7.1 Diseño Conceptual con EVEX.

El modelado con evex requiere de una salida impresa. Si se quisiera traducir a Postscript, un lenguaje vectorizado, sería conveniente vectorizar los iconos de las entidades e interrelaciones de EVE que actualmente se representan por mapas de bits y sus puntos de conexión.
La salida impresa debe contener anotaciones de los datos referentes al proyecto. Por lo cual hay que añadir su captura al crear cada diagrama e idear su forma de almacenamiento.
Los algoritmos para conectar y desconectar iconos del editor evex pueden mejorarse.
El editor de evex requiere de un indicador de la celda activa y del icono activo.
El editor pudiera ofrecer facilidades para mover y copiar celdas dentro de un mismo diagrama, como una primera fase. Una segunda fase, podría extender la facilidad para mover y copiar diagramas de ejecuciones distintas del programa, y mover y copiar elementos de diagramas de ejecuciones del programa en equipos remotos.

7.2 Diseño Lógico con EVEX.

7.2.1 Traducción a esquemas relacionales.

Se puede plantear un trabajo de traducción de eve a un lenguaje de definición de datos estándar como SQL2 norma ISO en 1992.

La gramática siguiente es un subconjunto de SQL2. Con ella podemos representar las traducciones
del modelo Entidad Vínculo Extendido.

< definición de tabla> ::= CREATE TABLE
< nombre de tabla >
< lista de elementos de tabla >
< lista de elementos de tabla > ::= < parent. izq. > < elemento de tabla >
{ < coma > < elemento de tabla > } *
< parent. der. >
< elemento de tabla > ::= < nombre columna>
< tipo de datos >
< definición de restricciones de columna > *
< definición de restricciones de columna> ::= < restricción de columna >
< restricción de columna > ::= NULL | NOT NULL
< definición de restricción de tabla > ::= < restricción de tabla >
< restricción de tabla > ::= <definición de restricción de unicidad >
| <definición de restricción referencial>
<especificación de unicidad> ::= UNIQUE| PRIMARY KEY
<definición de restricción referencial> ::= FOREIGN KEY
< parent. izq> < lista de columnas de referencia>
< parent. der>

< especificación de la referencia >
< especificación de la referencia > ::= REFERENCES < columna y tabla referenciadas >
[< acción referencial disparada >]
<columnas que ref.> ::= < lista de columnas de referencia >
<columnas y tabla referenciadas > ::= < nombre de tabla >
[<parent. izq> <lista de columnas de referencia> <parent. der.>]
< lista de columnas de referencia> ::= < lista de nombres de columnas>
< acción referencial disparada> ::= <regla de modificación> [<regla de borrado>]
| <regla de borrado> [<regla de modificación]
<regla de modificación> ::= ON UPDATE < acción referencial>
<regla de borrado> ::= ON DELETE <acción referencial>
< acción referencial> := CASCADE
<tipos de datos> ::= INTEGER| SMALLINT| CHARACTER < parent. iz> <longitud> <parent. der>
| NUMERIC | REAL | FLOAT | DOUBLE PRECISION

Como un ejemplo, una traducción de una interrelación uno a muchos entre Editorial y Libro, con nom_editorial siendo identificador de Editorial e id_libro como identificador de Libro.
Generaría la traducción a SQL2:

```
CREATE TABLE Editorial (nom_editorial  CHARACTER(15) NOT NULL UNIQUE,
                        PRIMARY KEY (nom_editorial))
CREATE TABLE Libro(id_libro CHARACTER(8) NOT NULL UNIQUE,
                   nom_editorial CHARACTER(15) NOT NULL,
                   PRIMARY KEY (id_libro),
                   FOREIGN KEY(nom_editorial)
                   REFERENCES Editorial
                   ON DELETE CASCADE
                   ON UPDATE CASCADE)
```

A pesar de haber sido reconocido como norma el SQL2, los Sistemas Manejadores de Bases de Datos (SMBD) comerciales siguen gramáticas ligeramente distintas.
Se propone la traducción de eve a un SQL específico de un Sistema Manejador de BD comercial. Por ej. Sybase version 4.9

Sybase puede leer un archivo consistente de commandos SQL al cual denomina script.
Cada comando de SQL, lo interpreta Sybase cuando encuentra la instrucción Go.
Se propone en una primera etapa, la traducción de un modelo eve a un script de Sybase versión 4.9 con la siguiente gramática reducida.

```
< definición de tabla> ::=      CREATE TABLE
                        < nombre de tabla>
                        < lista de elementos de tabla >
```

< lista de elementos de tabla > ::= < parent. izq. > < elemento de tabla>
{ < coma > < elemento de tabla > }*
< parent. der. >

< elemento de tabla > ::= < nombre columna>
< tipo de datos >
<restricción de columna>

< restricción de columna > ::= NULL | NOT NULL

< tipos de datos> ::= INT| SMALLINT| TINYINT| REAL|FLOAT
| MONEY | SMALLMONEY
| CHAR < parent. iz> <longitud> <parent.der>
| VARCHAR <parent. izq> < longitud> < parent. der>
| BINARY <parent. izq > < longitud> <parent. der>
| TEXT | IMAGE | BIT
| DATETIME | SMALLDATETIME

<definición índice> ::= CREATE < especificación de unicidad>
< especificación de agrupamiento de índice>
INDEX < nombre de índice>
ON < nombre de tabla>
< parent. izq>
< elemento de tabla>
{ <coma> <elemento de tabla> } *
< parent. der>

La traducción de la interrelación uno a muchos entre Editorial y Libro produciría un script:

```
CREATE TABLE Editorial (nom_editorial CHAR(15) NOT NULL)
Go

CREATE UNIQUE CLUSTERED INDEX K1_Editorial
        on Editorial(nom_editorial)
Go

CREATE TABLE Libro(id_libro CHAR(8) NOT NULL,
        nom_editorial CHAR(15) NOT NULL)
Go

CREATE UNIQUE CLUSTERED INDEX K1_Libro on Libro(id_libro)
Go
```

CREATE NONCLUSTERED INDEX K2_Libro on Libro(nom_editorial)
Go

La gramática reducida de Sybase no contiene restricciones referenciales.
Como una segunda etapa se propone la incorporación en los scripts de las restricciones referenciales.
Las integridades referenciales se definen mediante disparadores (triggers), activados en operaciones de borrado, modificación e inserción a tablas.
Los procedimientos disparados se construyen con comandos de lenguaje SQL de Sybase.
La gramática para disparadores es:

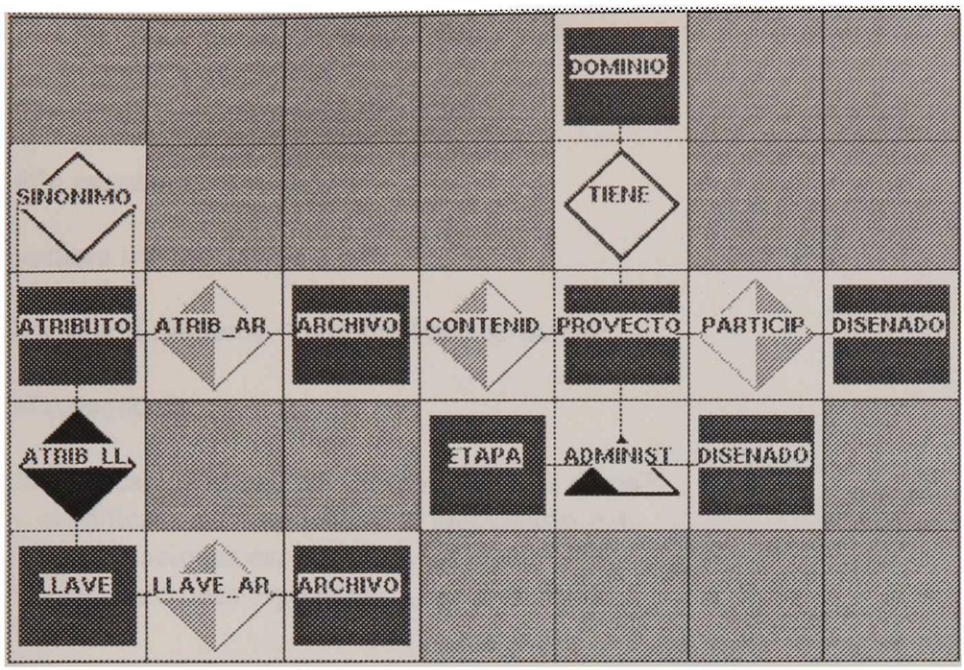
< definición de disparador > ::= CREATE TRIGGER < nombre disparador >
ON < nombre tabla >
FOR { INSERT | UPDATE | DELETE }
[< coma> { INSERT | UPDATE | DELETE}] *
AS
< definición de procedimiento con SQL >

7.2.2 Normalización.

En cuanto a los algoritmos de normalización, se han propuesto optimizaciones al algoritmo de síntesis de Bernstein por Diederich J y Milton J, en donde se emplea con menor frecuencia el cálculo de la cerradura de un descriptor con respecto a un conjunto de dependencias funcionales [DeMig93].

7.3 Diseño físico de BD.

Una carencia destacable de la BD basada en árboles B del proyecto, es que no crea un conjunto de tablas que funcionen como metadatos de la BD definida con el Lenguaje de Definición de Datos (LDD).
Con los metadatos se podrían tener facilidades para modificar la estructura de la BD, sin pérdida de la información y llevar la administración del proyecto de software.
Los archivos de los metadatos serían los únicos estáticos. La definición de interrelaciones de los metadatos la hemos modelado con eve para optimizar.
Se ha propuesto el siguiente diagrama. (metadatos.chn):



Con esquemas originales

Proyecto(id_proy)
Dominio(id_dominio, nom_dominio, num_caracteres)
Disenador(id_disenador, nom_disenador)
Administ(fecha_ini, fecha_fin)
Etapa(nom_etapa)
Archivo(id_archivo, nom_archivo,fecha_creacion)
Llave(nom_llave, fecha_creacion)
Atributo(id_atributo, nom_atributo, tipo_dato, longitud, letrero, fecha_creacion)

Al traducirse tendríamos

Proyecto(id_proy, fecha_proy, id_dominio)
Dominiodatos(id_dominio, nom_dominio)
Disenador(id_disenador, nom_disenador, id_proy)
Etapa(nom_etapa)
Archivo(id_archivo, nom_archivo, id_proy, fecha_creacion)
Llave(nom_llave, fecha_creacion, id_archivo)
Atributo(id_atributo, nom_atributo, tipo_dato, longitud, letrero, id_archivo, fecha_creacion)
Administ (id_proy, nom_etapa, id_diseñador, fecha_ini, fecha_fin)
Atrib_Ll(id_atributo, nom_llave)

La BD basada en árboles B carece de un módulo de procedimientos para administrar la integridad referencial en cascada, con default o de borrado. Con las funciones de acceso a los datos que proporciona la BD podría crearse dicho módulo.

El LDD debe modificar su gramática para aceptar procedimientos de modificación a una tabla.
La gramática que inicia:
#esquema < nombre base de datos>
...
#fin esquema

pudiera transformarse a una

#creacion esquema < nombre base de datos>
...
#fin esquema

#modificacion esquema <nombre base de datos>
< modificaciones específicas al diccionario > +
< modificaciones específicas a las llaves > *
< modificaciones específicas a los sinónimos> *

En las pantallas de captura de la tablas de la BD, se pudiera programar un objeto de interfaz, para desplegar los datos de cada tabla en forma de malla .

# Artículo	*Artículo*	*Unidades*	*Existencia*
001	Balero SKF 4'	pza	1
002	Balero SKF 6'	pza	2

[Batini94] Batini Carlo, Stefano Ceri, Navathe Shamkat. Diseño conceptual de Base de Datos. Un enfoque de entidad-interrelaciones. Addison Wesley/Diez Santos 1994.

[Ceri86] Ceri S, Gottlob G. Normalization on Relations and Prolog. Communications of the ACM 1986. Págs. 524-544.

[Chapa90] Chapa Vergara Sergio, Hernández Stefanoni R. Normalización de Base de Datos en C. Informe Técnico No. 107. Serie Amarilla. Octubre 1990. Publicaciones Técnicas del Depto. Ing. Eléctrica. CINVESTAV-IPN.

[Chapa91] Chapa Vergara Sergio, Hernández Stefanoni R. Programación Automática a partir de descriptores de flujo de información. Mexico, 1991. Tesis de Doctorado. CINVESTAV-IPN. Departamento de Ingeniería Eléctrica. Sección de Computación.

[Chapa93] Chapa Vergara Sergio, Alday Echavarría Pedro. Perspectivas en la automatización de las metodologías de Ingeniería de Software. Serie Amarilla. Publicaciones Técnicas del Depto. de Ingeniería Eléctrica. CINVESTAV - IPN. 1993.

[Chen76] Chen P. P The Entity-Relationship Model-Toward Unified View of Data. ACM Transactions on Database Systems. 1976

[DeMarc79] DeMarco Tom. Structured analysis and system specification. Foreword by P.J. Plauger. Englewood Cliffs. NJ. Prentice Hall 1979.

[DeMigu93] De Miguel Adoración, Piattini Mario. Concepción y Diseño de de Bases de Datos. Del modelo ER al modelo relacional. Addison Wesley Iberoamericana. 1993.

[Fioren96] Fiorentino Pérez Fernando. Modelo fisico de un sistema de información geográfica para la exploración petrolera. México, 1996. Tesis de Maestría. CINVESTAV-IPN. Departamento de Ingeniería Eléctrica. Sección de Computación.

[Hernan94] Hernández Stefanoni Raúl. Sistemas para el diseño de base de datos EVE. México, 1994. CINVESTAV-IPN. Tesis de Maestría. Departamento de Ingeniería Eléctrica. Sección de Computación.

[McClur93] McClure Carma. CASE la automatización del software. Addison Wesley Iberoamericana 1993.

[Marsha92] Marshall Brian. Motif Programming. The Esentials ... and more. Digital Press 1992.

[Nan92] Nan C. Shu Visual Programming. Van Nostrand Reinhold Computer Library, 1992.

[Sierra95] Sierra Romero Noé. HEVICOP herramienta visual para construcción de programas. México, 1995. Tesis de Maestría. CINVESTAV-IPN. Departamento de Ingeniería Eléctrica. Sección de Computación.

[Steven87] Stevens Al. C Database Development. Mis Press 1987.

[Teorey86] Teorey Toby, Yang Dongqing, Fry James. A logical Design Methodology of Relational Databases Using the Extended-Relationship Model. Computing Surveys of ACM. Vol 18 No. 2. June 1986.

[Teorey90] Teorey Toby J. Database Modeling and Design. The Entity-Relationship Approach. Morgan Kaufman Publishers Inc., 1990.

[Ullman88] Ullman Jeffrey D. Principles of Database and Knowledge-Base systems. Vol. 1: Classical Database Systems. Computer Science Presss 1988.

[Wirth73] Wirth Niklaus. Systematic Programming: an Introduction. Englewood Cliffs. NJ. Prentice Hall. 1973

[Young89] Young Doug. Xwindow System Programming. User Manual. 1989.

[Yourdo79] Yourdon Edward, Constantine Larry. Structured design: fundamentals of a discipline of computer program and system design. Englewood Cliffs, NJ. Prentice Hall 1979.

Los abajo firmantes, integrantes de jurado para el examen de grado que sustentará el **Ing. Pedro Enrique Alday Echavarría**, declaramos que hemos revisado la tesis titulada:

"DISEÑO DE BASE DE DATOS CON EVEX ENTIDAD VINCULO EXTENDIDO PARA XWINDOWS", consideramos que cumple con los requisitos para obtener el grado de Maestro en Ciencias, con especialidad en Ingeniería Eléctrica.

Atentamente

Dr. Sergio V. Chapa Vergara _____

Dr. Shiguang Ju _____

M. en C. Guillermo Rafael Domínguez _____

AUTOR ALDAY ECHAVARRIA, P. E.

TITULO DISEÑO DE BASE DE DATOS CON EVEX ENTIDAD VINCULO...

CLASIF. XM 97.4 RGTRO. BI 14966

NOMBRE DEL LECTOR	FECHA PREST.	FECHA DEVOL.
Amilcar Meneses Viveros	2-XII-98	9/12/98
Eduardo Villegas	14.12.98	5/1/99
Luis G. Reyes Velasco	17/FEB/99	19/2/99
Alejandro Amador G.	23/Feb/99	3/3/99
René Hernández Sánchez	13/04/99	29/04/99
Esteban E. Martínez Pelayo	28/06/99	6/7/99
Sergio V. Chapa	20/oct/99	24
Juan Carlos Medina Martínez	11/12/9	
Morales Moron Luz V.	20	
Morales Moron Luz Vi		
PEDRO DR		
Morales Moron		
Teresa V		
Ruben L		

.03/04