55

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ANÁLISIS Y DISEÑO DE UNA BASE DE DATOS DE IMÁGENES SATELITALES

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A N

JIMÉNEZ HERNÁNDEZ IRLANDA

REYES DELGADO HÉCTOR ENRIQUE

DIRECTOR:

M. en C. RANULFO RODRÍGUEZ SOBREYRA

MÉXICO, D. F. AGOSTO de 2002

Autorizo a la Dirección General de Bibliotecas de la UNAM a difundir en formato electrónico e impreso el contenido de mi trabajo recepcional.

NOMBRE: IRLANDA JIMÉNEZ HERNÁNDEZ

FECHA: 29/OCTUBRE/2002

FIRMA:

Autorizo a la Dirección General de Bibliotecas de la UNAM a difundir en formato electrónico e impreso el contenido de mi trabajo recepcional.

NOMBRE: HÉCTOR ENRIQUE REYES DELGADO

FECHA: 29/OCTUBRE/2002

FIRMA:

AGRADECIMIENTOS

Queremos agradecer al M. en C. Ranulfo Rodríguez Sobreyra, por el asesoramiento y la confianza brindada durante el desarrollo del presente trabajo. También agradecemos al Dr. Artemio Gallegos García por la paciencia y apoyo incondicional que tuvo con nosotros en todo momento...gracias por esperar!!!, y a todos los integrantes del Laboratorio de Oceanografía Física (LOF).

Un agradecimiento especial a la Ing. Gabriela Betzabé Lizarraga Ramírez por el gran apoyo en la revisión del Análisis y Diseño de la Base de Datos, así como por compartir un poco de sus conocimientos en el tema.

A la U.N.A.M., que nos da la oportunidad de adquirir ese conocimiento gratuito en esta preciosa institución, así como el de dar un gran paso en nuestras vidas para poder llevar muy en alto la formación académica que nos da.

Este trabajo fue realizado con el apoyo del Proyecto R32536-T de CONACYT , Acervo de Datos Radiométricos para el Estudio de Procesos Oceánicos y de la Climatología de la Temperatura de la Superficie de los Mares de México y Aguas Internacionales Adyacentes.

RESUMEN

En la presente tesis se plantea el análisis, diseño y desarrollo de una base de datos de imágenes satelitales con base en la información con la que se cuenta en el Laboratorio de Oceanografía Física (ICML-U.N.A.M.).

Debido al cúmulo de datos que se generan año con año, fue necesario automatizar esta información para tener un acceso rápido y eficiente de ésta. La información más importante se tiene en las imágenes satelitales que se procesan para obtener datos que se utilizan en los estudios de los procesos oceánicos.

De una investigación de los diversos modelos de datos y metodologías para la creación de bases de datos se eligió y se aplicó la metodología CASE, presentando paso a paso los resultados del desarrollo ya que proporciona un entorno de desarrollo eficiente y de fácil comprensión.

Junto con las características en el manejo de datos e imágenes, se utilizó la plataforma Oracle, en su versión 8i, para llevar a cabo el desarrollo e implantación de la base de datos.

ÍNDICE

CAPÍTULO 1. Introducción

1.1 Introducción General 1
1.2 Objetivos 1
1.3 Contenido 2

CAPÍTULO 2. Conceptos y Arquitectura de Bases de Datos

2.1 Introducción 3
2.2 Concepto de Base de Datos 4
2.3 Historia del Desarrollo de las Bases de Datos 6
2.4 Objetivos de una Base de Datos 7
2.5 Arquitectura de una Base de Datos 8
2.5.1 Nivel Externo 9
2.5.1.1 Componentes del Nivel Externo 10
2.5.2 Nivel Conceptual 10
2.5.2.1 Componentes del Nivel Conceptual 11
2.5.3 Nivel Interno 12
2.5.3.1 Componentes del Nivel Interno 12
2.6 Definición de un SMBD 13
2.7 Reseña Histórica de los SMBD 14
2.8 Objetivos de un SMBD 15
2.9 Arquitectura de un SMBD 16
2.9.1 Nivel Externo 17
2.9.2 Nivel Conceptual 18
2.9.3 Nivel Interno 18

CAPÍTULO 3. Modelos de Datos

3.1 Introducción 19
3.2 Conceptos de Modelos de Datos 19
3.3 Modelos de Datos 21
3.3.1 Modelos de Datos Lógicos Basados en Objetos 22
3.3.1.1 Modelo Binario 22
3.3.1.2 Modelo Infológico 25
3.3.1.2.1 Estructura 26
3.3.1.2.2 Restricciones 29
3.3.1.3 Modelo Funcional de Datos 31
3.3.1.4 Modelo Entidad/Relación 32
3.3.1.4.1 Estática del Modelo E/R 33
3.3.1.4.2 Tipos Especiales de Relación 35
3.3.1.4.3 Restricciones 37
3.3.1.4.4 Dinámica del Modelo E/R 37
3.3.1.4.5 Control de Redundancia en los Esquemas E/R 38
3.3.1.5 Modelo Orientado a Objetos 38
3.3.2 Modelos de Datos Lógicos Basados en Registros 42
3.3.2.1 Modelo Relacional 43
3.3.2.1.1 Estructura del Modelo Relacional 46
3.3.2.1.2 Dominio y Atributo 47
3.3.2.1.3 Relación 47
3.3.2.1.4 Claves 48
3.3.2.1.5 Restricciones 49
3.3.2.1.6 Restricciones Inherentes 49

3.3.2.1.7 Restricciones de Usuario..... 49
3.3.2.2 Modelo en Red..... 51
3.3.2.2.1 Presentación de un Modelo en Red General.... 53
3.3.2.3 Modelo Jerárquico..... 58
3.3.2.3.1 Características de la Estructura Jerárquica..... 59
3.3.2.3.2 Clasificación de los Árboles..... 60
3.3.2.3.3 Esquema y Ocurrencia de Árbol..... 62
3.3.2.3.4 Definición General del Modelo Jerárquico..... 63
3.3.2.3.5 Problemas del Modelo Jerárquico..... 63
3.3.2.3.6 La Función de Manipulación de Datos..... 66
3.3.3 Modelos Físicos de Datos..... 67
3.3.3.1 Modelo Semántico..... 67
3.3.3.1.1 Representación de los Modelos Semánticos.... 69

CAPÍTULO 4. Metodologías de Ingeniería de Software

4.1 Introducción..... 73
4.2 Metodología de EDWARD YOURDON..... 73
4.2.1 Estudio de Viabilidad..... 74
4.2.2 Análisis del Sistema..... 75
4.2.3 Diseño..... 75
4.2.4 Implementación o Producción..... 76
4.2.5 Pruebas y Test del Sistema..... 76
4.2.6 Control de Calidad..... 76
4.2.7 Documentación..... 77
4.2.8 Conversión de los Datos del Sistema Anterior..... 77
4.2.9 Instalación..... 77
4.3 Metodología MERISE..... 77
4.3.1 Etapas..... 78
4.3.2 Características Importantes de Merise..... 79
4.3.3 Etapas de Desarrollo..... 80
4.3.3.1 Etapa 1. Estudio Preliminar..... 80
4.3.3.2 Etapa 2. Estudio Detallado..... 82
4.3.3.3 Etapa 3. Realización..... 82
4.3.3.4 Etapa 4. Puesta en Marcha..... 83
4.3.4 Documentación..... 83
4.3.5 Evolución, Tendencias y Futuro del Método..... 84
4.4 Metodología SSADM..... 85
4.4.1 Fase 1. Estudio de Viabilidad..... 90
Etapa 01. Definición del Problema..... 90
Etapa 02. Identificación el Proyecto..... 92
4.4.2 Fase 2. Análisis..... 92
Etapa 01. Análisis de la Situación Actual..... 92
Etapa 02. Especificación de Requerimientos..... 93
Etapa 03. Selección de Opciones Técnicas..... 95
4.4.3 Fase 3. Diseño..... 97
Etapa 04. Diseño de Datos..... 97
Etapa 05. Diseño de Procesos..... 97
Etapa 06. Diseño Fisico de Datos y Procesos..... 99
4.4.4 Ventajas e Inconvenientes..... 102
4.5 Metodología MÉTRICA..... 103
4.5.1 Técnicas..... 104
4.5.2 Fases..... 106
4.5.3 Participantes y Funciones..... 107
4.5.4 Productos..... 108
4.5.5 Metas..... 108
4.6 Metodología JAMES MARTIN..... 109

4.6.1 Fases ... 111
4.6.2 Objetivos de las Fases ... 112
4.6.3 Desarrollo ... 112
4.6.4 Beneficios a Corto y Largo Plazo ... 113
4.6.5 Metas Alcanzadas ... 113
4.7 Metodología CASE ... 114
4.7.1 Etapas en la Metodología CASE ... 116
4.7.1.1 Estrategia ... 117
4.71.2 Análisis ... 117
4.7.1.3 Diseño ... 118
4.7.1.4 Construcción ... 119
4.7.1.5 Documentación ... 120
4.7.1.6 Transición ... 121
4.7.1.7 Producción ... 121
4.7.2 Bloques Básicos de CASE ... 122
4.7.3 Componentes Hardware de un Sistema CASE Básico ... 124
4.7.4 Herramientas CASE ... 125
4.7.5 Entornos CASE Integrados ... 127
4.7.6 La Arquitectura de Integración ... 128
4.7.7 Desarrollo de Bases de Datos con Herramientas CASE ... 129
4.7.7.1 Diseño Conceptual ... 130
4.7.7.2 Diseño Lógico ... 131
4.7.7.3 Diseño Físico ... 132

CAPÍTULO 5. Aplicación de la Metodología CASE

5.1 Introducción ... 133
5.2 Etapa de Estrategia ... 133
5.2.1 Antecedentes ... 133
5.2.2 Objetivo ... 134
5.2.3 Herramientas de Trabajo ... 134
5.2.4 Grupo de Trabajo ... 135
5.2.5 Condiciones Actuales ... 135
5.3 Etapa de Análisis ... 136
5.3.1 Modelo Funcional ... 137
5.3.2 Diagrama de Procesos ... 137
5.3.3 Diagrama de Flujo de Datos ... 142
5.3.4 Diagrama Jerárquico ... 143
5.3.5 Diagrama Entidad/Relación ... 145
5.4 Etapa de Diseño ... 146
5.4.1 Diseño de Red ... 146
5.4.2 Diseño de la Base de Datos ... 147
5.4.2.1 Mapeando Atributos a Columnas ... 148
5.4.2.2 Supertipos ... 149
5.4.2.3 Subtipos ... 149
5.5 Etapa de Construcción ... 150
5.5.1 Creación de Tablas ... 151
5.5.2 Creación de Llaves Primarias y Foráneas ... 152
5.5.3 Creación de Triggers ... 153
5.5.4 Creación de Constraints ... 154
5.5.5 Creación de Arcos ... 155
5.5.6 Creación de Procedimientos Almacenados ... 155
5.6 Etapa de Pruebas ... 157
5.6.1 Inserción de Datos ... 157
5.6.2 Consultas de Información ... 158

CAPÍTULO 6 Conclusiones 161

BIBLIOGRAFÍA 163

ANEXOS 165

Anexo A 167
Diagrama de Procesos
Anexo B 173
Diagrama de Flujo de Datos
Anexo C 177
Diagrama Jerárquico
Anexo D 181
Diagrama Entidad/Relación
Anexo E 185
Diseño de la Base de Datos
Anexo F 198
Código de la Base de Datos
Anexo G 223
Introducción a Oracle8i y El Tipo de Dato Blob

CAPÍTULO 1

INTRODUCCIÓN

1.1 INTRODUCCIÓN GENERAL

En los últimos años, las bases de datos han experimentado profundos cambios y no son ya, como ocurría hace algunos años, competencia exclusiva de grandes instalaciones con sistemas de información que manejen millones de registros; tampoco el diseño de bases de datos está reservado actualmente, como ocurría anteriormente, a uno pocos "gurus", especialistas en las técnicas de estructuración de los datos.

Por el contrario, las bases de datos se han extendido alcanzando a sistemas de tipo medio y pequeño, con moderadas, e incluso bajas cargas de trabajo, viéndose implicados en su diseño, administración y manejo muchos profesionales y multitud de usuarios que reclaman de ellas flexibilidad, sencillez y eficiencia. Es preciso, por tanto, una difusión de los conceptos básicos y de las técnicas relativas a la creación y utilización de las bases de datos que tengan en cuenta las premisas anteriores.

El propósito de esta tesis es plasmar el desarrollo del proyecto "Acervo de Imágenes Satelitales" el cual tiene por objetivo construir un banco de información de imágenes de la Temperatura Superficial del Mar (TSM) de los mares de México principalmente, bien organizado y de fácil acceso. Las características de este banco de información, una vez instalado, se ajusta a las necesidades de diversos usuarios: meteorólogos, oceanógrafos, climatólogos y otros especialistas investigadores en ciencias geofísicas y ambientales. Se requiere que lo puedan consultar personas involucradas en relación con este ámbito en general por medio de servicios en línea.

1.2 OBJETIVOS

El objetivo general del presente trabajo es aplicar una metodología para la implementación de una base de datos de imágenes satelitales. A partir del objetivo general se contemplan los siguientes objetivos particulares:

- Realizar un estudio de los conceptos fundamentales de las bases de datos, así como de los modelos de datos.
- Realizar un estudio de las metodologías de la Ingeniería de Software. Para seleccionar aquella que permita analizar, diseñar y construir una base de datos de manera eficiente, clara y sencilla.
- Desarrollar una base de datos de imágenes satelitales, empleando la metodología seleccionada.

1.3 CONTENIDO

La presente tesis, para cumplir con los objetivos planteados, se divide en capítulos. El contenido de los mismos se presenta a continuación:

Capítulo 1 *Introducción*

El capítulo presenta una introducción general del trabajo realizado, los objetivos y la descripción del contenido de cada capítulo para la presente tesis.

Capítulo 2 *Conceptos y Arquitectura de Bases de Datos.*

En este capítulo se presenta el concepto de base de datos, el desarrollo histórico de los sistemas de bases de datos, los objetivos a cumplir al crear una base de datos, los niveles que conforman la arquitectura de una base de datos. Se define el concepto de Sistema Manejador de Base de Datos (SMBD), los objetivos de un SMBD y la arquitectura.

Capítulo 3 *Modelos de Datos.*

El capítulo 3 aborda el concepto de modelo de datos, ofreciendo un marco de los modelos al establecer los tipos de modelos de datos. Iniciando con el análisis del modelo binario, siguiendo con el modelo infológico, su estructura y restricciones. Se incluyen el modelo funcional, el modelo Entidad/Relación y las características esenciales de este último. Se presenta el modelo orientado a objetos, el modelo relacional, en el se describen también las características del modelo de red y el modelo jerárquico. Finalmente se mencionan los que componen los modelos físicos de datos, entre los que destaca el modelo semántico.

Capítulo 4 *Metodologías de Ingeniería de Software.*

Este capitulo presenta información sobre las metodologías y técnicas más empleadas en cada una de las fases del desarrollo de aplicaciones. Se describen las metodologías de EDWARD YOURDON, MERISSE, SSADM, JAMES MARTIN y CASE. Esta última metodología, se aborda detalladamente porque es fundamental en el desarrollo de la presente tesis.

Capítulo 5 *Aplicación de la Metodología CASE*

Este capítulo presenta la descripción de cómo se aplicaron las etapas que conforman la metodología CASE en el desarrollo de la Base de Datos de Imágenes Satelitales. A partir de la estrategia, el análisis realizado, así como los diferentes modelos y diagramas obtenidos en esta etapa. Después se describen los métodos y técnicas empleadas para la elaboración del diseño de la Base de Datos y posteriormente llevar realizar la construcción. También, como parte de la metodología CASE, se practicaron diversas pruebas, para comprobar que los objetivos trazados al inicio del desarrollo, fueron alcanzados y que los requerimientos detectados hayan sido completamente cumplidos.

Capítulo 6 *Conclusiones*

CAPÍTULO 2

CONCEPTOS Y ARQUITECTURA DE BASES DE DATOS

2.1 INTRODUCCIÓN

El procesamiento de la información es esencial para la administración de los gobiernos, los negocios y la educación. El pronostico del tiempo, por ejemplo, se basa en el procesado de información precisa. En la sociedad es vital para una organización o empresa proporcionar información correcta y puntual, para apoyar la toma de decisiones y otras actividades gerenciales.

Como resultado del crecimiento económico y avances tecnológicos, muchas organizaciones han crecido tanto en el tamaño como en la sofisticación de sus funciones administrativas. Mientras el volumen de procesamiento de datos crece a una rapidez sin precedentes, también crece la demanda de medios eficientes para manejarlos.

La invención de computadoras revolucionó los medios tradicionales del procesamiento. A principios de los años 60's, varias firmas comerciales empezaron a computarizar sus sistemas de información; los datos se guardaban en medios electrónicos en lugar de guardarlos en papel, y se usaban lenguajes de alto nivel para recuperar y manejar los datos en aditamentos de almacenamiento.

En los sistemas de información convencionales, las aplicaciones individuales se desarrollaban independientemente y cada programa de aplicación presentaba sus propios archivos privados. Al final de los años 60's surgió el sistema de bases de datos para superar los problemas asociados con los sistemas de información tradicionales [Tsai, 1990].

Las exigencias de los usuarios, respecto a sistemas de información más flexibles y eficientes. A obligado a dedicar una mayor atención a los datos y a su estructuración, buscándose una gestión más racional de la información en su conjunto, por lo cual ha pasado a ser considerada como un recurso fundamental de la organización.

A medida que los diseñadores de sistemas de información, se van convenciendo de la trascendencia que la gestión racional de los datos, para conseguir un desarrollo coherente y eficaz de estos sistemas [De Miguel, 1999].

2.2 CONCEPTO DE BASE DE DATOS

Son muy numerosas las definiciones de base de datos, y si se analizan detenidamente se suele observar en casi todas ellas coincidencias en ciertos elementos; aunque también se detecta la falta de otros conceptos, que son característicos de las bases de datos y que marcan la diferencia entre este concepto y el de ficheros. En la figura 2.1 se reproducen distintas definiciones de base de datos.

- "Colección de datos interrelacionados, almacenados en conjuntos sin redundancias perjudiciales o innecesarias; su finalidad es servir a una aplicación o más, de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados".
[Martin, 1975] .

- "Colección o depósito de datos, donde los datos estan lógicamente relacionados entre sí, tienen una definición y descripción comunes y estan estructurados de una forma particular. Una base de datos es también un modelo del mundo real y; como tal, debe poder servir para toda una gama de usos y aplicaciones".
[*Conference des Statisticiens Europeens, 1977*].

- "Conjunto de datos de la empresa memorizado en un ordenador, que es utilizado por numerosas personas y cuya organización está regida por un modelo de datos".
[Flory, 1982].

- "Conjunto estructurado de datos registrados en computadoras para satisfacer simultáneamente a varios usuarios de forma selectiva y en tiempo oportuno".
[Delobel, 1982].

- "Colección no redundante de datos que son compartidos por diferentes sistemas de aplicación" .
[Howe, 1983].

- "Conjunto de ficheros maestros, organizados y administrados de manera flexible de modo que los ficheros puedan ser facilmente adaptados a nuevas tareas imprevisibles" .
[Frank, 1988].

Figura 2.1 Diferentes Definiciones de Base de Datos

En primer lugar, y en esto coinciden todas las definiciones, una base de datos, es un *conjunto, colección o depósito de datos* almacenados en un soporte informático no volátil. Los datos están *interrelacionados* y *estructurados* de acuerdo a un modelo capaz de recoger el máximo contenido semántico. Dada la relevancia que tienen en el mundo real las interrelaciones entre los datos, es imprescindible que la base de datos sea capaz de almacenar estas interrelaciones. En el mundo real existen, además, restricciones semánticas, a las que se está concediendo una importancia creciente y que, en los sistemas actuales, tienden a almacenarse junto con los datos, al igual que ocurre con las interrelaciones. La base de datos se describe y se manipula apoyándose en un modelo de datos.

La redundancia de los datos debe ser controlada, de forma que no existan duplicidades perjudiciales ni innecesarias, y que las redundancias físicas, convenientes muchas veces a fin de responder a objetivos de eficiencia, sean tratadas por el mismo sistema, de modo que no puedan producirse inconsistencias. Esto podría resumirse diciendo que en las bases de datos no debe existir *redundancia lógica,* aunque sí se admite cierta *redundancia física* por motivos de eficiencia [De Miguel, 1999].

Por tanto, un dato se actualizará *lógicamente* por el usuario de forma única, y el sistema se preocupará de cambiar físicamente todos aquellos campos en los que el dato estuviese repetido en caso de existir redundancia física; es lo que se denomina también *redundancia controlada* por el sistema. Por tanto, las bases de datos han de atender a *múltiples usuarios* y a *diferentes aplicaciones*, en contraposición a los sistemas de ficheros, en los que cada fichero está diseñado para responder a las necesidades de una determinada aplicación.

Otro aspecto importante en las bases de datos es la *independencia,* tanto física como lógica, entre datos y tratamientos. Esta independencia, objetivo fundamental de las bases de datos, es una característica esencial que distingue las bases de datos de los ficheros y que ha tenido una enorme influencia en la arquitectura de los sistemas manejadores de bases de datos (SMBD), como se verá más delante [De Miguel, 1999].

La *definición* o *descripción* del conjunto de datos contenidos en la base (lo que se denomina *estructura* o *esquema* de las bases de datos) deben ser *únicas* y estar *integradas* con los mismos datos. En los sistemas basados en ficheros, los datos se encuentran almacenados en ficheros, mientras su descripción está separada de los mismos, formando parte de los programas, para lo cual se precisa que los lenguajes faciliten medios para la descripción de los datos. Suele haber, además, una documentación adicional, habitualmente en soporte de papel, en general insuficiente y desactualizada. En las bases de datos, la descripción, y en algunos casos una definición así como, la documentación completas, se almacenan junto con los datos. De modo que éstos están *autodocumentados*, y cualquier cambio que se produzca en dicha documentación, se ha de reflejar y quedar recogido en el sistema, con todas las ventajas que de este hecho se derivan [De Miguel, 1999].

La actualización y la recuperación de los datos debe realizarse mediante *procesos bien determinados*, incluidos en el SMBD, que proporciona los instrumentos para facilitar el mantenimiento de la *seguridad* (confidencialidad, disponibilidad e integridad) del conjunto de datos [De Miguel, 1999].

El concepto de base de datos ha ido cambiando y configurándose a lo largo del tiempo; en la actualidad, y de acuerdo con estas características que se mencionaron anteriormente, se puede definir una base de datos como se muestra en la figura 2.2.

" Colección o depósito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de ellos, y su definición (estructura de la base de datos) única y almacenada junto con los datos, se apoya en un modelo de datos, que permite captar las interrelaciones y restricciones existentes en el mundo real. Los procedimientos de actualización y recuperación comunes y bien determinados, facilitarán la seguridad del conjunto de los datos [De Miguel, 1999]."

Figura 2.2 Definición de Base de Datos

El SMBD es el conjunto de programas que permiten la implantación, acceso y mantenimiento de la base de datos. El SMBD junto con la base de datos y con los usuarios, constituyen *el Sistema de Base de Datos*.

2.3 HISTORIA DEL DESARROLLO DE LAS BASES DE DATOS

Durante los cincuenta y el comienzo de los sesenta, se inicia el proceso de datos, la regla era el tratamiento de archivos secuenciales. Todos los datos se almacenaban en archivos secuenciales, que exigían el tratamiento de archivos completos por los programas de aplicación. El almacenamiento en disco utilizando el acceso directo, fue ampliamente disponible y el procesamiento de archivos de acceso aleatorio fue factible y popular. Este método permitió el acceso directo a datos específicos en un archivo.

En la medida que los sistemas computacionales de procesamiento de datos se hicieron más importantes, los negocios comenzaron a reconocer que la información era un recurso corporativo de valor considerable. Estos percibieron más y más que los datos necesarios para contestar numerosas preguntas del negocio, estaban disponibles en sus archivos de procesamiento de datos. Como consecuencia, comenzaron a presionar a los sistemas de información, para la gestión, en cuanto a la utilización de la potencia de la computadora, para producir información a partir de los datos corporativos. Esto inició la demanda de los sistemas de bases de datos, los que garantizarían más efectivamente el acceso a los datos y su manipulación [W. Hansen, 1997].

A mediados de los sesenta se introdujeron los primeros sistemas de bases de datos, cuyo fundamento era una estructura jerárquica de los datos. Estos sistemas permitieron la recuperación de múltiples registros asociados con un registro único de otro archivo. Inmediatamente después, se desarrollaron los sistemas de base de datos en redes, que soportaron relaciones mucho más complejas entre registros de archivos diferentes. Ambos modelos de base de datos, el jerárquico y el de red, requirieron el uso de punteros físicos predefinidos para enlazar los registros relacionados.

En 1970, el artículo de E. F. Codd sobre el modelo de datos relacional revolucionó el pensamiento en la industria de las bases de datos. El enfoque de Codd proponía el acceso y la manipulación de los datos únicamente desde el punto de vista de sus características lógicas. Durante los años setenta y ochenta se desarrollaron numerosos sistemas de base de datos relacionales y, en la actualidad, éstos dominan el mercado comercial [W. Hansen, 1997].

En años recientes han proliferado las computadoras personales en los puestos de trabajo, por lo que se han desarrollado las redes computacionales, permitiendo a los usuarios de estas computadoras compartir recursos.

Una computadora, que funciona como servidor de una red, garantiza el acceso a la base de datos desde las estaciones de trabajo en éstos puestos, permitiendo una división poderosa y eficiente de las tareas. El servidor recupera los datos, que la máquina cliente solicita, para su manipulación.

Las redes computacionales en ambiente cliente/servidor han desarrollado un alto grado de sofisticación, y se encuentra cada vez con más frecuencia en las empresas comerciales.

2.4 OBJETIVOS DE UNA BASE DE DATOS

Las bases de datos se diseñan para manejar grandes cantidades de información, la manipulación de los datos involucran, tanto, la definición de estructuras para el almacenamiento de la información, como la provisión de mecanismos para la manipulación de la misma. Además, una base de datos debe tener implementados mecanismos de seguridad, que garanticen la integridad de la información, a pesar de caídas del sistema o intentos de accesos no autorizados [Tsai, 1990].

El objetivo principal, es proporcionar a los usuarios finales una visión abstracta de los datos, esto se logra escondiendo ciertos detalles de como se almacenan y mantienen los datos, además de los mostrados a continuación [Tsai, 1990]:

Evitar redundancia e inconsistencia de los datos. El guardar datos redundantes es, evidentemente, un desperdicio de espacio de almacenamiento. Sin embargo, el problema más serio asociado con la redundancia de los datos es su inconsistencia. Esto generalmente ocurre como resultado de actualizaciones incompletas de datos duplicados. En los sistemas antiguos los datos redundantes se actualizan por medio de programas distintos, donde cada programa se usa para actualizar a su propio archivo afectado. Consecuentemente, los costos de mantenimiento son altos porque hay que desarrollar y ejecutar más de un programa. Además, el mantenimiento de la consistencia de los datos en un sistema convencional es una tarea difícil y propensa a errores [Tsai, 1990].

En un ambiente dinámico, las operaciones de inserción, eliminación y actualización se efectúan con mucha frecuencia. Es entonces imperativo, que la base de datos evite la inconsistencia de los mismos, minimizando la redundancia y manteniendo la integridad de los datos.

Anomalías del acceso concurrente. Para mejorar el funcionamiento global del sistema y obtener un tiempo de respuesta más rápido, muchos sistema permiten que múltiples usuarios actualicen los datos simultáneamente. En este entorno la interacción de actualizaciones concurrentes puede dar por resultado datos inconsistentes. Para prevenir esta posibilidad debe mantenerse alguna forma de supervisión en el sistema [Tsai, 1990].

Tiempo de respuesta. Las base de datos diseñadas para ser utilizadas de forma interactiva deben de asegurar un tiempo de respuesta adecuado para el diálogo entre el servidor y el cliente. Además, el sistema debe de tener la capacidad suficiente para manejar un número concreto de maquinas cliente, y el flujo de transacciones a que éstos dan origen. Esto es importante en los sistemas interactivos de alto volumen de tráfico, cuando los datos deben de actualizarse al momento de haber sufrido una variación.

Integridad de los datos. La integridad de los datos se refiere a medidas de seguridad usadas para mantener correctos los datos. La tecnología de bases de datos proporciona recursos que refuerzan la integridad de los datos, sin necesidad de una programación extra por parte del usuario.

Es conveniente señalar que la integridad de los datos es más importante en una base de datos, que en un sistema de archivos privados, precisamente porque el primero se comparte y porque sin procedimientos de validación adecuados, es posible que un programa con errores genere datos incorrectos, que afecten a otros programas al utilizar la información [Date, 1986].

Seguridad de los datos. La seguridad de los datos está relacionada fuertemente con la integridad de los mismos Y hace refrencia a la protección de la base contra accesos o modificaciones no autorizados. Como en las bases de datos los datos, se comparten ampliamente, la información confidencial en la base es vulnerable a las intromisiones. Un acceso ilegal puede tener como resultado la destrucción ilegal o maliciosa de la base. Sin control de seguridad, los usuarios no tendrán la privacía requerida en sus datos confidenciales y el sistema no podrá mantener la integridad de los mismos.

Independencia de datos. Se refiere a la protección contra los programas de aplicación al ocurrir modificaciones ocasionadas cuando se altera la organización física o estructura lógica de la base. El costo de cambiar la estructura lógica o física de un sistema de información no integrado como una base de datos es, principalmente el de modificar los programas ya existentes en el sistema [Tsai, 1990].

Datos Compartidos. Esto significa que las aplicaciones existentes pueden compartir los datos de la base, además de que es factible desarrollar nuevas aplicaciones que operen con los mismos datos almacenados. En otras palabras, las necesidades de datos de la nuevas aplicaciones pueden atenderse sin tener que crear nuevos archivos almacenados [Date, 1986].

2.5 ARQUITECTURA DE UNA BASE DE DATOS

La distinción entre la representación lógica y física de los datos fue reconocida oficialmente en 1978, cuando el comité ANSI/SPARC propuso un esqueleto generalizado para sistemas de bases de datos (Tsichritzis y Klug, 1978). Este esqueleto brinda una arquitectura de tres niveles de abstracción bajo los que podría verse una base de datos, estos son: el externo, el conceptual y el interno [Hansen, 1997].

La siguiente figura muestra la arquitectura de una base de datos, incluyendo varios de los elementos que conforman cada nivel de la arquitectura.

2.5.1 NIVEL EXTERNO

Consiste de las vistas que tiene el usuario sobre la base de datos. Cada grupo de usuarios tendrá su propia vista de la base de datos. Cada una de estas vistas da una descripción de los elementos de los datos, sus relaciones orientadas al usuario y de las cuales se compone la vista. Ésta se puede derivar directamente del esquema conceptual. La colección de todas las vistas de usuario forma el nivel externo.

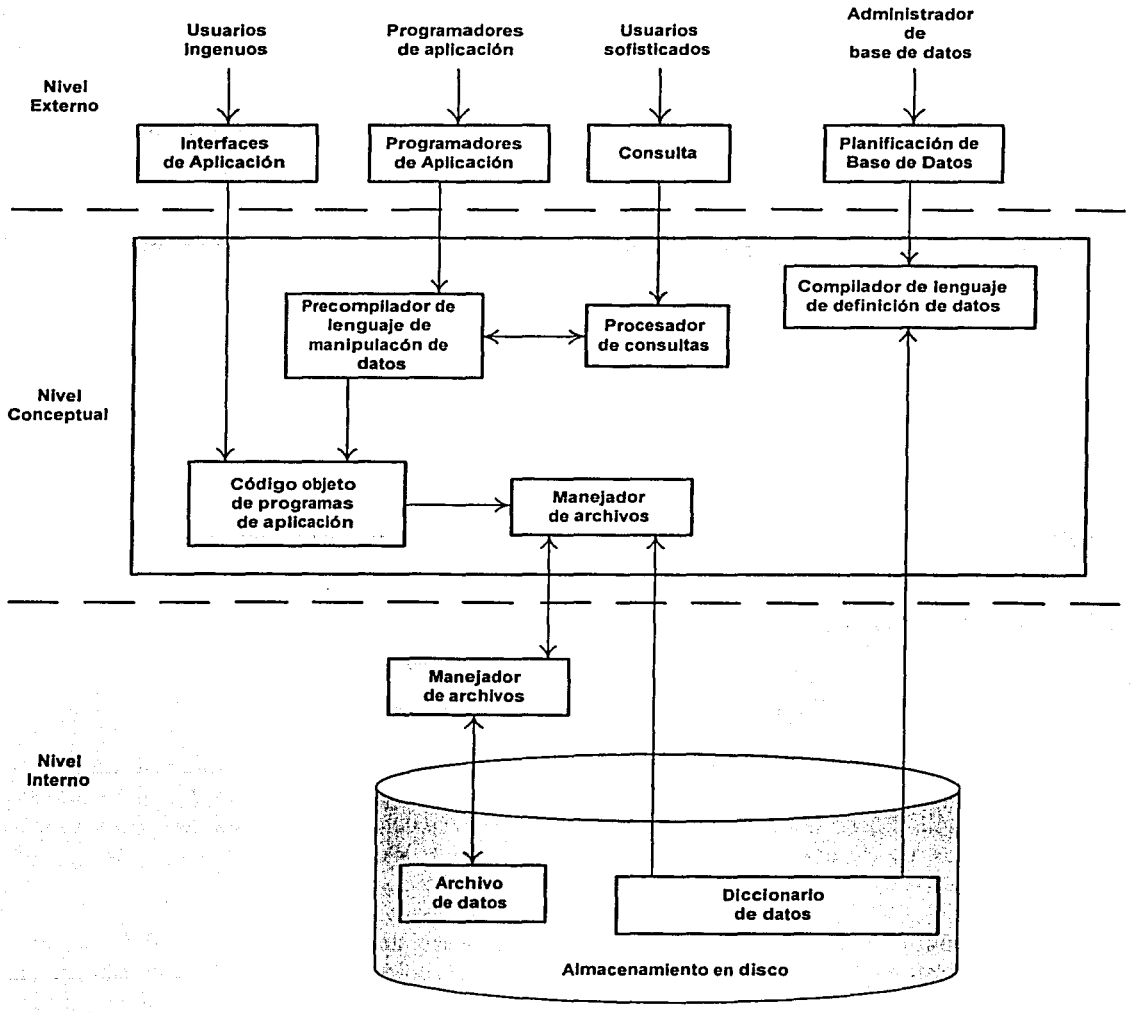


Figura 2.3 Arquitectura de una base de datos

2.5.1.1 COMPONENTES DEL NIVEL EXTERNO

Dentro del nivel externo se encuentran los usuarios, donde un usuario es toda persona que tenga todo tipo de contacto con la base de datos, desde su diseño, elaboración, terminación y utilización

Hay cuatro tipos de usuarios, diferenciados por la forma de interaccionar con el sistema:

Programadores de aplicaciones. Los profesionales en computación interaccionan con el sistema por medio de llamadas en el lenguaje de definición de datos (Data Definition Language, DML), incorporadas en un programa escrito en un lenguaje principal (como Pascal o C). Estos programas se denominan comúnmente programas de aplicación. Puesto que la sintaxis de DML es normalmente diferente a la del lenguaje principal, las llamadas en DML van normalmente precedidas de un carácter especial, de tal forma que se pueda generar el código apropiado.

Precompilador de DML. Es un precompilador especial que convierte las sentencias en DML a llamadas en el lenguaje principal. El programa resultante se ejecuta entonces por el compilador del lenguaje principal, que genera el código objeto apropiado.

Usuarios sofisticados. Interaccionan con el sistema sin escribir programas, en cambio escriben sus preguntas en un lenguaje de consultas. Cada consulta se somete a un procesador de consultas, cuya función es tomar una sentencia en DML y descomponerla en instrucciones que entienda el manejador de la base de datos.

Usuarios especializados. Algunos usuarios sofisticados desarrollan aplicaciones especializadas de base de datos que no encajan en el marco tradicional de procesamiento de datos, como diseño asistido por computador, sistemas basados en el conocimiento, etc.

Usuarios ingenuos. Los usuarios no sofisticados interactúan con el sistema invocando a uno de los programas de aplicación permanentes que se han escrito anteriormente.

2.5.2 NIVEL CONCEPTUAL

Es el nivel en el que se hace el diseño conceptual de la base de datos. Este diseño implica el análisis de las necesidades de información de los usuarios y la definición de las clases de datos que se necesitan para satisfacer estas necesidades. El resultado del diseño conceptual es el esquema conceptual, una simple y lógica descripción de todos los elementos de los datos y sus relaciones.

El administrador de la base de datos define el modelo conceptual a partir del esquema conceptual. Este nivel representa la visión organizacional de la base que se obtiene al integrar los requerimientos de todos los usuarios. Un esquema conceptual consta de las siguientes definiciones:

- Definición de los datos. En el esquema se describen el tipo de datos y la longitud de campo de todos los elementos direccionables en la base. Los elementos por definir incluyen artículos elementales (atributos), totales de datos (artículos de grupo), y registros conceptuales (entidades).

- Relaciones entre datos. En el esquema se definen relaciones entre datos para enlazar tipos de registros relacionados para el procesamiento de archivos múltiples.

Un esquema conceptual se formula sin importar el almacenamiento físico de los registros correspondientes. En el nivel conceptual la base de datos aparece sólo como una colección de registros lógicos, sin descriptores de almacenamiento. En realidad los archivos conceptuales no existen físicamente. La transformación de registros conceptuales en registros físicos para su almacenamiento se lleva a cabo por el sistema y es enteramente transparente para el usuario [Tsai,1990].

2.5.2.1 COMPONENTES DEL NIVEL CONCEPTUAL

Lenguaje de definición de datos. Un esquema de base de datos se especifica por medio de un conjunto de definiciones que se expresan mediante un lenguaje especial llamado lenguaje de definición de datos (DDL). El resultado de la compilación de sentencias de DDL es un conjunto de tablas que se almacenan en un archivo especial que llamado diccionario de datos o directorio.

Directorio de datos. Es un archivo que contiene metadatos, es decir, datos sobre datos. Este archivo se consulta antes de leer o modificar los datos reales en el sistema de base de datos. La estructura de almacenamiento y los métodos de acceso se especifican por medio de un conjunto de definiciones, en un tipo especial de DDL llamado lenguaje de almacenamiento y definición de datos. El resultado de la compilación de estas definiciones es un conjunto de instrucciones, que especifican los detalles de implementación de los esquemas que normalmente se esconden a los usuarios.

Lenguaje de manipulación de datos. La manipulación de datos se entiende como la recuperación y modificación de la información almacenada, además de la inserción y supresión de información. A nivel físico, se deben definir algoritmos que permitan acceso eficiente a los datos. En los niveles de abstracción más altos, se pone énfasis en la facilidad de uso. El objetivo es proporcionar una interacción eficiente entre las personas y el sistema.

Un lenguaje de manipulación de datos (Data Manipulation Lenguage, DML) es un lenguaje que capacita a los usuarios a acceder o manipular los datos. Existen básicamente dos tipos :

- Procedimentales. Requieren que el usuario especifique qué datos se necesitan y cómo conseguirlos.

- No procedimentales. El usuario debe especificar qué datos se necesitan, pero no cómo obtenerlos.

Los DML no procedimentales, normalmente son más sencillos de aprender y usar, sin embargo pueden generar código que no sea tan eficiente como el producido por los lenguajes procedimentales. Esta dificultad puede remediarse a través de varias técnicas de optimización.

Consulta. Es una sentencia que solicita la recuperación de información. El trozo de un DML que implica recuperación de información se llama lenguaje de consultas. Aunque es incorrecto, suelen utilizarse los términos lenguaje de consultas y lenguaje de manipulación de datos como sinónimos.

Precompilador de DML. Convierte las sentencias DML incorporadas en un programa de aplicación en llamadas normales a procedimientos del lenguaje principal. Debe interaccionar con el procesador de consultas para generar el código apropiado.

Compilador de DDL. Convierte sentencias DDL en un conjunto de tablas que contienen metadatos.

Procesador de consultas. Traduce sentencias en un lenguaje de consultas a instrucciones de bajo nivel que entiende el manejador de la base de datos.

Manejador de archivos. Gestiona la asignación de espacio en la memoria del disco y de las estructuras de datos usadas para representar la información almacenada en el disco.

2.5.3 NIVEL INTERNO

Es la representación del nivel inferior de una base de datos. Mapea a la base lógica hacia el almacenamiento físico y establece trayectorias de datos (por ejemplo, mediante señaladores e índices) para el acceso aleatorio a la base de datos. Este nivel es responsabilidad de los diseñadores de la base de datos física, quienes deciden cuales dispositivos físicos contendrán los datos, cuáles métodos de acceso se usarán para recuperar y actualizar los datos, y cuáles medidas de tomarán para mantener o mejorar el rendimiento de la base de datos. Ningún usuario tiene que ver con esta vista en calidad de usuario [Hansen, 1997].

2.5.3.1 COMPONENTES DEL NIVEL INTERNO

El nivel interno es una representación a nivel muy bajo de la base de datos. La vista interna se describe por medio de un esquema interno, el cual no sólo define los diversos tipos de registros almacenados, sino que también especifica cuáles índices existen, de que manera se representan los campos almacenados, en que secuencia física se hallan los registros almacenados, etc. Estas características se definen a través de los siguientes elementos:

Diccionario de Datos. Que almacena metadatos sobre la estructura de la base de datos. Se usa ampliamente, por lo que debe ponerse mucho énfasis en le desarrollo de un buen diseño y una implementación eficiente del diccionario.
Indices. Proporcionan acceso rápido a los elementos de datos que contienen valores determinados.

Archivos de datos. Utilizados para almacenar la base de datos.

2.6 DEFINICIÓN DE UN SMBD

Un Sistema Manejador de Base de datos (SMBD) es una pantalla entre los usuarios y las memorias secundarias, que tiende a crear la ilusión de que los datos deseados por cada uno de los usuarios están almacenados en las memorias secundarias, agrupados y codificados como se desee, como si el usuario fuese el único que va a utilizar los datos [Gardarin, 1990].

A esto se le conoce cómo niveles de abstracción de la información, y varían los niveles dependiendo del tipo de usuario. Para la mayoría, se ocultan los detalles de la forma y el lugar en que están almacenados los datos, así como los procedimientos de recuperación y actualización de la información [Parrilla, 1997]. Un punto de vista bastante estándar de los niveles de abstracción se muestra en la figura 2.4, en ella se muestra una base de datos, que puede ser una de varias usadas por el mismo SMBD, a tres niveles de abstracción. Se puede pensar que la base es una colección de archivos y seguramente de estructuras de datos en lugar de una colección de bits.

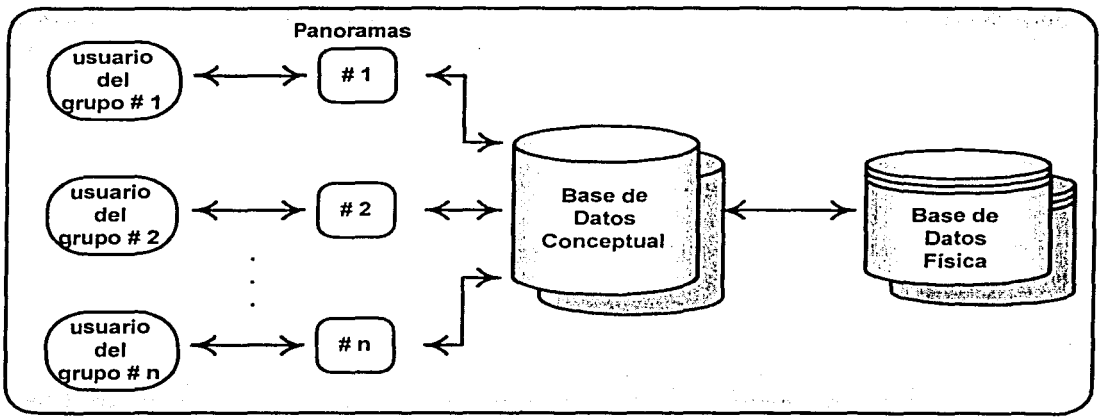


Figura 2.4 Niveles de Abstracción de una Base de Datos

La base de datos conceptual es una representación abstracta de la base de datos física, y los panoramas son abstracciones de porciones de la base de datos conceptual [Begoña, 1989].

Un SMBD es también una herramienta que permite insertar, modificar y buscar eficazmente datos específicos de entre un volumen masivo de información compartida por todos los usuarios.
Las búsquedas se pueden realizar a partir del nombre de un dato. Un SMBD se compone generalmente de un conjunto de paquetes "software", pero también puede contar con elementos "hardware" especializados. En resumen, un SMBD puede aparecer como una herramienta para ordenar, buscar, recuperar y convertir datos. Estas son unas funciones primarias utilizadas por funciones que suelen ser más complejas para proteger los datos de cualquier incidente o accidente y obtener un rendimiento aceptable del sistema. Los SMBD se distinguen claramente de los sistemas de archivos por el hecho de que permiten la descripción de los datos (definición de los nombres, formatos y características), de forma independiente de su utilización (actualización y búsqueda) [Gardarin, 1990].

Un SMBD se compone, en una primera aproximación, de tres niveles o capas sucesivas de funciones, apiladas desde las memorias secundarias hacia los usuarios.

- La gestión de los recipientes de datos en las memorias secundarias se compone, tradicionalmente, de la primera capa; es el *sistema de manejo de archivos.*

- El manejo de los datos almacenados en los archivos, la colocación y la agrupación de estos datos, el manejo de los enlaces entre datos y de las estructuras que permiten recuperarlos rápidamente, constituyen la segunda capa; es el sistema de acceso a los datos o SMBD *interno* que cada vez se hace más invisible a los programas de las aplicaciones.

- La presentación de los datos a los programas de las aplicaciones y a los usuarios terminales que han expresado sus necesidades de datos, con la ayuda de lenguajes más o menos elaborados, constituye la función esencial de la tercera capa: es el SMBD *externo* que asegura por una parte, el análisis y la interpretación de las peticiones de los usuarios y, por otra, el formateo de los datos intercambiados con el mundo exterior.

En la figura 2.5, aparecen estas tres capas de funciones que constituyen un SMBD.

2.7 RESEÑA HISTÓRICA DE LOS SMBD

Los SMBD tienen ya una larga historia. Durante los años 60's se conoció el primer desarrollo de los sistemas de archivos que esencialmente intentaba hacer parecer a las memorias secundarias como ideales, compartidas, simplificadas, directamente direccionables y de capacidad infinita. Son éstos los sistemas que después de haber evolucionado, componen hoy en día el corazón de los SMBD. A mediados de los 60's tuvo ligar el nacimiento de la primera generación de SMBD que estaba caracterizada por la separación de la descripción de los datos de los programas de aplicación y la llegada de los lenguajes de acceso de navegación, es decir, que permiten desplazarse por las estructuras de tipo grafo y obtener, uno a uno, grupos de datos llamados artículos.

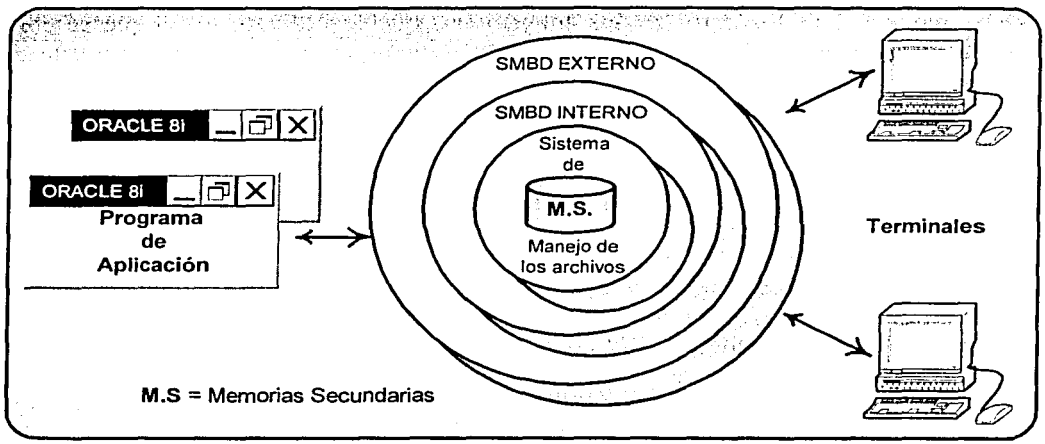


Figura 2.5 Primera vista de un SMBD

Esta primera generación, representada por los sistemas que cumplían las primeras recomendaciones del CODASYL y por el IMS de IBM, está basada en los modelos de acceso, es decir, en los modelos de datos que intentan primordialmente optimizar los métodos de colocación de los datos en las memorias secundarias a fin de reducir los tiempos de acceso [Gardarin, 1990].

Desde 1970, la segunda generación de SMBD ha crecido en los laboratorios, a partir del modelo relacional. Esencialmente, trata de enriquecer el SMBD externo con el fin de simplificar el acceso a los datos para los usuarios externos. Así, la segunda generación ofrece unos lenguajes de afirmación basados en la lógica, permitiendo especificar los datos que se desean obtener sin decir cómo hay que accederlos. Es el sistema el que debe determinar el mejor plan de acceso posible. Esta segunda generación vuelve a adoptar, después de haberlos hecho evolucionar y convertido en más dinámicos, algunos modelos de acceso de la primera generación a nivel del SMBD interno, a fin de optimizar aún más los accesos. Los primeros sistemas de la segunda generación se comercializaron a partir de los primeros años 80's y todavía hay que experimentar con ellos.

Una tercera generación de SMBD, basada en lenguajes de acceso más potentes y más naturales, soportando tipos de datos más variados, incluyendo posibilidades de deducciones y de distribución, se encuentra ya en estudio en los laboratorios de investigación [Gardarin, 1990].

2.8 OBJETIVOS DE UN SMBD

Un SMBD debe cumplir una serie de objetivos para ser lo más rápido, eficiente, polivalente y seguro posible [Parrilla, 1997].

- Debe independizar los datos de las aplicaciones que lo utilizan. Es lo que se conoce como *independencia física* (se puede modificar el esquema físico sin que afecte a los superiores) e *independencia lógica* (si se modifica el esquema conceptual no es necesario modificar los programas de aplicación).

- Conseguir que los datos repetidos innecesariamente en la BD sean los mínimos posibles (redundancia mínima de información).

- Suministrar mecanismos de seguimiento de las operaciones realizadas en la base de datos. Esto se consigue mediante procesos "espías" que mantienen archivos en los que se almacena la fecha y hora de conexión de los usuarios, las operaciones realizadas en cada sesión, los datos modificados, etc.

- Proporcionar versatilidad en las posibilidades de búsqueda de información facilitando al usuario varios criterios.

- Asegurar la protección de los datos contra accesos no autorizados o malintencionados de los usuarios.

- Controlar la integridad de la información en la BD. Para ello, debe dar respuesta a posibles fallos de "hardware", defectos en el código de los programas de aplicación, actualizaciones incompletas, inserción de datos incorrectos o no válidos, etc.

- Proporcionar mecanismos para realizar copias de seguridad de la información.

- Conseguir un tiempo de respuesta suficientemente pequeño para evita que el usuario se desespere. El *tiempo de respuesta* se define como el tiempo que transcurre desde que el usuario termina de realizar una petición al sistema hasta que empieza a recibir respuesta.

- Solucionar los problemas planteados por la concurrencia. Actualmente es posible que una BD esté compartida por varios usuarios situados en distintas terminales. Por ello, puede darse el caso de que dos o más usuarios distintos intenten actualizar de forma concurrente (en el mismo instante de tiempo) un registro determinado. Fundamentalmente, estos problemas son la actualización incorrecta y el bloqueo mutuo.

2.9 ARQUITECTURA DE UN SMBD

El SMBD se encarga de conectar e implantar los distintos niveles de la arquitectura de la BD. Uno de sus componentes, el *sistema de control de datos* (SCBD), es el encargado de transformar los datos requeridos por los programas de aplicación en registros físicos a leer, hacer la petición al sistema operativo y, cuando la información está disponible, transferirla al área de trabajo del programa que la solicitó. La figura 2.6 sintetiza este proceso.

A través del SMBD, una persona o grupo de personas, llamado *administrador del sistema,* organiza y controla los recursos del sistema [Parrilla, 1997]. Sus principales funciones son:

- Definir el esquema conceptual con el lenguaje de definición del SMBD.
- Controlar el acceso a la BD, concediendo permisos a los usuarios.
- Definir estrategias de recuperación frente a posibles fallos

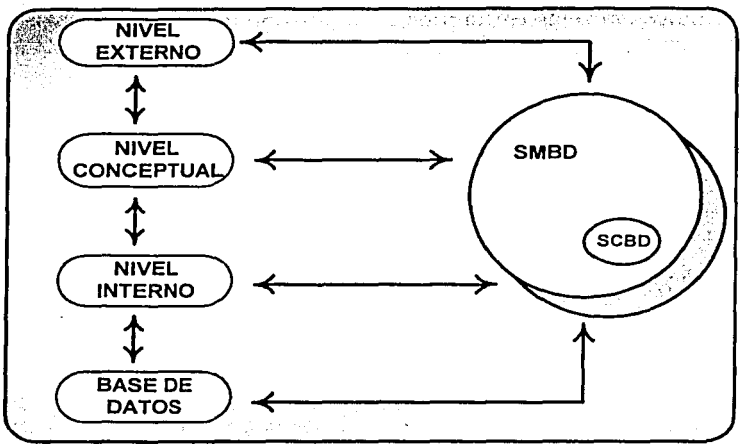


Figura 2.6 Iteración de Niveles con el SMBD

2.9.1 NIVEL EXTERNO

A continuación se describen brevemente los primeros niveles de la figura 2.6. En el nivel externo, cada grupo de trabajo que utiliza datos posee una descripción de los datos percibidos. Esta descripción se efectúa según la manera en que el grupo ve la base en sus programas de aplicación. Cuando en el nivel conceptual e interno los esquemas describen toda una base de datos, en el nivel externo éstos describen simplemente la partida de datos que presenta interés para cada usuario o grupo de usuarios. En consecuencia, un esquema externo a menudo es calificado de vista externa.

Hay que señalar que la noción de esquema externo permite asegurar cierta seguridad de los datos. En efecto, un grupo de trabajo sólo puede acceder a los datos descritos en el esquema externo. Los otros datos se protegen así contra los accesos no autorizados o mal intencionados de parte del grupo de trabajo [Gardarin, 1994].

2.9.2 NIVEL CONCEPTUAL

El nivel central es el conceptual, ya que corresponde a la estructura canónica de los datos que existen en la empresa, es decir, su estructura semántica inherente, sin implantación en máquina, representando la visión integrada de todos los grupos de trabajo. Por ejemplo, el esquema conceptual permite definir [Gardarin, 1994]:

- Los tipos de datos elementales que definen las propiedades elementales de objetos de la empresa.
- Los tipos de datos compuestos que permiten reagrupar los atributos para describir los objetos del mundo real o las relaciones entre los objetos.
- Los tipos de datos que permiten reagrupar los atributos para describir las asociaciones del mundo real.
- Eventualmente, las reglas que deberán seguir los datos a lo largo de su vida en la empresa.

2.9.3 NIVEL INTERNO

El nivel interno corresponde a la estructura de almacenamiento que sostienen los datos. Por lo que permite describir los datos del modo en que son almacenados en la máquina, por ejemplo [Gardarin, 1994]:

- Los archivos que lo contienen (nombre, organización, localización...etc.).
- Los registros de estos archivos (longitud, componentes de los campos, modo de desplazamiento en los archivos...etc.).
- Las vías de acceso a estos registros (índices, tablespaces, archivos invertidos...etc.).

CAPÍTULO 3

MODELOS DE DATOS

3.1 INTRODUCCIÓN

El almacenamiento, manipulación y recuperación de información en forma eficiente, es vital y estratégico para cualquier organización. Las bases de datos juegan un rol crítico en casi todas las áreas donde las computadoras son usadas, incluyendo negocios, ingeniería, medicina, leyes, educación, etc. Una característica fundamental del enfoque de BD es que proporciona cierto nivel de abstracción de los datos, al ocultar detalles de almacenamiento que la mayoría de los usuarios no necesitan conocer. Los modelos de datos son el principal instrumento para ofrecer dicha abstracción.

3.2 CONCEPTOS DE MODELO DE DATOS

Modelo. Instrumento que se aplica a una parcela del mundo real (un *universo del discurso*) para obtener una estructura de datos a la que se denomina *esquema.*

Modelar. Ejemplar que uno se propone imitar. (Diccionario ideológico de Julio Casares) [Hernández, 2001].

Un modelo de datos. Es un conjunto de conceptos que pueden ser usados para describir la estructura de una BD. Con el concepto de *estructura de una BD* nos referimos a los tipos de datos, las relaciones y las restricciones que deben cumplirse para esos datos. Por lo general, los modelos de datos contienen un conjunto de operaciones básicas para especificar lecturas y actualizaciones de la base de datos.

Los modelos de datos aportan la base conceptual para diseñar aplicaciones, que hacen un uso intensivo de datos. Así como la base formal para las herramientas, técnicas empleadas en el desarrollo y uso de sistemas de información.

Un modelo de datos es por tanto una colección de conceptos bien definidos matemáticamente, que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación, con un uso de datos intensivo [Moreno, 2000]. Conceptualmente una aplicación puede ser caracterizada; como se muestra en la figura 3.1.

Un modelo de datos se distingue de otro por el tratamiento que da a estas tres características. Sin embargo el resultado de un modelo de datos es una representación que tiene dos componentes, como se muestra en la figura 3.2

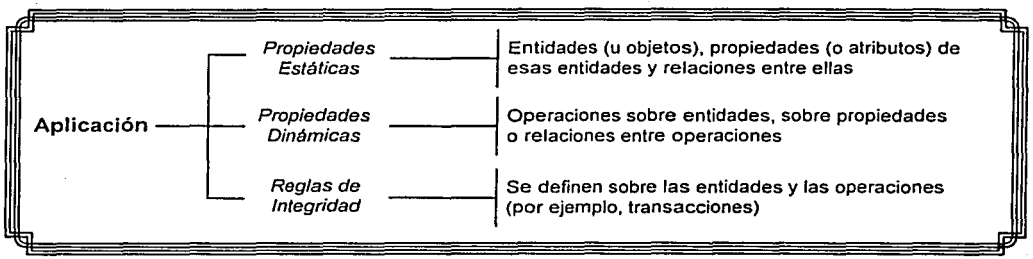


Figura 3.1 Propiedades y Reglas de una Aplicación

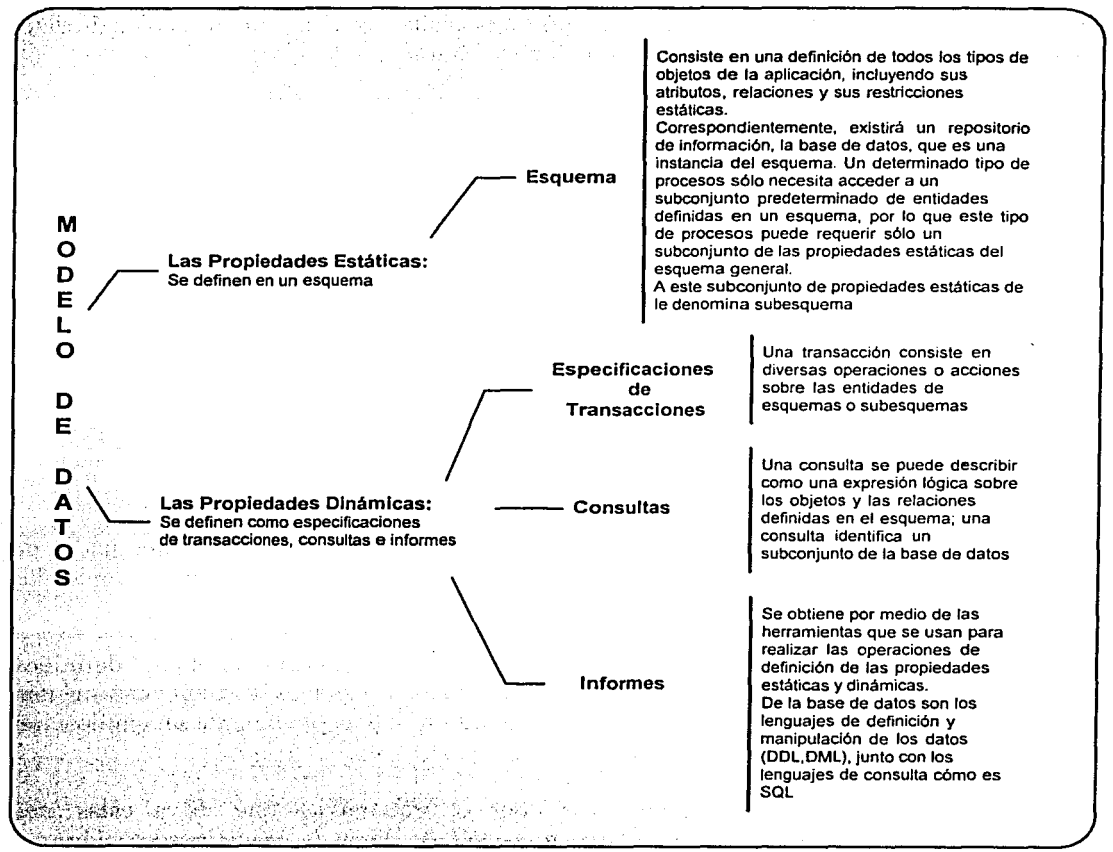


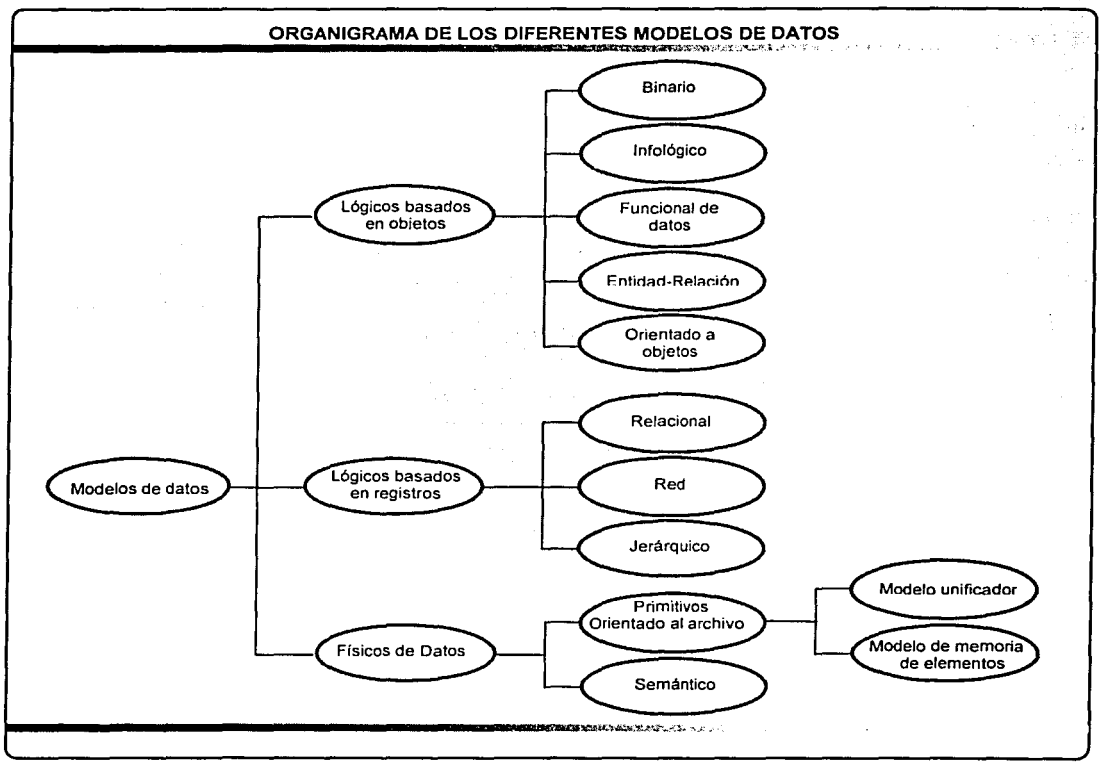
Figura 3.2 Características del Modelo de Datos

La investigación moderna sobre modelos de datos se ha centrado en los aspectos lógicos de las bases de datos y sobre los conceptos, herramientas y técnicas para el diseño de las mismas [Brodie, 1984]. Aspectos relativos a la implantación de los modelos, tales como velocidad de ejecución, concurrencia, integridad física y arquitecturas no son factores relevantes en el estudio de análisis de modelos de datos.

3.3 MODELOS DE DATOS

Los principales objetivos del proceso de modelamiento es saber cuál es el problema y encontrar la forma de representarlo en un sistema. Esto significa saber de los datos, saber quienes van a usarlos y como van a ser usados [Guerrero, 2001].

El siguiente organigrama muestra los tipos de modelos de datos.



Organigrama 3.1 Tipos de Modelos de Datos

A continuación se describen los modelos de datos expuestos en la figura anterior.

3.3.1 MODELOS DE DATOS LÓGICOS BASADOS EN OBJETOS

Los modelos lógicos basados en objetos poseen un mayor nivel de abstracción que los modelos lógicos basados en registros; una consecuencia inmediata de esto es la flexibilidad y mayor capacidad semántica de los lógicos basados en objetos. Este tipo de modelos tienen como finalidad ayudar al diseño de bases de datos en la fase de representación del universo del discurso, por lo que no suele ser habitual que se encuentren implementados en los DBMS. Los modelos lógicos basados en objetos, y muy especialmente el Entidad-Relación, son la base de las herramientas de ayuda al diseño asistido por computadora (CASE) .

En general, este tipo de modelos, por su nivel de abstracción y su riqueza semántica, constituyen una interfaz útil entre el informático y los usuarios finales en las primeras etapas del proceso de diseño de bases de datos [De Miguel, 1999].

3.3.1.1 MODELO BINARIO

Un modelo de datos binario es un modelo gráfico que se representa simplemente con nodos, atributos simples y arcos, que representan tipos de relaciones binarias entre dos atributos.

Los nodos pueden representar grupos o grupos extendidos. Los arcos pueden representar una adición de dos atributos de un tipo de entidad o una adición de dos tipos de entidades dentro de un tipo de relación. El significado exacto de un arco depende de la interpretación que se de a los dos nodos conectados. Las n-áreas adicionales de atributos y tipos de entidades pueden también ser representadas usando nodos internos [Tsichritzis, 1982].

La estructura gráfica de modelos de datos binarios, pueden ser consideradas como un doble de la estructura de tablas del modelo de datos relacional. El modelo de datos binario recibe la mayor atención como modelo de datos gráfico. Muchas declaraciones tienen que ser formuladas para la superioridad de uno sobre otro para modelar datos.

El modelo de datos binario puede representar relaciones complejas concisas, pero este no es orientado al usuario. La semántica del modelo de datos binario puede ser considerado como un puente entre el modelo de datos de red semántico (otros modelos de datos).

Un nodo representa una clasificación de instancias de datos dentro de un tipo llamado, categoría. El arco representa un tipo de relación binaria entre categorías y es llamado una relación binaria. Un gráfico que soporta esta estructuración de reglas es llamado un tipo de gráfico, que representa la intención de una base de datos binario semántica. La figura 3.3 muestra un tipo de gráfico consistente de dos categorías y una relación binaria estructurado de acuerdo al modelo de datos binario [Tsichritzis, 1982].

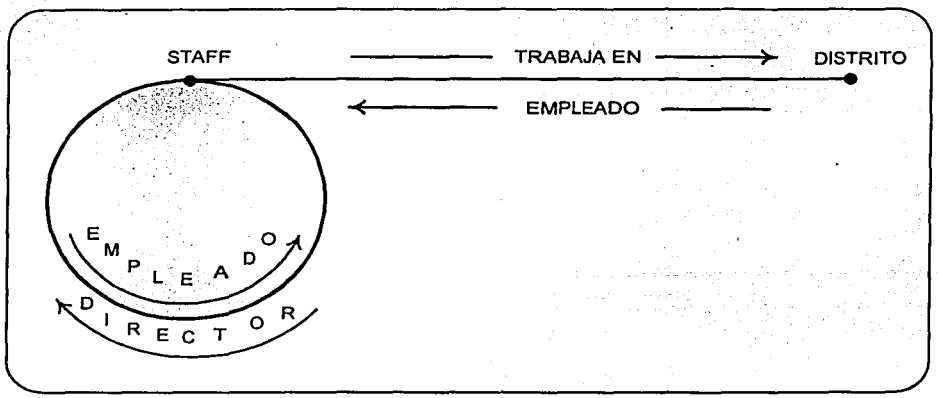


Figura 3.3 Tipo de gráfica de categorías y relaciones binarias

En el modelo de datos binario las dos direcciones de una relación binaria es llamada única, como se muestra en la figura 3.3. Cada nombre, es llamada una función de acceso que corresponde a una relación binaria seguida de una dirección. Este modelo define funciones de acceso para ambas direcciones de una relación binaria. Algunos modelos especifican funciones de acceso en una dirección, como se muestra en la figura 3.4.

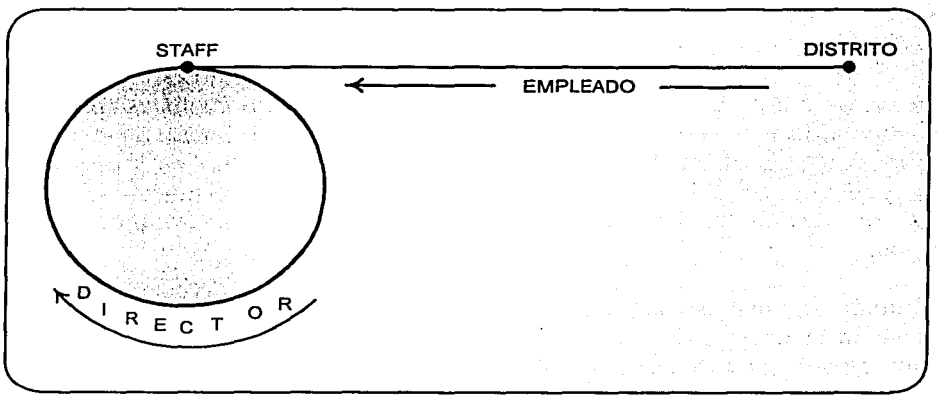


Figura 3.4 Función de acceso unidireccional

Las dos direcciones de una relación binaria pueden también ser definidas usando un nombre genérico, como en la figura 3.5. En este caso hay una dificultad cuando una relación binaria es definida entre las mismas categorías [Tsichritzis, 1982].

Deberá ser notado que las funciones de acceso en el modelo de datos binario, no son realmente funciones en el sentido matemático. Sus aplicaciones pueden producir más de un valor. Otras versiones del modelo de datos binario tienen que ser propuestas cuando las funciones de acceso hacen corresponder a funciones de un valor único.

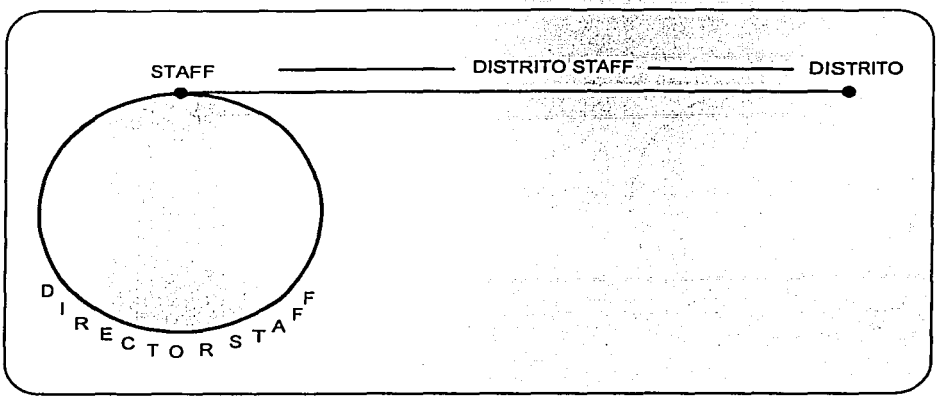


Figura 3.5 Relaciones binarias genéricas

La extensión de una semántica, del gráfico de un tipo binario consiste de objetos y conexiones entre objetos. Un objeto es una instancia de una categoría. Los objetos son de dos clases: objetos abstractos y objetos concretos. La diferencia entre objetos abstractos y concretos es similar a la noción de extensión de dominio (objeto abstracto) y extensión de atributo (objeto concreto). Esto es, los objetos abstractos existen una vez y para todos, mientras que los objetos concretos ingresan y salen de la descripción del mundo representado por el modelo de datos. Todas las categorías en el modelo de datos binario están creadas específicamente como abstractas o concretas [Tsichritzis, 1982].

Los objetos están relacionados por conexiones entre ellos. Una conexión es una instancia de una relación binaria. Cada conexión es etiquetada en ambas direcciones con el nombre de la función de acceso asociada. Las conexiones pueden ser seguidas en cualquiera de las direcciones, aunque con diferentes nombres. Para crear una relación binaria se especifica el nombre de la relación, la función de acceso y las categorías de los objetos que están relacionados.

En la definición original del modelo, un nombre para una relación binaria es opcional. Esto es, una relación binaria puede ser definida sin algún nombre asociado. Si hay únicamente una relación binaria entre dos tipos de objetos, estas no necesitan un nombre. Sin embargo, en las siguientes, para propósitos de referencia, usualmente se especifica un nombre para una relación binaria. Las funciones de acceso son funciones multivaluadas. Esto es, hay funciones para el conjunto de objetos en la categoría de su rango.

Una forma de indicar restricciones para dos funciones de acceso, es que sean inversas de otra. Esto implica que cada conexión es establecida en ambas direcciones. Las restricciones son especificadas implícitamente cuando las dos funciones de acceso están definidas con la misma definición de relación binaria [Tsichritzis, 1982].

Más tipos de restricciones pueden ser especificados por el modelo. La mayoría de ellas sin embargo, se especifican vía operaciones. Este acceso corresponde a vistas ligeramente diferentes de restricciones que pueden ser consideradas extremas. Las restricciones tienen que ser descritas principalmente en una forma declarativa como propiedades que deben ser verdaderas en algún estado de la base de datos.

La definición de una operación incluye cualquier restricción; la operación es supuesta para verificar y aplicar. Para cada nivel de una instancia, el modelo provee operaciones para manipular objetos en una categoría y para manejar conexiones entre objetos.

Todas las consultas a una base de datos están formuladas como conjeturas que pueden ser verdaderas o falsas. Los sucesos o fallas de una conjetura provee las respuestas a las consultas. En estos modelos de datos, los datos están marginados usando selecciones y entonces son accesados. La información es puesta en la base de datos para crear objetos y conexiones [Tsichritzis, 1982].

3.3.1.2 MODELO INFOLÓGICO

El campo infológico es independiente de los atributos físicos de los dispositivos de almacenamiento. En realidad los dispositivos de almacenamiento final podrían ser un humano, antes que una computadora o en su defecto la memoria.

Similarmente el campo de datos lógico y el mapeo entre ellos fue introducido primero por Langefors y fue estudiado intensivamente por Sweden. Muchas otras investigaciones a través del mundo trabajan en modelos de datos orientados a capturar requerimientos de información. Sin embargo, el trabajo de Langefors con respecto a Sweden es mucho más extensivo y comprensivo, por esta razón, se señala principalmente el trabajo del segundo [Tsichritzis, 1982].

Un modelo de datos infológico concierne en un sentido muy formal, con la representación de como existen las estructuras de datos en el mundo real. Hay otros modelos generales, que pueden ser usados para este propósito. El primero y el más extensamente usado, es el lenguaje natural, tal cómo el inglés. La principal objeción para usar el lenguaje natural como modelo de datos infológico, son las dificultades obvias con el procesamiento computacional y la ambigüedad inherente del lenguaje natural. La facilidad de fundar la filosofía en el lenguaje natural, es por ser insustituible, por esta razón ellos proponen el uso de la lógica formal, tal como el cálculo de predicado.

Sin embargo en una representación del mundo real, el tiempo es un ingrediente esencial y el cálculo de predicados carece de referencia. El modelo de datos infológico, trata los datos formalmente en la manera de cálculo de predicados, aunque también introduce el tiempo como una idea esencial del modelado de datos.

3.3.1.2.1 ESTRUCTURA

En el mundo real, hay objetos que pueden ser entidades o fenómenos, acerca de los cuales se desea reunir información. Cada percepción usualmente se compone de muchos datos interrelacionados. Un dato es llamado un dato elemental.

En la representación del mundo real se debe ser capaz de plasmar fielmente las percepciones, aceptando que el orden en que se muestren reflejan fielmente los datos más elementales. Sin embargo, en el mundo real no hay mecanismos de representación asociados con datos elementales [Tsichritzis, 1982].

De esta manera, únicamente las propiedades que deben estar asociadas con la existencia de un objeto pueden cambiar sin afectarse. Esta última observación nace naturalmente del concepto "tiempo", como un aspecto importante de algunos datos.

Introduciendo el tiempo en un modelo de datos infológico, uno mismo puede retener información acerca de las propiedades o relaciones que no están validando su duración.

Usando los conceptos básicos de objetos, propiedades, relaciones y el tiempo para estructurar la representación de datos. Se pueden combinar estos conceptos básicos para derivar otros, que están presentes en otros modelos de datos. Cuando sea posible, se establecerá la correspondencia en las construcciones, cuando los conceptos antes mencionados aparezcan en el modelo de datos infológico [Tsichritzis, 1982].

Un dato elemental fue vagamente definido antes como un objeto, una propiedad de un objeto o del tiempo, se puede ahora definir formalmente una representación para un dato elemental como una tupla (x, y, z) donde:

x, es una tupla de objeto $O_1,...,O_n$,
y, es una propiedad de una relación entre estos objetos y
z, es el tiempo

Esta estructura básica para la representación de un dato elemental, es llamada una *constelación elemental.*

Un objeto atómico es un objeto que no puede ser descompuesto dentro de otros objetos. En el caso de que cuando x, es un objeto atómico y/o es una propiedad, se le denomina una constelación elemental de un tipo de propiedad. Esta construcción puede representar la situación entre otros modelos de datos, cuando un atributo toma un valor en un tiempo en particular. Una diferencia importante entre estas unidades básicas de información, y un atributo de una instancia en otros modelos de datos es la mención explícita del tiempo [Tsichritzis, 1982].

Las constelaciones pueden presentar situaciones que son difíciles de manejar con los modelos de datos convencionales. Para una instancia, una propiedad asociada con una relación entre objetos puede ser manejada muy fácilmente como una constelación.

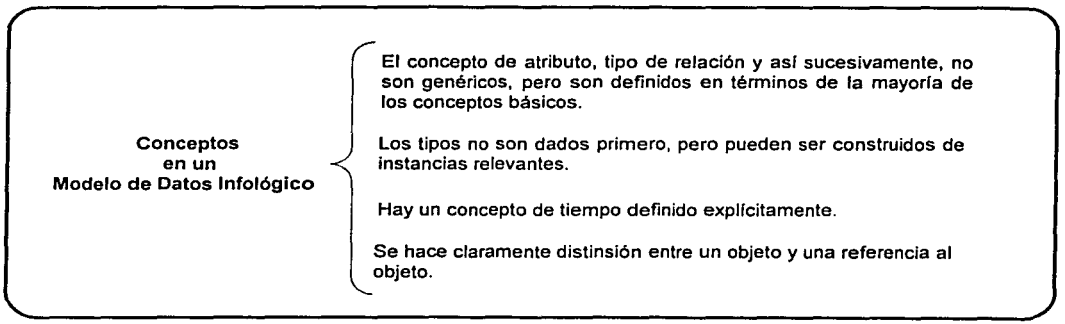
En la mayoría de los modelos de datos una relación entre objetos no pueden tener alguna propiedad asociada, solamente los objetos pueden tener propiedades (atributos).

Un modelo de datos infológico es el vivo ejemplo de un tipo de modelo de datos. Un objeto en el modelo de datos, alguna relación o propiedad como el tiempo, puede ser definido como *dato.* No hay una definición a priori de tipos de objetos, propiedades o relaciones para que todos los datos tengan que ajustarse. La única representación de ellos aparece en el campo infológico. El mundo real es como este; la representación del mundo real es necesaria para poderlo entender [Tsichritzis, 1982].

En el modelo de datos infológico, un objeto puede existir independientemente de alguna propiedad o relación perteneciente a esta. Sin embargo, en muchos modelos de datos la existencia de un objeto esta relacionado con una llave del atributo que no puede ser nula. Un objeto en un modelo de datos no existe sin un valor de esta llave. Esto no es una ventaja, para el vinculo de la existencia de un objeto con una propiedad en particular o relación.

No hay un esquema definido con anterioridad. Sin embargo, los tipos de objetos pueden ser definidos, si hay necesidad, por grupos de objetos apropiados a sus propiedades.

Hay sin embargo, algunas diferencias importantes entre estos conceptos como definiciones en un modelo de datos infológico y en otros modelos de datos.



o

Figura 3.6 Diferencias del Modelo de Datos Infológico con otros Modelos de Datos

En la mayoría de los modelos de datos un objeto es definido por una propiedad. Una referencia puede ser explícitamente referida a un objeto, esta puede ser implícita, refiriéndose a un objeto. Alternativamente una referencia puede estar de acuerdo a un tipo de dato infológico. Para una instancia, un objeto, propiedad, relación, tiempo de referencia y objeto de referencia, el concepto de referencia es un dato infológico diferente del dato infológico actual y diferente también de cualquier representación de datos infológicos del mundo real. Si se prueba la existencia de algún objeto y/o una referencia a éste entonces existe una diferencia para una instancia. La misma referencia puede llevar una ventaja de datos infológicos diferentes dependiendo del contexto de referencia [Tsichritzis, 1982].

Los modelos de datos tradicionales evitan la separación de objetos y de referencias de objetos de varias maneras:

- Primero, hay en cada sitio un objeto dentro de un tipo seguro.
- Las propiedades del tipo de entidades están definidas por los valores de los atributos.
- Una entidad es identificada por un conjunto de propiedades.
- Relaciones entre entidades están únicamente permitidas de acuerdo al tipo de relación definida en el esquema.
- Cada instancia de la relación es definida por las entidades que participan en esta.
- Todos los atributos, tipos de entidades y tipos de relaciones son llamadas únicas en un esquema.

Bajo estas condiciones no hay casi necesidad de distinguir objetos de nombres de objetos.

Un problema es que en ocasiones se necesita tener nombres temporales para referenciar objetos. Muchos modelos de datos resuelven este problema usando indicadores aceptados.

Un indicador aceptado y el valor sirve como una referencia temporal del objeto. Otro problema más serio surge cuando hay necesidad de cambiar la propiedad de identificación de un objeto. Para evitar estos problemas introducir propiedades artificiales como identificadores únicos [Tsichritzis, 1982].

Aunque estas propiedades artificiales sirven como referencia para identificar objetos, hay limitantes de muchas maneras. Ellas se comportan bien como variables globales y estas no tienen ningún concepto de contexto. Como resultado ellas no ofrecen un entorno flexible, ya que para un objeto usan muchas referencias hacia éste o también obligan a referenciar objetos en tiempos diferentes.

Las referencias a datos infológicos pueden ser usadas para representar constelaciones. Esta representación de una constelación es referida como un mensaje. Un *mensaje elemental* es una terna (x, y, z) donde :

x, es una tupla de referencia de objetos únicos
y, es una propiedad única o relación de referencia
z, es el tiempo de referencia

En un *mensaje de referencia elemental* a una constelación elemental única de cualquier tipo de propiedad o tipo de relación, un mensaje es considerado incompleto si una de estas referencias no es única. Un mensaje incompleto puede ser considerado como una consulta. Los mensajes similares pueden ser agrupados dentro de un tipo. Para un par (x, y), donde:

x, es una referencia a un grupo de objetos
y, es una referencia a un atributo

Se debe notar la diferencia entre un tipo de constelación y un tipo de mensaje. En el primer caso se revuelve con los mismos objetos y en el segundo caso con referencia a los objetos [Tsichritzis, 1982].

El campo infológico es conceptual, no es una descripción física del mundo real. Se puede naturalmente ver mensajes elementales como datos, pero se debe primero poder representarlos físicamente. Esta es la función de mapearlos dentro de datos lógicos.

Un mensaje elemental en el campo infológico es representado en un registro elemental, en el campo de datos lógico [Tsichritzis, 1982].

Una *colección de registros elementales* de un mismo tipo forma un archivo elemental.

Un *archivo elemental* puede también ser considerado cómo la realización física de un tipo de mensaje elemental.

Los registros elementales pueden ser moldeados para formar registros compuestos.

Los *registros compuestos* son realizaciones físicas de mensajes compuestos. Una notación es usada por algunos mapeos sobre una realización física.
Una *restricción estética* en el mapeo de registros es que los registros elementales se parecen estrechamente a la estructura de mensajes elementales, además de satisfacer la restricción de la arquitectura del sistema involucrado.

El *mapeo* es definido por si mismo en términos de mensajes elementales, que son en este caso representados por registros elementales. De esta manera el mapeo de campo infológico al campo de datos lógico, puede ser representado por las capacidades de descripción de un modelo de datos infológico [Tsichritzis, 1982].

3.3.1.2.2 RESTRICCIONES

El modelo de datos infológico permite la representación de restricciones sintácticas. Una restricción sintáctica define el conjunto de constelaciones que potencialmente pueden ser válidas. Todos los mensajes elementales que posiblemente pueden ser especificados corresponden a estas constelaciones válidas [Tsichritzis, 1982].

- El conjunto de restricciones sintácticas permite limitar mensajes para tener algunas propiedades deseadas o para ajustarse a un tipo. Si se necesitan tipos de datos infológicos, se necesita especificar que limitantes en los mensajes son admisibles. En un modelo de datos infológico se pueden especificar las restricciones sintácticas que limiten este tipo de mensajes.

- Entre todos los posibles mensajes que corresponden a constelaciones válidas, algunos de los mensajes son especificados como verdaderos. Estos mensajes verdaderos en turno especifican constelaciones verdaderas. Sin embargo, el mapeo

de mensajes para constelaciones no es sencillo, dependen del contexto. Un mensaje tiene que ser referenciado y asociado con objetos, la constelación se válida antes que se tenga un incremento del conocimiento en la BD. Este punto es muy crítico para un modelo de datos infológico.

- Conocer parte de la información no significa nada, a menos que este especificada dentro de un contexto seguro. Estos requerimientos prueban un reflejo del comportamiento de la gente. La gente puede solamente entender algo si este se especifica dentro de un contexto particular. El concepto de un extenso contexto complica la especificación de las restricciones.

- En la mayoría de los modelos de datos hay una noción de consistencia. Esta asume que un dato no debe ser permitido para ser especificado dentro de la BD. Este también asume que si un dato no ha sido especificado, este debe ser considerado falso. Esta noción es llamada suposición del mundo cerrado. Bajo esta suposición, las restricciones intentan delinear categorías generales de datos, que son falsos y deben permitir ser especificadas.

En algún momento las restricciones intentan cierta consistencia y checan en cuales pueden ser consideradas. De otra manera, en un modelo de datos infológico las restricciones dividen datos dentro de tres categorías:

- verdadero
- significativo
- falso

Esta flexibilidad en la especificación de una restricción introduce muchas complicaciones. Puesto que cada mensaje es especificado con respecto al contexto, para dos diferentes contextos el mismo mensaje puede ser mapeado para diferentes constelaciones. Otra vez, ambos datos pueden ser especificados en un modelo de datos infológico. En suma, el concepto de distancia infológica entre mensajes, permite al usuario especificar mensajes que no son verdaderos, pero están "cerrados" para ser verdaderos.

Para explicar la diferencia entre restricciones de un modelo de datos regular o en un modelo de datos infológico, se usa cómo analogía la diferencia entre opciones verdaderas. En la mayoría de los modelos de datos, algunos mensajes tienen una verdad o falsedad universal. Únicamente los mensajes verdaderos están en la BD, pero si el mensaje no esta almacenado, este es falso. Todas las comparaciones en los mensajes son hechas al tiempo que el mensaje esta disponible. Un modelo de datos permite al usuario especificar uno o más filtros que refuercen las restricciones durante la interpretación de las constelaciones. De esta manera, la información debe ser interpretada en un contexto y el usuario puede estar protegido por los filtros de inconsistencia o información no verdadera. Esto es muy diferente del enfoque del manejo de restricciones que en la mayoría de los otros modelos de datos. Las restricciones son puestas para prevenir al usuario de ingresar errores o inconsistencias en la información.

En un modelo de datos infológico los errores o inconsistencias de la información pueden todavía ser permitidos, y mapeados como constelaciones válidas dentro de un contexto [Tsichritzis, 1982].

3.3.1.3 MODELO FUNCIONAL DE DATOS

El modelo funcional define los procesamientos que un objeto realiza "como se calculan las salidas a partir de los datos de entrada". Especifica las operaciones que se realizan dentro de los modelos dinámico y de datos. La importancia principal del modelo funcional para aplicaciones de bases de datos esta en mostrar la navegación del modelo de datos [Lascano, 2000].

Además permite representar las operaciones que son ejecutadas por el sistema y no son identificados los métodos para cada clase de objetos.

El modelo funcional muestra como los valores de salida en un cálculo son derivados a partir de los valores de entrada, sin importar el orden en el cual los valores son calculados. Así mismo, este modelo consiste de múltiples diagramas de flujo de datos, los cuales muestran el flujo de los valores externos de entrada, a través de las operaciones y almacenamientos de datos internos, hasta las salidas externas. También incluye restricciones entre los valores en el modelo de objetos.

Describe exactamente lo que el sistema hace y tiende a perder la perspectiva general que el modelo conceptual ofrece. El nivel de detalle en un modelo funcional puede ser muy bajo, mostrar las entradas y las salidas de todo el sistema. O bien, muy alto y mostrar los componentes individuales y sus entradas y salidas correspondientes.

Los modelos funcionales muestran la manera como están unidas las partes del sistema y como interactúan con el resto del mundo. Los puntos de contacto, donde las cosas suceden, son llamados *Interfaces*. Las interfaces están donde una pieza de software interactúa con otra, y donde el hardware y el software se conjuntan [Rodriguez, 2000].

Especifica los resultados de un cálculo sin mencionar como o cuando son calculados, indica el significado de las operaciones en el modelo de objetos y las acciones en el modelo dinámico, así como cualquier restricción en el modelo de objetos.

Los programas no interactivos, como los compiladores, tienen un modelo dinámico trivial; el modelo funcional es el principal modelo para tales programas, las bases de datos usualmente tienen un modelo funcional trivial, debido a que su propósito es almacenar y organizar datos, no el transformarlos.

Los principales objetivos de crear un Modelo Funcional en una organización, son:

- Proveer un modelo preciso de las necesidades funcionales de la organización, los cuales actuarán como marco referencial para el desarrollo o manutención de los sistemas.

- Proveer un modelo que es independiente de cualquier mecanismo o método de procesamiento, permitiendo la toma de decisiones objetivas acerca de técnicas de implementación y coexistencia con sistemas existentes.

Todas las clases, relaciones y atributos en el modelo funcional deben constar en el modelo de objetos del sistema. Todos los otros conceptos deben estar definidos en el diccionario de datos o en alguna otra fuente referenciada [Lascano, 2000].

El modelo funcional debe preservar obligatoriamente los invariantes (valores de atributos que no cambian durante el ciclo de vida del sistema).

Consistencia entre los Modelos

- La frontera del modelo de objetos del sistema debe ser coherente con el modelo de ciclo de vida del sistema.
- Todas las operaciones del sistema que aparecen en el modelo dinámico deben tener una especificación ("esquema") en el modelo funcional.
- Todos los identificadores en todos los modelos deben tener entradas en el DD.
- Las salidas o eventos en el modelo de ciclo de vida del modelo dinámico y las salidas de las operaciones del modelo funcional deben ser las mismas.
- El modelo funcional debe preservar obligatoriamente los invariantes (valores de atributos que no cambian durante el ciclo de vida del sistema).

3.3.1.4 MODELO ENTIDAD/RELACIÓN

El modelo Entidad Relación E/R fue propuesto por Peter P. Chen en sus dos artículos ya históricos, Chen (1976) y Chen (1977). Según Chen (1976), " el modelo E/R puede ser usado como una base para una vista unificada de los datos", adoptando "el enfoque más natural del mundo real que consiste en *entidades* y *relaciones*" [De Miguel, 1993].

Posteriormente otros muchos autores [Paul (1980), Teory et al.(1986), Ferg (1984), Shang y Shixuan (1984), Elmasri et al. (1985), etc.] han investigado y escrito sobre el modelo, proponiendo importantes aportaciones, por lo que realmente no se puede considerar que exista un único modelo E/R, sino más bien lo que podríamos llamar una *familia de modelos* [De Miguel, 1993].

El modelo E/R, como su nombre lo indica, esta centrado en dos conceptos fundamentales: el de *entidad* y el de *relación.* Se entiende por entidad cualquier objeto(real o abstracto) sobre el cual queremos tener información de la base de datos y que tiene existencia por si mismo.

Relación es la asociación o correspondencia entre entidades [conexión entre entidades, según Casanova Amaral de Sa (1984)] [De Miguel, 1993].

Desde que fue propuesto, el modelo E/R ha tenido una gran difusión y ha despertado un enorme interés en la comunidad dedicada a las bases de datos, prueba de ello son los innumerables congresos dedicados al tema, entre los que cabe destacar las conferencias Internacionales sobre el Enfoque E/R que se celebran anualmente en distintos lugares del mundo. En esas reuniones se recopilan y presentan los últimos estudios realizados sobre el modelo.

3.3.1.4.1 ESTÁTICA DEL MODELO E/R

En le modelo E/R se pude distinguir como elementos fundamentales las entidades, los atributos y las interrelaciones, además del *conjunto de valores "value set"* análogo al concepto de dominio.

Entidad. Se pude definir la entidad como aquel objeto (real o abstracto) acerca del cual queremos almacenar información en la base de datos. Según ANSI (1977), es como "una persona, lugar cosa, concepto o suceso, real o abstracto, de interés para la empresa" [De Miguel, 1993].

La representación gráfica de este objeto es un rectángulo etiquetado con el nombre del tipo de entidad. Tardieu *et al.* (1979) propone tres reglas generales que debe cumplir una entidad [De Miguel, 1993]:

- Tener existencia propia.
- Cada ocurrencia de un tipo de entidad debe poder distinguirse de los demás.
- Todas las ocurrencias de un tipo de entidad deben tener los mismos tipos de características (atributos).

Existen dos clases de entidades: *regulares o fuertes*, que son aquellas que tienen existencia por si mismas, sin depender de la existencia de alguna otra entidad, y *débiles*, cuya existencia depende de la existencia previa de otra entidad.

- Si la entidad débil puede ser identificada sin necesidad de identificar previamente la entidad de cuya existencia depende, se dice, que la entidad débil lo es por *existencia* únicamente.
- Si la entidad débil no puede ser identificada independientemente, sino que previamente es necesario identificar la entidad de cuya existencia depende, se dice, que la entidad débil es por identificación.

Por *extensión* se considera a una relación entre entidades débiles, pero también se le dice débil por existencia o por identificación según sea el tipo de debilidad de las entidades que intervengan en la relación. Una idea para el reconocimiento de entidades débiles, es pensar qué sucede cuando se borra una instancia concreta de la entidad fuerte [Pereda, 2001].

Relación. Se entiende por relación aquella asociación o correspondencia existente entre entidades. Denominamos tipo de relación a la estructura genérica del conjunto de relaciones entre dos o más tipos de entidades.

Elementos de un tipo de relación. En un tipo de relación existen los siguientes elementos [De Miguel, 1993]:

- *Nombre:* Como todo un objeto del modelo E/R, cada tipo de relación tiene un nombre que lo distingue unívocamente del resto y mediante el cual ha de ser referenciado.
- *Grado:* Es el número de tipos de entidad que participan en un tipo de relación. Puede existir un tipo de relación que asocie más de dos tipos de entidades (grado *n*).

Cuando se presenta un tipo de relación de grado *n*, hay que tener en cuenta que a veces no es propiamente de tal grado, ya que puede descomponerse en varios tipos de relación de grado 2. Sin embargo, otras veces no es posible tal descomposición, ya que la semántica recogida en una y otra solución no es la misma.

Tipo de correspondencia o Cardinalidad. Es el número máximo ocurrencias o instancias de una entidad con otra. Se presenta mediante una pareja de datos, en minúsculas, de la forma (cardinalidad mínima, cardinalidad máxima), asociada a cada una de las entidades que intervienen en la relación. Son posibles las siguientes cardinalidades: (0,1), (1,1), (0,n), (m,n) [De Miguel, 1993].

Tipo de relación. Se define tomando los máximos de las cardinalidades que intervienen en cada relación. Para representarlo gráficamente hay cuatro posibles etiquetas y son:

- Una a una (1,1). En este tipo de relación, una vez fijado un elemento de una entidad se conoce la otra. Ejemplo: nación y capital.
- Una a muchas (1,M). Ejemplo: cliente y pedidos.
- Muchas a una (M,1). Simetría respecto al tipo anterior según el punto de vista de una u otra entidad.
- Muchas a muchas (M,M). Ejemplo: personas y viviendas

Papel o Rol. Es la función que cada uno de los tipos de entidad realiza en el tipo de relación.

Dominio y Valor. Las distintas propiedades o características de un tipo de entidad o de relación toman VALORES para cada instancia de éstas. El conjunto de posibles valores que puede tomar una cierta característica se denomina DOMINIO.

Atributo. Cada una de las propiedades o características que tiene un tipo de entidad o un tipo de relación se denomina ATRIBUTO. Los atributos toman valores de uno o varios dominios. Por tanto, se dice que el atributo le da una determinada interpretación al dominio en el contexto de un tipo de entidad o de un tipo de relación [De Miguel, 1993].

La representación gráfica de un atributo consiste en cualificar con su nombre el arco que une el dominio con el tipo de entidad o de relación. Sin embargo, para simplificar la representación gráfica, en muchos casos (siempre conocida con el nombre del dominio con el atributo) será suficiente con el círculo u óvalo del dominio, eliminando el nombre del atributo. En el esquema conceptual resultante del modelado sólo se especifican los atributos más significativos.

Entre todos los atributos de un tipo de entidad se debe elegir uno o varios, que identifiquen unívocamente cada una de las instancias de este tipo de entidad. Este atributo o conjunto de atributos se denomina *atributo identificador principal* (AIP), y los atributos que lo componen deben ser mínimos en el sentido de que la eliminación de cualquiera de ellos le haría perder su carácter identificador.

Puede ocurrir que exista más de un conjunto de atributos que verifiquen la condición de ser identificador unívoco y mínimo de cada ocurrencia del tipo de entidad, por lo que se denomina a cada uno de ellos *Atributo identificador candidato* (AIC). Se elige uno como AIP y el resto serán *Atributos identificadores alternativos* (AIA). La representación gráfica de estos atributos queda reflejada en la figura 3.7. Para los AIA se reserva una zona del circulo para poner un número identificador que indica a qué clave alternativa pertenece dicho atributo (en caso de que exista más de un AIA) [De Miguel, 1993].

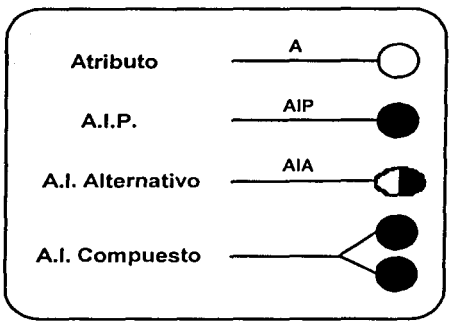


Figura 3.7 Representación Gráfica de Atributos

3.3.1.4.2 TIPOS ESPECIALES DE RELACIÓN

- Relación R*eflexiva* o R*ecursiva.* Relaciona una entidad consigo misma. Ejemplo: empleados que pueden ser jefes de otros empleados.

- *Dos Relaciones entre dos mismas Entidades.* Muy útil en el caso de necesitar almacenar información histórica completa. Ejemplo: proyectos en los que trabaja actualmente un empleado y proyectos en los que ha trabajado anteriormente.

- *Relación Ternaria.* Asociación de tres entidades. La forma de hallar el tipo de relación en las relaciones ternarias, es fijar una combinación de elementos de dos de los extremos de la relación y obtener lógicamente las cardinalidades mínima y máxima en el otro extremo libre [Pereda, 2001]. Ejemplo: el título libro (en un año concreto, con un ISBN y con un determinado número de páginas en la edición). Para determinar las cardinalidades hay que preguntar por:

  - Cuántos autores pueden tener un determinado libro publicado en una determinada editorial (cardinalidad en el extremo de la entidad autor).
  - Cuentos libros puede tener un determinado autor publicados en una determinada editorial (cardinalidad en el extremo de la entidad libro).
  - En cuántas editoriales puede determinado autor publicar un mismo libro (cardinalidad en el extremo de la entidad editorial).

- *Relación de Especialización (ES-UN).* Tipificación de una entidad en subtipos, en número finito y conocido. Cada subtipo puede poseer atributos propios y/o también heredan los atributos que pudiera tener la entidad supertipo o también llamada entidad general. Este tipo de relación puede clasificarse de dos manera distintas [Pereda, 2001]:

  - La primera es según si una instancia o elemento concreto de la entidad puede ser de más de un subtipo a la vez. En caso afirmativo se dice que la relación es *inclusiva* o *con solapamiento*, mientras que en caso contrario será *exclusiva* o *sin solapamientos.*

  - La segunda clasificación se basa en si es obligatoriamente cada instancia o elemento concreto debe ser obligatoriamente de alguno de los subtipos especificados, es decir, si no pueden existir elementos de la entidad que no pertenezcan a ninguno de los subtipos. Si es así, la relación se dice *total* y en caso contrario *parcial.* La situación más común en una relación de especialización es que sea exclusiva y total. Ejemplos:

    Una entidad persona tiene los subtipos hombre y mujer. Una misma persona no puede ser hombre y mujer a la vez por lo que la relación es exclusiva. No puede existir una persona que no sea hombre ni mujer, por lo que también es total.

    La entidad que representa a un universitario tiene los subtipos profesor y estudiante. Un mismo universitario puede ser ambas cosas a la vez, por lo que la relación es inclusiva. No puede existir un universitario que no sea ni profesor ni estudiante, por lo que también es total.

    Se puede expresar mediante una relación de especialización el que una función matemática tiene asociados los subtipos continua y derivable. La relación es inclusiva y parcial, pues una misma función puede ser ambas cosas a la vez y porque existen funciones que no son continuas ni derivables.

Se supone una entidad A que se especializa es dos subtipos A1 y A2. La identificación del tipo de relación (exclusiva, total, etc) puede hacerse atendiendo a la tabla de verdad que se muestra en la figura 3.8 [Pereda, 2001].

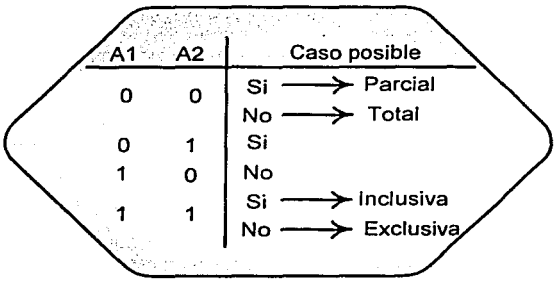


Figura 3.8 Tabla de Verdad

La cardinalidad en las relaciones de especialización es siempre (1,1) en el extremo de la entidad que se especializa en subtipos y (0,1) en el extremo de los subtipos si la relación es exclusiva o ({0,1},1) si es inclusiva. Una relación de especialización puede fácilmente convertirse en total, añadiendo un nuevo subtipo "otros".

3.3.1.4.3 RESTRICCIONES

El modelo E/R (MER) no tiene inherentes, y por lo que respecta a las restricciones de usuario, el MER permite definir [De Miguel, 1993]:

- Restricciones sobre valores, esto es, las que delimitan los valores que pueden corresponder a un cierto objeto del modelado (entidad, dominio, atributo, etc...).

- Restricciones sobre número de ocurrencias, son aquellas que delimitan el número de objetos que pueden intervenir en un tipo de interrelación, ya sea ocurrencias de entidad, tipos de entidad, etc.

El modelo E/R no está preparado para representar gráficamente todo este tipo de restricciones y menos aún las restricciones dinámicas, que no pueden expresarse en este modelo.

3.3.1.4.4 DINÁMICA DEL MODELO E/R

Chen, en su artículo donde presentaba el MER, apenas se refirió a su parte dinámica, pero en varios trabajos posteriores se han propuesto lenguajes que operan en el nivel conceptual basándose en el MER [De Miguel, 1993].

Estos lenguajes permiten formular consultas a la base de datos con sentencias que se parecen al lenguaje natural y son sencillas de formular. Entre los lenguajes destacan el CABLE (Chain Based Language) [Shoshani (1980)] y el CLEAR (Conceptual Language for Entities And Relationships) [Poonen (1978)].

3.3.1.4.2 CONTROL DE REDUNDANCIA EN LOS ESQUEMAS E/R

Una vez construido un esquema E/R, hay que analizar si se presentan redundancias, ya que pueden acarrear problemas a la hora de instrumentar a la base de datos.

Además de la existencia de atributos redundantes, como los que se derivan de otros mediante algún cálculo y que deben ser eliminados del esquema E/R o marcarse cómo redundantes, hay que estudiar determinadamente los ciclos en el diagrama E/R, ya que pueden indicar la existencia de interrelaciones redundantes.

3.3.1.5 MODELO ORIENTADO A OBJETOS

El Modelo Orientado a Objetos de basa en encapsular código y datos en una única unidad, llamada *objeto* [Varas, 1999].

En general, un objeto tiene asociado:

- Un conjunto de *atributos* que contienen datos acerca del objeto. A su vez, cada valor de un atributo es un objeto.
- Un conjunto de mensajes a los que el objeto responde.
- Un conjunto (puede ser unitario) de *métodos* que es un procedimiento o trozo de código para implementar la respuesta a cada mensaje. Un método devuelve un valor (otro objeto) como respuesta al mensaje.

Normalmente en la realidad al modelar, existen muchos objetos similares. Por similar se entiende que se tiene una naturaleza parecida, por lo que son candidatos a poseer atributos, métodos y responder a mensajes comunes en un contexto orientado a objetos.

Para el caso de los objetos similares, sería un trabajo inútil definir cada uno de ellos por separado, por lo tanto, se agrupan para que conformen una *clase*. A cada uno de estos objetos se le llama *instancia* de su clase. Todos los objetos de una clase comparten una definición común, aunque difieran en los valores asignados a los atributos. Un esquema orientado a objetos, normalmente requiere un gran número de clases. Una vez definida la clase, los atributos pueden reutilizarse al crear nuevas instancias de la clase.

Los conceptos fundamentales que llevan a un diseño de alta calidad, son igualmente aplicables a sistemas desarrollados, usando métodos convencionales y orientados a objetos. Por esta razón, un modelo orientado a objetos de software de computadora debe exhibir las abstracciones de datos, y los procedimientos que conducen a una modularidad eficaz [Pressman, 1998].

Una clase es un concepto orientado a objetos que encapsula las abstracciones de datos y procedimientos que se requieren para describir el contenido y comportamiento de alguna entidad del mundo real. Taylor usa la notación que se muestra en la figura 3.9, para describir una clase (y objetos derivados de una clase) [Pressman, 1998].

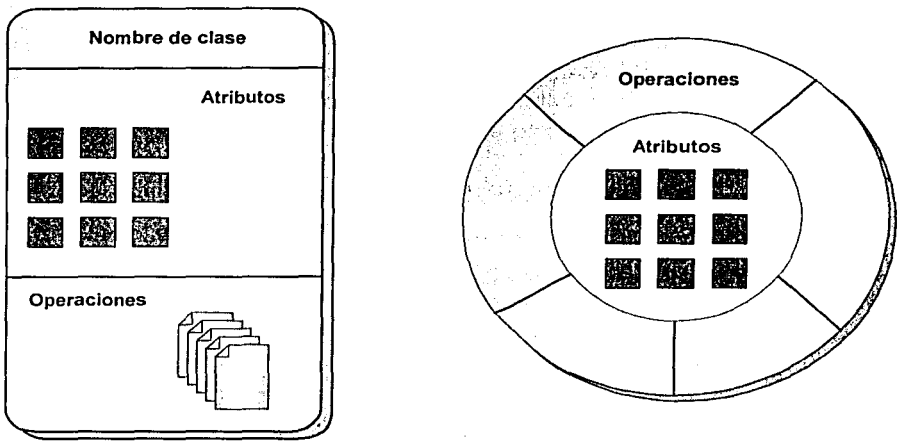


Figura 3.9 Representación de una Clase Orientada a Objetos

Las abstracciones de datos (atributos) que describen la clase están encerradas por una "muralla" de abstracciones formadas por procedimientos (llamadas operaciones, métodos o servicios) capaces de manipular los datos de alguna manera. La única forma de alcanzar los atributos (y operar sobre ellos) es ir a través de alguno de los métodos que forman la muralla. Por lo tanto, la clase encapsula datos, dentro de la muralla, y el proceso que manipula los datos (los métodos que componen la muralla). Esto posibilita el ocultamiento de información y reduce el impacto de efectos colaterales asociados a cambios. Como estos métodos tienden a manipular un número limitado de atributos, esto es con una alta cohesión, y como la comunicación ocurre sólo a través de los métodos que encierra la muralla, la clase tiende a un acoplamiento con otros elementos del sistema [Pressman, 1998].

Por definición, todos los objetos que existen dentro de una clase heredan sus atributos y las operaciones disponibles para la manipulación de los atributos. Una *superclase* es una colección de clases y una *subclase* es una instancia de una clase.

Estas definiciones implican la existencia de una *jerarquía de clases* en la cual los atributos y operaciones de la superclase son heredados por subclases que pueden añadir, cada una de ellas, atributos privados y métodos [Pressman, 1998].

Una clase incluye:

- Un atributo con valores en un conjunto cuyo valor es el conjunto de todos los objetos que son instancias de la clase.

- La implementación de un método para el mensaje nuevo, el cual crea una nueva instancia de la clase.

Los mensajes son el medio a través del cual los objetos interactúan. Usando la terminología introducida anteriormente, un mensaje estimula la ocurrencia de cierto comportamiento en el objeto receptor. La interacción entre objetos se ilustra esquemáticamente en la Figura 3.10. Una operación dentro de un objeto *emisor* genera un mensaje de la forma [Pressman, 1998]:

mensaje: [destino, operación, parámetros]

donde *destino* define el objeto *receptor* el cual es estimulado por el mensaje, *operación* se refiere al método que recibe el mensaje y *parámetros* proporciona información requerida para el éxito de la operación.

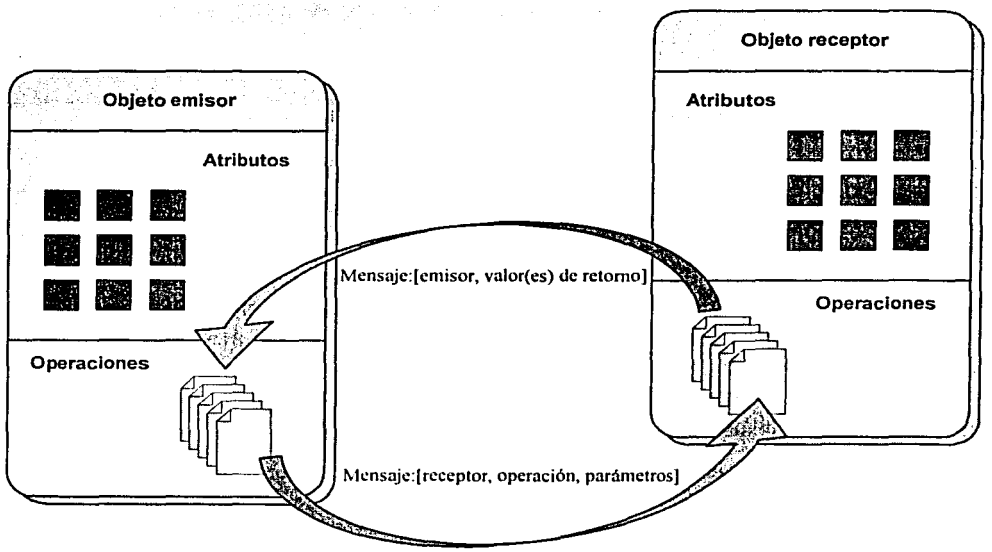


Figura 3.10 Paso de mensajes entre objetos

Este comportamiento se realiza cuando se ejecuta una operación. El paso de mensajes mantiene comunicado un sistema orientado a objetos. Los mensajes proporcionan una visión interna del comportamiento de objetos individuales, y del sistema orientado a objetos como un todo.

Las clases orientadas a objetos y los objetos derivados de ella encapsulan los datos, y las operaciones que trabajan sobre estos en un único paquete. Esto proporciona un número importante de beneficios:

- Los detalles de implantación interna de datos y procedimientos están ocultos al mundo exterior (ocultamiento de la información). Esto reduce la propagación de efectos colaterales cuando ocurren cambios.

- Las estructuras de datos y las operaciones que las manipulan están mezcladas en una entidad sencilla: la clase. Esto facilita la reutilización de componentes.

- Las interfaces entre objetos encapsulados están simplificadas. Un objeto que envía un mensaje no se tiene que preocupar, de los detalles de las estructuras de datos internas, en el objeto receptor. Por tanto, se simplifica la interacción y el acoplamiento del sistema tiende a reducirse.

Aunque la estructura y terminología introducida diferencian los sistemas orientado a objetos a partir de sus componentes convencionales, tres características de sistemas orientados a objetos los hacen únicos [Pressman, 1998].

La herencia es una de las diferencias clave entre sistemas convencionales y sistemas orientados a objetos. Una subclase Y hereda todos los atributos y operaciones asociadas con su superclase X. Esto significa que todas las estructuras de datos y algoritmos originalmente diseñados e implementados para X están inmediatamente disponibles para Y (no es necesario más trabajo extra).

Es importante destacar que en cada nivel de la jerarquía de clases, pueden añadirse nuevos atributos y operaciones a aquellos que han sido heredados de niveles superiores de la jerarquía. De hecho, cada vez que se deba crear una nueva clase, se tienen varias opciones [Pressman, 1998]:

- La clase puede diseñarse y construirse de la nada. Esto es, no se usa la herencia.

- La jerarquía de clases puede ser rastreada para determinar si una clase que ascendente, contiene la mayoría de los atributos y operaciones requeridas.

- La nueva clase hereda de su clase ascendente, y pueden añadirse los elementos requeridos.

- La jerarquía de clases puede reestructurarse, de tal manera que los atributos y operaciones requeridos puedan heredarse a la nueva clase.

- Las características de una clase existente pueden sobrescribirse, y se pueden implementar versiones privadas de atributos u operaciones para la nueva clase.

En algunos casos, es tentador heredar algunos atributos y operaciones de una clase y otros de otra clase. Esto se llama *herencia múltiple* y es controvertida. En general, la herencia múltiple complica la jerarquía de clases y crea problemas potenciales en el control de la configuración. Como las secuencias de herencia múltiple son más difíciles de seguir, los cambios en la definición de una clase que reside en la parte superior de la jerarquía. Pueden tener impactos no deseados originalmente en las clases definidas, en zonas inferiores de la arquitectura [Pressman, 1998].

El *polimorfismo* es una característica que reduce en gran medida el esfuerzo necesario para extender un sistema orientado a objetos; este permite que un número de operaciones diferentes tengan el mismo nombre. Esto reduce el acoplamiento entre objetos, haciendo a cada uno más independiente.

3.3.2 MODELOS DE DATOS LÓGICOS BASADOS EN REGISTROS

Los modelos lógicos basados en registros se encuentran soportados por los DBMS y están orientados a describir los datos del nivel lógico para el DBMS (de ahí que también reciban el nombre de modelos de bases de datos). Por lo que sus conceptos son propios de cada DBMS (tablas o relaciones en el caso del modelo Relacional, modelo de Red y árboles en el Jerárquico, etc.). Estos tres modelos de datos convencionales difieren, esencialmente en el modo de representar las asociaciones entre entidades y en la forma de acceso a la base de datos [De Miguel, 1993].

Los modelos lógicos basados en registros, se pueden considerar como interfaz entre el informático y la computadora. Apoyando al diseñador en etapas posteriores del proceso de diseño, ya que por ejemplo, la sencillez es una característica del modelo relacional que lo hace muy asequible a los usuarios; esta sencillez es la estructura que ponen de manifiesto en las operaciones de consulta y de actualización. Este modelo ha tenido, hasta el momento y en ciertos aspectos, un menor contenido semántico que el modelo de Red, principalmente porque no permite distinguir entre objetos y asociaciones entre ellos; ambos se presentan en el modelo Relacional mediante una estructura única: la relación o tabla. El último estándar de ISO, el SQL92 [ISO(1992)], ha introducido importantes aspectos semánticos de los que carecía el anterior, y el que se está elaborando actualmente, el SQL3, ofrecerá interesantes mejoras semánticas [De Miguel, 1993].

En cuanto al tema de la independencia físico/lógica, el modelo de Red. Aún más el Jerárquico, presenta una correspondencia directa entre las relaciones lógicas y los caminos de acceso físico, lo que dificulta la independencia entre los niveles lógico y físico.

En el modelo Relacional, sin embargo, el acceso a los datos se realiza en función de las propiedades de éstos, y el usuario no conoce los caminos de acceso. Puesto que, es el sistema el que se ocupa de seleccionar el camino físico optimizando los recursos y el tiempo de respuesta.

Cabe destacar que el rendimiento ha sido un argumento fundamental para los que defendían la supremacía de los sistemas tipo red. Sin embargo, CODD (1990) y DATE (1986) siempre ha sostenido que no había razones objetivas para que los sistemas relacionales no fuesen tan eficientes como los basados en otros modelos; el rendimiento de las actuales versiones de algunos Sistemas Manejadores de Bases de Datos Relacional (DBMSR) parece que viene a dar la razón a estos autores [De Miguel, 1993].

En la década de los ochenta los DBMSR tuvieron una gran difusión, la tendencia que durante esta década está llevando, a distintas *extensiones* del modelo Relacional básico, como multimedia, gestión de objetos y conocimiento, etc.

3.3.2.1 MODELO RELACIONAL

La introducción por Codd, muy a finales de los años sesenta, de la teoría de las relaciones en el campo de las bases de datos supuso un importante paso a la investigación de los DBMS, suministrando un sólido fundamento teórico para el desarrollo, dentro de este enfoque relacional, de nuevos productos [De Miguel, 1993].

El documento de Codd propone un modelo de datos basado en la teoría de las relaciones, en donde los datos se encuentran lógicamente en forma de relaciones –tablas-, siendo un objetivo fundamental del modelo, enfocar las características de un buen diseño [Hawryszkiewycz, 1994].

Además de la independencia de esta estructura lógica respecto al modelo de almacenamiento y a otras características de tipo físico. En palabras de Codd (1970), "la vista relacional de los datos...parece ser superior al modelo en grafos o en red...Proporciona un medio para describir datos con su estructura natural únicamente, es decir, *sin superponer ninguna estructura adicional con el propósito de su representación en la máquina*" [de Miguel Castaño, 1993]. El trabajo publicado por Codd en 1970, presentaba un nuevo modelo de datos que perseguía una serie de objetivos, muchos de ellos comunes a otros modelos, que se pueden resumir en la figura 3.11.

Es importante mencionar, la importancia que Codd concede al tema de la independencia de la representación lógica de los datos, respecto a su almacenamiento interno (independencia de *ordenación*, independencia de *indexación* e independencia de los *caminos de acceso*). Tal como expresa Codd desde su primer artículo dedicado al modelo relacional, en cuyo resumen se puede leer: "...se propone un modelo relacional de datos como una base para proteger a los usuarios de sistemas de datos formateados de los cambios que potencialmente pueden alterar la representación de los datos, causados por el crecimiento del banco de datos y por los cambios en los caminos de acceso"[De Miguel, 1993].

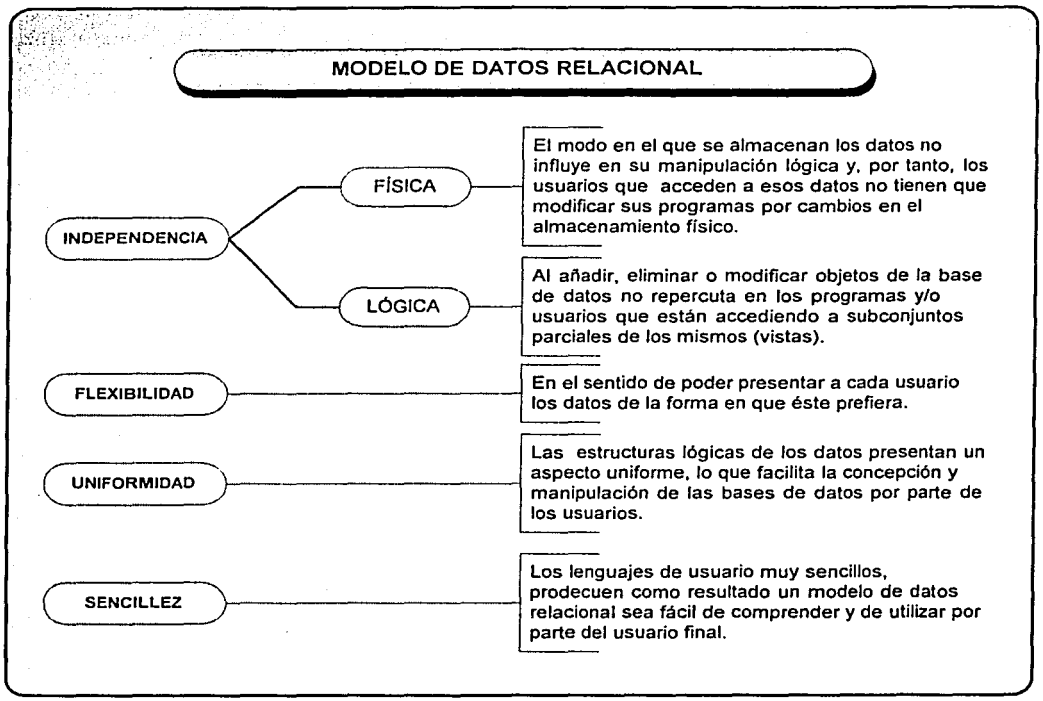


Figura 3.11 Objetivos del Modelo de Datos Relacional

La información de una base de datos, se representa en forma de relaciones cuyo contenido varía en el tiempo. Con respecto a la parte dinámica del modelo, se propone un conjunto de operadores que se aplican a las relaciones. Algunos de estos operadores son clásicos de la teoría de conjuntos, mientras que otros fueron introducidos específicamente para el modelo relacional. Todos ellos conforman el álgebra relacional definida formalmente por Codd (1972) [De Miguel, 1993].

La teoría de normalización, cuyas tres primeras formas normales fueron introducidas por Codd desde sus primeros trabajos, elimina dependencias entre atributos que originan anomalías en la actualización de la base de datos, y proporciona una estructura más regular en la representación de relaciones, constituyendo el soporte para el diseño de base de datos relacionales.

Si se analiza la evolución del modelo relacional que se muestra en la figura 3.12, se observa que después de los primeros estudios teóricos que se extienden a partir de 1970, comienza el desarrollo de diversos prototipos, entre los que destacan el Sistema R, que fue el origen de los productos relacionales de IBM (DB",SQL/DS, etc.), e INGRES, que fue creado en la Universidad de Berkeley, Stonebraker (1986), y convertido más tarde en un sistema comercial [De Miguel, 1993].

A pesar de que desde su introducción, en 1970, el modelo relacional se convirtió en uno de los principales temas de investigación en bases de datos, los primeros sistemas relacionares tardaron unos diez años en aparecer en el mercado. Llegándose a calificar de *juguetes,* más aptos para la investigación o para el desarrollo de base de datos experimentales o de pequeño tamaño, que para el soporte de verdaderos sistemas de información.

Probablemente, la teoría relacional nació cuando la tecnología existente en aquellos momentos, no podía ofrecer una adecuada base para instrumentaciones, que respondiesen eficientemente a las necesidades de los usuarios. Es decir, se podía considerar la teoría relacional, como un niño *prematuro* cuya *cuna* no estaba preparada en el momento de su nacimiento [De Miguel, 1993].

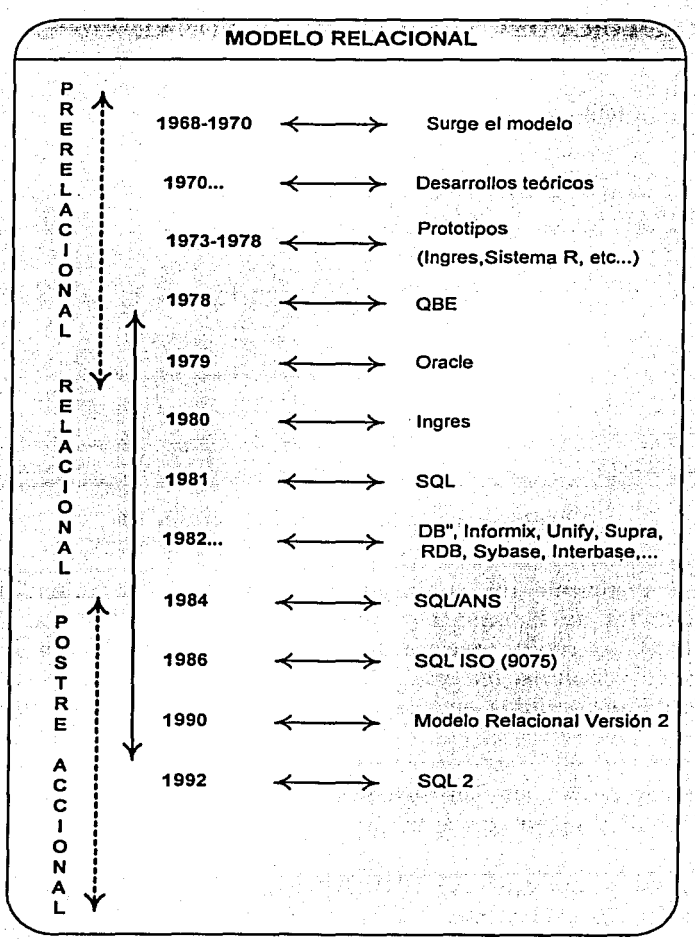


Figura 3.12 Evolución del Modelo Relacional

A pesar de ello, el modelo de datos relacional ha tenido un auge espectacular desde finales de los años setenta y, sobre todo, en los años ochenta. Una vez que empezaron a venderse las dificultades que presentaba su instrumentación y gracias la desarrollo tecnológico, que ha permitido una mayor eficiencia de los productos relacionales. A lo largo de estas dos décadas, se han publicado miles de artículos y libros que han ido aclarando y ampliando el modelo originalmente propuesto por Codd, y también han ido apareciendo productos comerciales que corren en las más diversas plataformas con rendimientos muy aceptables. Incluso, en muchos casos, comparables a los de los sistemas soportados en modelos convencionales, aunque en este terreno sigue habiendo autores, que ponen en duda la posibilidad de que los productos relacionales puedan llegar, en cuanto a eficiencia, a la altura de los basados en otros modelos.

3.3.2.1.1 ESTRUCTURA DEL MODELO RELACIONAL

La relación es el elemento básico del modelo relacional, y se puede representar como se muestra en la tabla 3.1.

Nombre

atributo 1	atributo 2	.....	atributo n	
xxxxxxx	xxxxxxx	.....	xxxxxxx	tupla 1
xxxxxxx	xxxxxxx	.....	xxxxxxx	tupla 2
xxxxxxx	xxxxxxx	.....	xxxxxxx	
xxxxxxx	xxxxxxx	.....	xxxxxxx	
xxxxxxx	xxxxxxx	.....	xxxxxxx	
xxxxxxx	xxxxxxx	.....	xxxxxxx	
xxxxxxx	xxxxxxx	.....	xxxxxxx	tupla n

Tabla 3.1 Representación de una relación en forma de tabla

En ella se puede distinguir un conjunto de columnas, denominadas *atributos*, que representan propiedades de la misma y que están caracterizadas por un nombre, y un conjunto de filas llamadas *tuplas*, que son las ocurrencias de la relación. El número de filas de una relación se le denomina *cardinalidad*, mientras que el número de columnas es el *grado*. Existen también *dominios* en donde los atributos toman sus valores [De Miguel, 1995].

Una relación se puede representar en forma de tabla, aunque tiene una serie de elementos característicos que la distinguen de una tabla:

- No puede haber filas duplicadas, es decir, todas las tuplas tienen que ser distintas.
- El orden de las filas es irrelevante.
- La tabla es plana, es decir, en el cruce de una fila y una columna sólo puede haber un valor (no se admiten atributos multivaluados).

Una relación siempre tiene un nombre, y en ella es posible distinguir una cabecera (esquema de relación o intensión) que define la estructura de la tabla; es decir, sus atributos con los dominios subyacentes, y un cuerpo, extensión, que está formado por un conjunto de tuplas que varían en el tiempo.

3.3.2.1.2 DOMINIO Y ATRIBUTO

Un dominio D es un conjunto finito de valores homogéneos y atómicos, V_1, V_2, ... V_n caracterizados por un nombre; se dice que los valores *homogéneos* porque son todos del mismo tipo, y atómicos porque son indivisibles en lo que al modelo se refiere, es decir, si se descompusieran perderían la semántica a ellos asociada.

Todo dominio tiene que tener un nombre por el cual se pueda hacer referencia hacia él y un tipo de datos. También se le puede asociar una unidad de medida, como metros, kilos, etc., y ciertas restricciones. Los dominios pueden definirse por intensión o por extensión.

Es muy usual dar el mismo nombre al atributo y al dominio subyacente. En el caso, de que sean varios los atributos de una misma tabla definidos sobre el mismo dominio, habrá que darles nombres distintos, ya que una tabla no puede tener dos atributos con el mismo nombre. Además de los dominios y atributos simples que se acaban de definir, en los últimos trabajos de algunos autores [Codd (1990), Date (1990)] se introduce el concepto de dominio compuesto que, es muy importante desde un punto de vista, para el diseño de bases de datos, y cuya ausencia en el modelo relacional básico no se encontraban justificadas [De Miguel, 1993].

Un *Dominio Compuesto*. Se puede definir como una combinación de dominios simples que tiene un nombre y a la que se pueden aplicar ciertas restricciones de integridad.

Tanto los atributos compuestos como los dominios compuestos pueden ser tratados, si así lo precisa el usuario, como *piezas únicas* de información, es decir, como valores atómicos. Prácticamente, ningún producto relacional reconoce ni trata adecuadamente el concepto de dominio, ni tampoco, en general, los productos son capaces de trabajar con atributos compuestos.

3.3.2.2 RELACIÓN

Matemáticamente, una relación definida sobre los *n* dominios $D_1,D_2,...,D_n$, no necesariamente distintos, es un subconjunto del producto cartesiano de estos dominios, donde cada elemento de la relación, tupla, es una serie de *n* valores ordenados.

Intensión o Esquema de Relación. Denotado R (A_1: D_1, A_2:D_2,..., A_n:D_n) es un conjunto de n pares atributo-dominio subyacente $\{(A_i:D_j)\}$ donde n es el *grado* del esquema de relación. La intensión es la parte definitoria y estática de la relación, que le corresponde con la cabecera cuando la relación se percibe como una tabla (el conjunto A de atributos sobre lo que define la relación se suele denominar contexto de la misma) [De Miguel, 1993].

Extensión u Ocurrencia (instancia) de la Relación. Llamada a veces simplemente relación, denotada por r(R) es un conjunto de m tuplas $\{t_1, t_2,..., t_m\}$ donde cada tupla es un conjunto de n pares atributo-valor $\{(A_i:V_{ij})\}$, donde V_{ij} es el valor j del dominio D_i asociado al atributo A_i; el número de tuplas m es la *cardinalidad.*

La relación r(R) es, por tanto, el conjunto de tuplas que, en un instante determinado, satisfacen el correspondiente esquema de relación R; así como el esquema de relación es –relativamente- invariante, su extensión varía en el transcurso del tiempo [De Miguel, 1993].

Cuando la relación se percibe como una tabla, el cuerpo de la tabla sería la extensión. En la figura 3.13 se presenta la intensión y la extensión de la relación de la tabla A; observándose que en la extensión se ha incluido también, tal como se hace habitualmente la cabecera con los nombres de los atributos, en lugar de la representación que correspondería estrictamente a la definición dada anteriormente.

INTENCIÓN DE UNA RELACIÓN

TABLA A (

ATRIB1:ATRIBUTOS1,
ATRIB2:ATRIBUTOS2,
ATRIB3:ATRIBUTOS3
)

EXTENSIÓN DE UNA RELACIÓN

ATRIB1	ATRIB2	ATRIB3
dato 1	dato 1'	dato 1''
dato 2	dato 2'	dato 2''
dato 3	dato 3'	dato 3''
dato 4	dato 4'	dato 4''

Figura 3.13 Intención y Extensión de una Relación

3.3.2.1.4 CLAVES

Una *clave candidata* de una relación es un conjunto no vacío de atributos, que identifican unívoca y mínimamente a cada tupla. Por la propia definición de relación, siempre hay, al menos, una clave candidata, ya que al ser una relación, en un conjunto no existen dos tuplas repetidas. Por tanto, el conjunto de todos los atributos identificará unívocamente a las tuplas; si no se cumpliera la condición de mínima, se eliminarían aquellos atributos que lo impidiesen. Una relación puede tener más de una clave candidata, entre las cuales se debe distinguir [De Miguel, 1993]:

- Clave Primaria. Es aquella clave candidata que el usuario escogerá, por consideraciones ajenas al modelo relacional, para identificar las tuplas de la relación.

- Claves Alternativas. Son aquellas claves candidatas que no han sido escogidas como clave primaria.

Se denomina *clave ajena* de una relación R_2 a un conjunto no vacío de atributos, cuyos valores han de coincidir con los valores de la clave primaria de una relación R_1 (R_1 y R_2 no son necesariamente distintas). Cabe destacar que la clave ajena y la correspondiente clave primaria han de estar definidas sobre los mismos dominios.

3.3.2.1.5 RESTRICCIONES

En el modelo relacional, al igual que en otros modelos, existen restricciones, es decir, estructura u ocurrencias no permitidas, siendo preciso distinguir entre restricciones inherentes y restricciones de usuario. Los datos almacenados en la base tienen que adaptarse a las estructuras impuestas por el modelo (por ejemplo, no tener tuplas duplicadas), y han de cumplir las restricciones de usuario para constituir una ocurrencia válida del esquema [De Miguel, 1993].

3.3.2.1.6 RESTRICCIONES INHERENTES

En el modelo relacional cabe mencionar como restricciones inherentes, además de las derivadas de la definición matemática de relación, la *integridad de entidad.*

De la definición matemática de relación, se deduce inmediatamente una serie de características propias de una relación, que se han de cumplir obligatoriamente. Por lo que se trata de restricciones inherentes y que diferencian una relación de una tabla [De Miguel, 1993].

- No hay dos tuplas iguales.
- El orden de las tuplas no es significativo.
- El orden de los atributos (columnas) no es significativo.
- Cada atributo sólo puede tomar un valor único del dominio, no admitiéndose por tanto los grupos repetitivos.

La *regla de integridad de entidad* establece que "Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo"; esto es, un valor desconocido o inexistente. Esta restricción debería aplicarse también a las claves alternativas, pero el modelo no lo exige [De Miguel, 1993].

3.3.2.1.7 RESTRICCIONES DE USUARIO

La restricción de usuario se considera dentro del contexto relacional, como un predicado definido sobre un conjunto de atributos, de tuplas o de dominios, que debe ser verificado por los correspondientes objetos, para que éstos constituyan una ocurrencia válida del esquema.

Dentro de las restricciones de usuario se destaca la restricción de *integridad referencial,* que se expresa de la siguiente manera "Si una relación R_2 (relación de referencia) tiene un distintivo que es la clave primaria de la relación R_1 (relación referenciada), todo valor de dicho distintivo debe, concordar con un valor de la clave primaria de R_1, o ser nulo". El distintivo es, por tanto, una clave ajena de la relación R_2 [De Miguel, 1993].

R_1 y R_2 son relaciones no necesariamente distintas. Además, cabe destacar que la clave ajena puede ser también parte de la clave primaria de R_2.

La integridad referencial es una restricción de comportamiento, ya que viene impuesta por el mundo real y es el usuario quien la define al describir el esquema relacional; es también de tipo implícito, ya que se define en el esquema y el modelo la reconoce sin necesidad de que se programe, ni de que se tenga que escribir ningún procedimiento para obligar a que se cumpla. Hay que observar que todo atributo de una clave primaria compuesta de una relación R_2, si no esta definido sobre un dominio compuesto, debe ser clave ajena de R_2, referenciando a una relación R_1, cuya clave primaria sea simple. Además de definir las claves ajenas, hay que determinar las consecuencias que pueden tener ciertas operaciones (BORRADO y MODIFICACIÓN) realizadas sobre tuplas de la relación referenciada, distinguiéndose las siguientes opciones (figura 3.14) [De Miguel, 1993].

OPERACIÓN RESTRINGIDA (RESTRICT)
Es el borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada, sólo se permite si no existen tuplas con dicha clave en la relación que contiene la clave ajena.

OPERACIÓN CON TRANSMISIÓN EN CASCADA (CASCADE)
Es el borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada, lleva consigo el borrado o modificación en cascada de las tuplas de la relación que contiene la clave ajena.

OPERACIÓN CON PUESTA A NULOS (SET NULL)
Es el borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada, lleva consigo poner a nulos los valores de las claves ajenas de la relación que referencia.

OPERACIÓN CON PUESTA POR DEFECTO (SET DEFAULT)
Es el borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada lleva consigo poner el valor por defecto a la clave ajena de la relación que referencia; valor por defecto que habría sido definido al crear la tabla correspondiente.

OPERACIÓN QUE DESENCADENA UN PROCEDIMIENTO DE USUARIO
En este caso, el borrado o la modificación de tuplas de la tabla referenciada pone en marcha un procedimiento definido por el usuario.

Figura 3.14 Operaciones comunes sobre tuplas

La opción seleccionada en caso de borrado es independiente de la de modificación. Cuándo las restricciones de integridad referencial afectan a varias tablas hay que tener en cuenta, que la combinación de opciones que se definan pueden provocar graves problemas de integridad. Existen otras restricciones de usuario como son:

- Las dependencias
- Restricciones entre elementos

También es importante en una restricción el momento en el que ésta se verifica dentro de una transacción. Así, si el modo de verificación es inmediato, la restricción se verificará al finalizar cada sentencia, mientras que si es diferido se verificará al finalizar la transacción.

3.3.2.2 MODELO EN RED

Euler (1707-1782) es considerado el precursor de la teoría de grafos, cuando en 1736 resolvió un problema famoso en su tiempo, llamado el problema de los puentes de Königsberg (figura 3.15). El río Pegel formaba islas en esa ciudad, y habían 7 puentes:

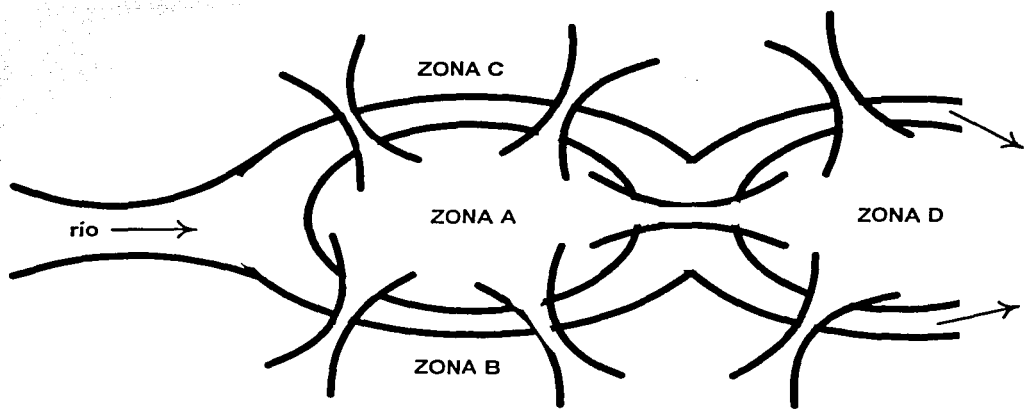


Figura 3.15 Mapa de los Puentes de Königsberg

El problema era comenzar en cualquiera de las cuatro zonas de tierra (A, B, C o D), atravesar sólo una vez por cada puente y volver al punto de partida. Para resolver este problema, Euler reemplazó cada zona de tierra por un punto, y cada puente por una línea uniendo los puntos correspondientes y produciendo un *multigrafo* (ver figura 3.16) [Elmasri, 2000].

Euler, así demostró que el problema no tenía solución. Para ello, generalizó el problema y desarrolló un criterio para que un multigrafo dado, fuera recorrible de la manera requerida. Este criterio pide que el grafo sea conexo y que cada punto tenga un número par de líneas incidentes. El multigrafo de los puentes de Königsberg es conexo, pero no todo punto tiene un número par de líneas incidentes [Elmasri, 2000].

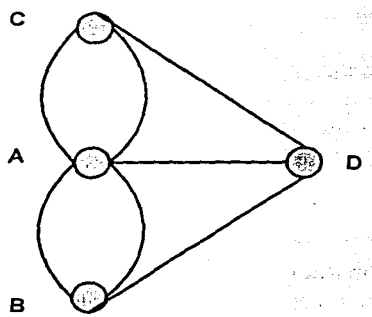


Figura 3.16 Multigrafo

Las redes constituyen una manera natural de representar las relaciones entre los objetos. Se utilizan ampliamente en las matemáticas, la investigación operativa, la química, la física, la sociología y otros campos. Como los objetos y sus relaciones constituyen maneras útiles de modelar muchos de los fenómenos en los negocios, no es sorprendente que en la arquitectura de retículos se aplique también a la organización de las bases de datos [Hansen,1997].

Generalmente, las redes se pueden representar mediante una estructura matemática llamada *grafo orientado.* Los grafos orientados tienen una estructura simple. Dentro del marco de los modelos de datos, los nodos pueden considerarse como tipos de registros de los datos, y las aristas pueden considerarse como la representación de las relaciones uno-uno o uno-muchos. Así, el modelo de datos en red representa los datos en estructuras de retículos de los tipos de registros, conectados por relaciones uno-uno o uno-muchos.

La estructura del grafo facilita la representación simple de las relaciones jerárquicas (como los datos genealógicos), las relaciones de pertenencia (como el departamento que se asigna a un empleado) y muchas otras. Además, una vez que se ha establecido una interrelación entre dos objetos, la recuperación y manipulación de los datos asociados puede realizarse eficientemente.

La organización Conference on Data System Languages (CODASYL), formada por representantes de los vendedores de hardware, de los vendedores de software y de los usuarios más importantes, inicialmente desarrolló y normalizó el lenguaje COBOL al comienzo de la década de los años sesenta. A finales de ésta década, esta organización nombró un subgrupo denominado Database TasK Group (DBTG) para desarrollar las normas para los sistemas de gestión de base de datos. Este subgrupo DBTG estaba fuertemente influido por la arquitectura utilizada para el primero de los SGBDs, el Integrated Data Store (IDS), desarrollado por General Electric.

Las recomendaciones para un modelo en red que aparecen publicadas en un informe preliminar en 1969, estuvieron bajo esta influencia. Este primer informe dio lugar a numerosas sugerencias para su perfeccionamiento, y se publicó un informe oficial revisado en 1971 que se sometió a la consideración del American National Standards Institute (ANSI). Esta organización no realizó acción alguna al respecto y los informes modificados de 1978, 1981 y 1984 sucedieron al informe de 1971 [Hansen, 1997].

Sin embargo, el documento de 1971 permanece como la proposición fundamental del modelo de red, que se convirtió en el modelo DBTG de CODASYL. Este ha servido de base para el desarrollo de los sistemas de gestión de bases de datos en redes. Aunque, el modelo de red cada vez más parece ceder el paso en el futuro al modelo de datos relacional, como el SGBD a elegir. Actualmente se utiliza con efectividad en numerosos sistemas de base de datos [Hansen, 1997].

3.3.2.2.1 PRESENTACIÓN DE UN MODELO DE RED GENERAL

En el modelo de datos en red general, se representan las entidades en forma de nodos de un grafo, y las asociaciones o relaciones entre éstas, mediante los arcos que unen dichos nodos. Esta representación, no impone en principio ninguna restricción ni al tipo ni al número de arcos, permite el modelado de estructuras de datos tan complejas como se desee.

En la figura 3.17 se presenta una estructura de datos (esquema) en el modelo de red, en la cual existen cuatro tipos de entidades (A, B, C, y D) y siete tipos de relaciones (de I1 a I7). La figura muestra la existencia de varias relaciones N:M (I1,I4); de una interrelación reflexiva de tipo 1:N (I2); de dos relaciones distintas entre las mismas entidades (I3 y I4 entre B y C); de una interrelación ternaria (I7), en la que intervienen tres entidades (B, C y D); y una interrelación reflexiva de tipo N:M sobre la entidad D (I5). Como puede observarse, este modelo en red permitirá la representación de cualquier tipo de interrelación sin ninguna restricción inherente [De Miguel, 1993].

Se puede definir el modelo en red general con una mayor formalización, como un conjunto finito de tipos de entidades:

$$\{ E_1, E_2, \ldots, E_n \},$$

con sus respectivas propiedades (atributos):

$$\{A_{I1}, A_{I2}, .. A_{Ik}, \ldots, A_{n1}, A_{n2}, .., A_{nm} \},$$

y un conjunto finito de relaciones (al igual que las entidades y que los atributos, tiene un nombre):

$$\{ I^h_{j,k,\ldots,n} \},$$

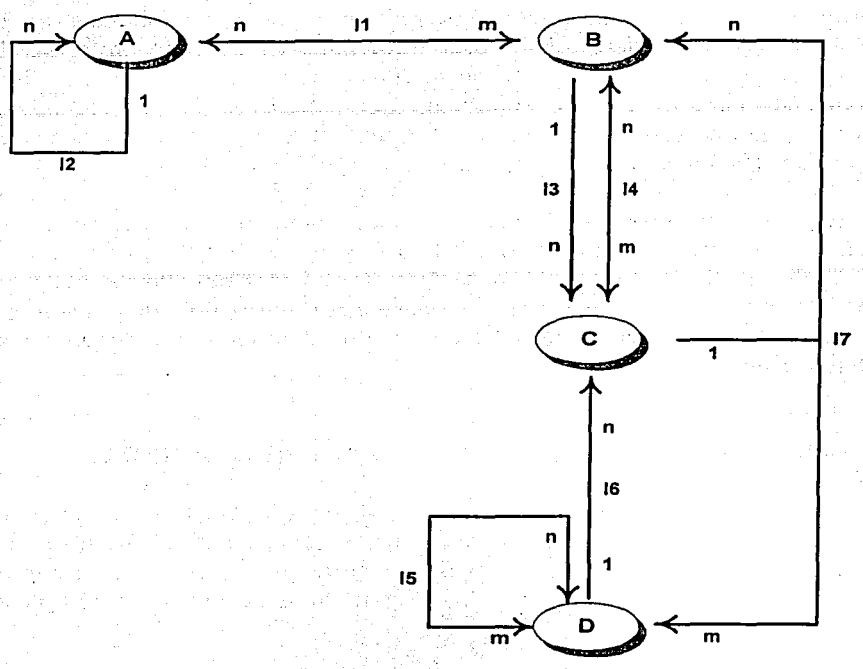


Figura 3.17 Estructura General de Datos en Red

Con la notación anterior representamos la interrelación entre los elementos j, k,..., n (bien sean entidades y/u otras relaciones), donde el superíndice h permite diferenciar dos relaciones distintas entre los mismos elementos, ya que se refiere al nombre de la relación. En la figura 3.18 se presenta un cuadro resumen del modelo, siguiendo una estructura formal. El esquema representa los aspectos estáticos, es decir, la estructura de los datos (tipo de entidades, tipo de relaciones, etc.), mientras que una ocurrencia del esquema (base de datos) son los valores que toman los elementos del esquema en un determinado momento, los cuales irán variando a lo largo del tiempo, por el efecto de aplicar los operadores de manipulación de datos a una ocurrencia del esquema.

Este modelo en red general es muy flexible, debido a la inexistencia de restricciones inherentes, pero también por esta misma razón su instrumentación física resulta difícil y poco eficiente. Esta es la causa de que el modelo, tal como lo hemos presentado, sea teórico, y que al llevarlo a la práctica se introduzcan restricciones [De Miguel, 1993].

El concepto básico en el enfoque de red es el conjunto ("set"), definido por el comité CODASYL. Un conjunto está constituido por dos tipos de registros que mantienen una relación de muchos a muchos.

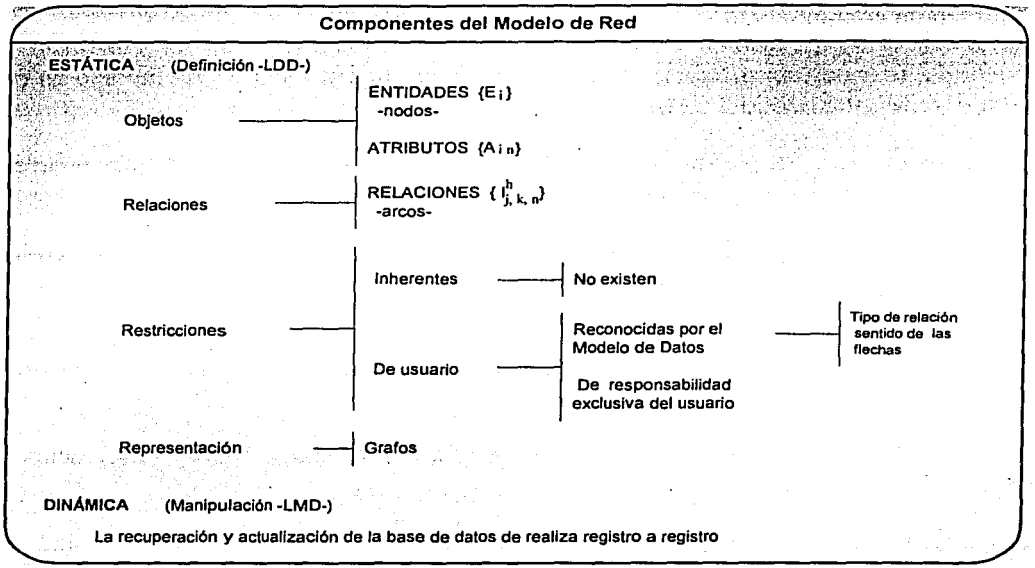


Figura 3.18 Estructura del Modelo en Red

Para conseguir representar este tipo de relación es necesario que los tipos de registros estén interconectados, por medio de un registro conector llamado conjunto conector. Los conjuntos poseen las siguientes características [De Miguel, 1993]:

- El registro padre se denomina propiedad del conjunto, mientras que el registro hijo se denomina miembro.
- Un conjunto está formado en un solo registro padre y uno o más registros hijos.
- Una ocurrencia de conjuntos es una colección de registros, uno de ellos es el registro padre y los otros los registros hijos.
- Todos los registros padre de ocurrencias del mismo tipo de conjunto, deben ser del mismo tipo de registro.
- El tipo de registro padre de un tipo de conjunto, debe ser distinto de los tipos de registros hijos.
- Sólo se permite que un registro hijo aparezca una vez en las ocurrencias de conjuntos del mismo tipo.
- Un registro hijo puede asociarse con más de un registro padre, es decir, puede pertenecer al mismo tiempo a dos o más tipos de conjuntos distintos.

- Se pueden definir niveles múltiples de jerarquías, donde un tipo de registro puede ser hijo en un conjunto, y al mismo tiempo padre en otro conjunto diferente.

En seguida, se muestran diferentes relaciones que se pueden tener en un modelo en red en general [De Miguel, 1993]:

Caso 1.- Este caso presenta un esquema de un tipo de relación N:M y algunas ocurrencias del mismo.

Caso 2.- Representa un esquema con un tipo de relación 1:N entre dos tipos de entidad y algunas ocurrencias del mismo.

Caso 3.- Este caso representa un tipo de relación reflexiva 1:N y una ocurrencia de la misma. Un tipo de relación reflexiva tiene lugar entre ocurrencias de un mismo tipo de entidad.

Caso 4.- En este caso se representa un tipo de relación reflexiva N:M y ocurrencias de la misma.

Caso 5.- Este caso presenta un tipo de relación entre más de dos tipos de entidad. En el ejemplo, se han presentado tres tipos de entidad, pero el modelo en red en general permite las relaciones entre cualquier numero de entidades.

La figura 3.19 muestra los diferentes casos descritos anteriormente, presentado el esquema general y ejemplos de ocurrencias de las propias relaciones.

Se observa por tanto, que el esquema de una base de datos en red general se puede representar mediante diagramas, y la base de datos correspondiente sería el conjunto de todas las ocurrencias de los distintos tipos de entidad, representados en el esquema con las vinculaciones existentes entre ellas.

Como ejemplos de DBMS comerciales basados en el modelo de red, se citan, el DMS 1100 de UNIVAC, el IDMS de Cullinane, el TOTAL, de Cincom, el EDMS, de Xerox, el PHOLAS, de Phillips, el DBOMP, de IBM y el IDS, de Honeywell

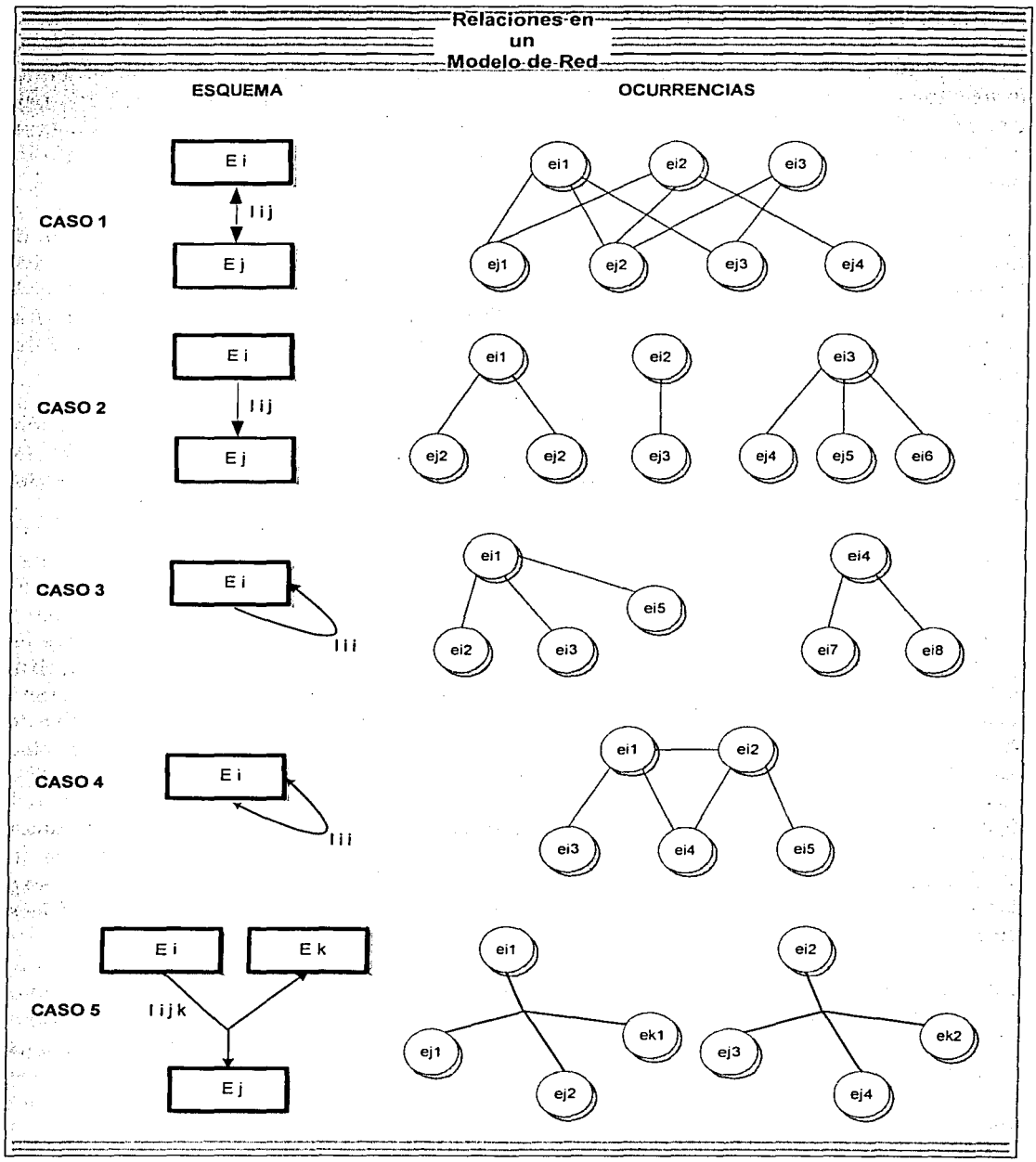


Figura 3.19 Relaciones Comunes en un Modelo de Red General

3.3.2.3 MODELO JERÁRQUICO

A diferencia del modelo de datos relacional, que se fundamenta firmemente en las matemáticas, y del modelo de datos en red, que se desarrolló a partir de un esfuerzo para establecer estándares detallados, el modelo de datos jerárquico se ha desarrollado a partir de la práctica. No existen documentos originales que definan el modelo jerárquico, como los hay para los otros modelos [Hansen, 1997].

Debido a que el modelo jerárquico no tiene un estándar, su estudio requiere el examen de los DBMS usados en la práctica. Afortunadamente, las implementaciones de BD jerárquicas están dominadas por un sistema, IMS (Sistema de Gestión de información de IBM). De hecho, IMS es actualmente el DBMS que generalmente más se usa. Las exposiciones en el modelo jerárquico invariablemente incorporan el vocabulario y las convenciones de IMS.

Sin embargo, se usan otros sistemas jerárquicos, incluyendo el TDMS (System Development Corporation's Time-Shared Data Management System), el MARK IV (Control Data Corporation`s Multi-Access Retrieval System) y el System-2000 (SAS Institute).

Ambos SMBD, jerárquico y red, fueron desarrollados a principios de 1960. El IMS se desarrolló con esfuerzo conjunto entre IBM y North American Aviation (mas tarde convertida en Rockwell), para desarrollar un SMBD como soporte al proyecto lunar Apolo –uno de los mayores proyectos de ingeniería cometidos en ese tiempo-. Un factor clave en el desarrollo de IMS fue la necesidad de manipular millones de piezas que se relacionaban unas con otras de manera jerárquica. Esto significa que piezas más pequeñas se usaban para construir montajes más grandes, que a su vez se convertirían en los componentes de módulos más grandes y así sucesivamente. Una razón complementaría para la durabilidad del IMS, es que muchas estructuras de datos son inherentemente jerárquica [Hansen, 1997].

Entre los primeros modelos de datos que surgieron en los SGBD comerciales, se encuentran las estructuras en árbol propias de los productos jerárquicos. Los árboles, tan extendidos en informática, como instrumentos para la representación de estructuras de datos. En este caso, por su poca flexibilidad, da origen a una falta de adaptación a muchas organizaciones reales.

Aunque no ha llegado a una formalización matemática del modelo y de sus lenguajes, como ha ocurrido en el caso del relacional; ni tampoco se ha intentado su estandarización, como en el CODASYL. Los productos jerárquicos (el IMS y el DL/I de IBM como máximos exponentes de estos sistemas) consiguieron altas cuotas en el mercado. Actualmente, la difusión de la tecnología los ha llevado a convertirse en sistemas superados, lo cual no quiere decir, que no persistan todavía importantes aplicaciones soportadas en estos productos. Esto debido a su eficiente respuesta, a satisfacción de sus usuarios, siempre que las aplicaciones desarrolladas sobre ellos se mantenga con algunos cambios.

3.3.2.3.1 CARACTERÍSTICAS DE LA ESTRUCTURA JERÁRQUICA

En el modelo de datos Jerárquico, el esquema es una estructura arborescente compuesta de nodos, que representan las entidades, enlazados por arcos, que representan las asociaciones o relaciones entre dichas entidades (en terminología IMS, los nodos se llaman *segmentos*, y en System 2000, *grupos repetitivos*) [De Miguel, 1993].

La estructura del modelo de datos jerárquico es un caso particular del modelo de red, con fuertes restricciones adicionales derivadas de las asociaciones del modelo jerárquico, que deben formar un árbol ordenado. Es decir, un árbol en el que el orden de los nodos es importante. Una estructura jerárquica , cuya representación se muestra en la figura 3.20, tiene las siguientes características [De Miguel, 1993]:

- El árbol se organiza en un conjunto de niveles.
- El nodo raíz, el más alto de la jerarquía (nodo A en la figura 3.20), corresponde con el nivel 0 (algunos autores consideran que se trata del nivel 1).
- Los arcos representan las asociaciones jerárquicas entre dos entidades y no tienen nombre, porque entre dos conjuntos de datos sólo puede haber una relación.
- Mientras que un nodo de nivel superior (padre) puede tener un número ilimitado de nodos de nivel inferior (hijos), al nodo del nivel inferior solo le puede corresponder un único nodo de nivel superior. En otras palabras, un progenitor (o padre) puede tener varios descendientes (o hijos), pero un hijo sólo tiene un padre (en la figura 3.20, A es padre de B, E y F; B de C y D; etc.; pero B sólo tiene a A como padre).
- Todo nodo, a excepción del nodo raíz, ha de tener obligatoriamente un padre.
- Se llaman *hojas* a los nodos que no tienen descendientes (en la figura 3.20 se han marcado las hojas con sus asteriscos).
- Se llama *altura* al número de niveles de la estructura jerárquica.
- Se denomina *momento* al número de nodos.
- Sólo están permitidas las relaciones 1:1 ó 1:M.
- Cada nodo no terminal y sus descendientes forman un subárbol, de forma que un árbol es una estructura recursiva.

El árbol, en este modelo, se recorre en preorden; es decir, raíz, subárbol izquierdo y subárbol derecho (en la figura 3.20 se indica el recorrido del árbol).

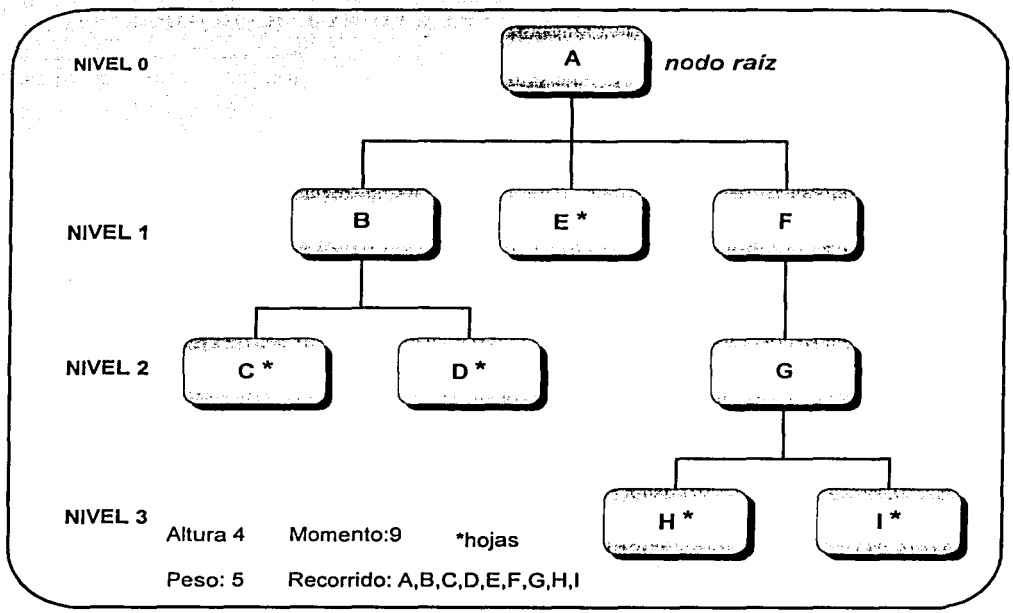


Figura 3.20 Estructura Arborescente a varios niveles

3.3.2.3.2 CLASIFICACIÓN DE LOS ÁRBOLES

Los árboles se pueden clasificar atendiendo a varios criterios. Así, podemos distinguir entre:

- *Árbol Balanceado*. Es aquel que en todos los nodos tienen el mismo número de hijos, excepto el último en el preorden (ver figura 3.21).

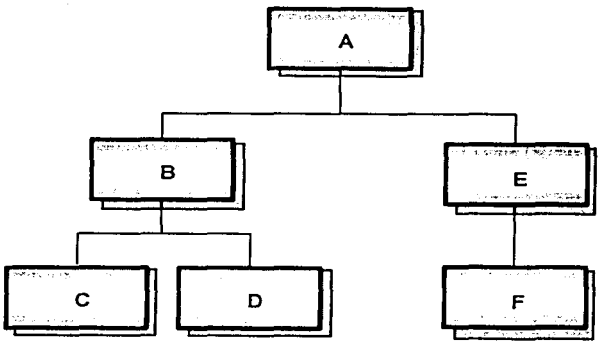


Figura 3.21 Estructura de un Árbol Balanceado

- *Árbol no Balanceado*. Es aquel en el que los nodos pueden tener diferente número de hijos (ver figura 3.22)

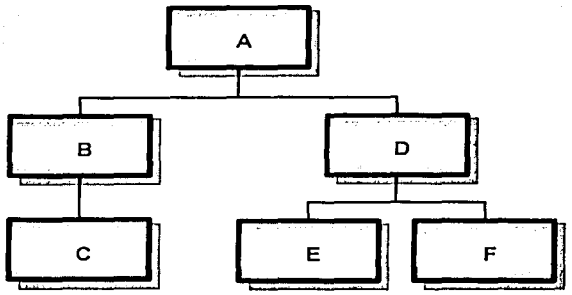


Figura 3.22 Estructura de Árbol no Balanceado

- *Árbol Binario*. Es aquel en el que cada nodo tiene cero, uno o dos hijos (ver figura 3.23).

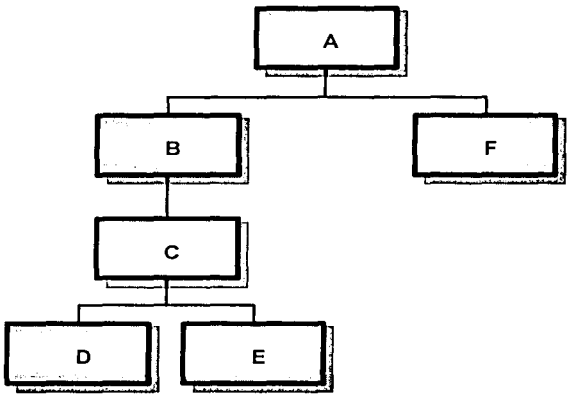


Figura 3.23 Estructura de Árbol Binario

- *Árbol Estrictamente Binario*. Es aquella en el que cada nodo tiene cero o dos hijos (ver figura 3.24.

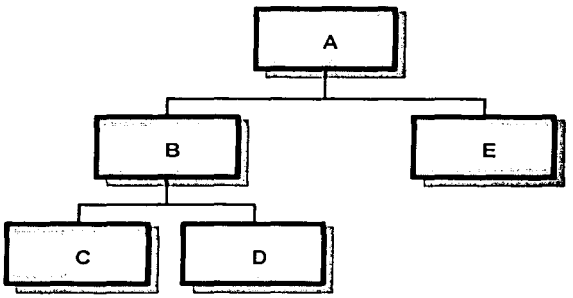


Figura 3.24 Estructura de un Árbol Estrictamente Binario

Las estructuras jerárquicas se clasifican también como:

- *Lineales*. Es un caso particular y simple en el que cada tipo de registro padre sólo puede tener un tipo de registro hijo (ver figura 3.25), donde se muestra la relación entre A - B que es a padre-hijo.

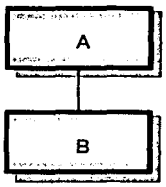


Figura 3.25 Estructura Jerárquica Lineal

- *Arborescente*. Propiamente dicho, es un tipo de registro donde el padre puede tener varios tipos de registros descendientes. En la figura 3.26 se muestra un ejemplo de este tipo de estructura donde, ahora existe otro hijo llamado proyecto por lo cual esto queda A – B – C.

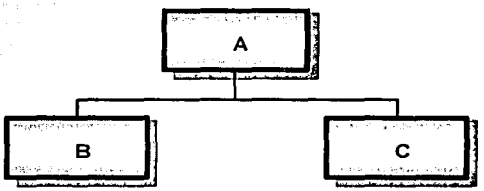


Figura 3.26 Estructura Jerárquica Arborescente

Es importante mencionar el hecho, de que las organizaciones jerárquicas pueden servir para describir tanto estructuras lógicas como físicas. La clasificación anterior se aplica principalmente a las estructuras arborescentes de tipo físico [De Miguel, 1993].

3.3.2.3.3 ESQUEMA Y OCURRECIA DE ÁRBOL

Un esquema jerárquico consiste en una descripción de un determinado universo del discurso, mediante un árbol en el que los nodos representan los tipos de registro (entidades), y los arcos los tipos de relaciones jerárquicas existentes entre los mismos. Una ocurrencia o instancia de dicho esquema será también un árbol, pero en él los nodos representan las ocurrencias de los registros, y los arcos las relaciones jerárquicas entre dichas ocurrencias. En la figura 3.27 se representan un esquema y una ocurrencia del mismo [De Miguel, 1993].

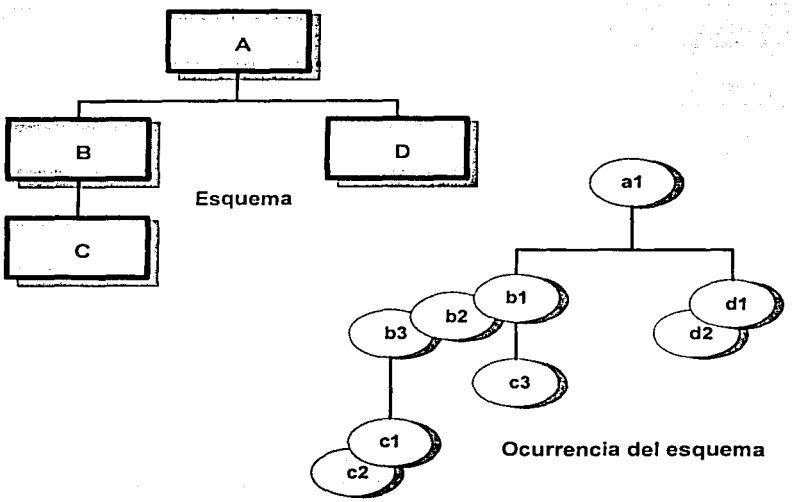


Figura 3.27 Esquema en Árbol y una Ocurrencia del mismo

Una base de datos jerárquica está formada por una colección o bosque de árboles disjuntos.

3.3.2.3.4 DEFINICIÓN GENERAL DEL MODELO JERÁRQUICO

El modelo jerárquico se define como [De Miguel, 1993]:

- Un conjunto de tipos de entidad (segmentos, grupos repetitivos, registros, etc.) E_1, E_2, ... , E_n (nodos de un grafo).
- Un conjunto de relaciones o asociaciones no nominadas R_{ij} que conectan los tipos de entidad E_i y E_j (arcos del grafo).
- Un conjunto de restricciones inherentes que provienen de la estructura jerárquica.

3.3.2.3.5 PROBLEMAS DEL MODELO JERÁRQUICO

El modelo de datos jerárquico presenta inconvenientes, que provienen principalmente de su rigidez. La cual deriva de la falta de capacidad de las organizaciones jerárquicas para representar, sin redundancias, ciertas estructuras muy difundidas en la realidad, como lo son las relaciones reflexivas y N:M, las redes, etc. La poca flexibilidad de este modelo puede obligar a la introducción de redundancias. Cuando es preciso instrumentar, mediante el modelo jerárquico, situaciones del mundo real que no responden a una jerarquía; así ocurre con el esquema no jerárquico de la figura 3.28 [De Miguel, 1993].

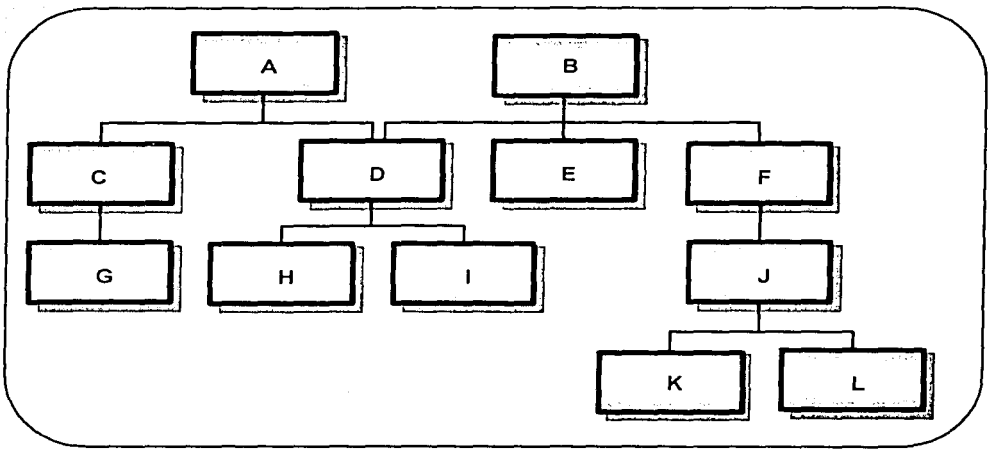


Figura 3.28 Estructura de Red

Cuando se quiere adaptar la estructura en red de la figura anterior al modelo jerárquico, es preciso crear dos árboles e introducir redundancias, ya que los registros D, H e I aparecen en ambas jerarquías (ver figura 3.29). Se puede comprender que se trata de una pobre solución de diseño.

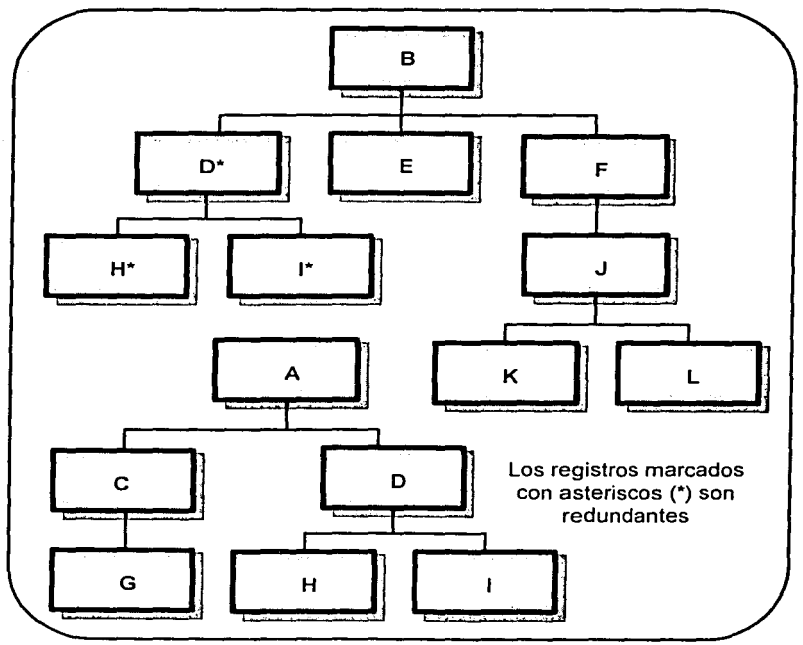


Figura 3.29 Estructura jerárquica a partir de una estructura de red

Se puede calcular un índice de redundancia (IR) mediante la siguiente fórmula [De Miguel, 1993]:

$$IR = \frac{\text{Nº. nodos extra}}{\text{Nº. total de nodos}} \times 100$$

En el caso de la figura 3.28, el índice de redundancia sería:

$$IR = \frac{3}{15} \times 100$$

Este índice de redundancia permite determinar hasta qué punto, una estructura del mundo real se adapta más o menos a un árbol (cuanto mayor es el índice de redundancia, más lejos se está de una estructura de árbol).

La redundancia que se calculó es una redundancia lógica en general, no coincide con la redundancia física. Esto es debido a que, al almacenar los datos en el dispositivo físico, no se suelen repetir los registros completos, sino sólo los identificadores, o bien se introducen punteros. Se trata ya de soluciones de instrumentación propias de cada suministrador. En general, los productos ofrecen algún tipo de solución a estos problemas. Una limitación importante del modelo jerárquico es el no estar preparado para representar relaciones N:M.

Además del grave problema que provocan estas redundancias no controladas por el sistema (con el aumento de almacenamiento y sobre todo las posibles incoherencias), existe otro inconveniente importante en este tipo de solución, como es, la no conservación de las simetrías naturales existentes en el mundo real. Las actualizaciones en la base de datos jerárquicas pueden también originar problemas, debido a las restricciones inherentes al modelo, tales como [De Miguel, 1993]:

- Toda alta, a no ser que corresponda a un nodo raíz, debe tener un padre.

- La baja de un registro implica que desaparezca todo el subárbol que tiene dicho registro como nodo raíz, con lo que pueden desaparecer datos importantes que tal vez se quisieran conservar en la BD.

Los DBMS basados en el modelo jerárquico, suelen facilitar instrumentos que los denotan de una mayor flexibilidad para representar estructuras que no son estrictamente jerárquicas. Sin embargo, puede ocurrir que dicha implantación no proporcione la debida independencia física/lógica.

Los sistemas jerárquicos de IBM (IMS y DL/I) permiten definir, lo que se llaman *relaciones lógicas de bases de datos lógicas,* para salvar este tipo de situaciones. Por lo que respecta a las restricciones de usuario, el modelo jerárquico no ofrece ninguna posibilidad de definirlas [De Miguel, 1993].

En cuanto a las ventajas proporcionadas por los SMBD de tipo jerárquico, cabe citar su sencillez de comprensión y la mayor facilidad de instrumentación en los soportes físicos. Aunque esto depende en gran medida de los productos. Además, si se trata de estructuras del mundo real que sean de tipo jerárquico, este modelo puede ser muy idóneo. Por otra parte, los productos jerárquicos suelen ser muy eficientes (en especial el IMS).

3.3.2.3.6 LA FUNCIÓN DE MANIPULACIÓN DE DATOS

La no existencia de estándares para el modelo jerárquico, así como la difusión de los sistemas jerárquicos de IBM, ha llevado a que el DL/I, lenguaje de datos de estos sistemas, se impusiera, en su momento, como un estándar de facto. La manipulación de datos jerárquicos, al igual que ocurre en todo modelo, necesita, al menos en un plano de abstracción, *localizar* (seleccionar) primero los datos sobre los que se va a trabajar para realizar la acción de recuperación o actualización sobre dichos datos [De Miguel, 1993].

A) Localización o Selección

La función de selección jerárquica es de tipo navegacional, es decir, trabaja registro a registro, en oposición a otros modelo como el relacional en los que se selecciona conjuntos de registros. Dada la sencillez del modelo, la función de selección es también muy sencilla, existiendo únicamente las siguientes formas básicas de búsqueda [De Miguel, 1993]:

- Seleccionar un determinado conjunto de datos (como un registro) que cumpla una cierta condición.

- Seleccionar el siguiente conjunto de datos, que se encuentra perfectamente definido al existir un único camino jerárquico (preorden). También en este caso se puede imponer una condición que ha de cumplir el registro para ser seleccionado.

- Seleccionar el registro padre de otro dado (que ha sido activado previamente), se conoce como normalización jerárquica ascendente, mientras que la selección de descendientes se llama normalización jerárquica descendente.

Puesto que una instrucción jerárquica selecciona un único registro, será preciso que el lenguaje de datos esté embebido de un lenguaje de programación mediante el cual se describa el procedimiento necesario, para ir recorriendo el árbol con el fin de buscar y manipular los datos.

B) La Función de Acción

Una vez seleccionado un registro, se tendrá que realizar sobre él una acción, sea de recuperación o de actualización. La recuperación consiste, como en cualquier otro lenguaje de datos, en llevar el registro marcado como activo, en la selección realizada previamente, al área de entrada/salida [De Miguel, 1993].

Debido a la naturaleza jerárquica de las conexiones entre registros, las inserciones y borrados de registros requieren consideraciones especiales:

- Cuando un nuevo registro se inserta en una base de datos jerárquica, excepto para la raíz, se conecta automáticamente a un nodo padre previamente localizado mediante alguna sentencia de selección. El nuevo registro se inserta como hijo del registro padre seleccionado.
- Cuando un registro se borra en una base de datos jerárquica, excepto si se trata de una hoja, se borran todos los registros descendientes de él.

3.3.3 MODELOS FÍSICOS DE DATOS

Se utilizan para describir datos en el nivel más bajo. A diferencia de los modelos lógicos de datos, hay muy pocos modelos físicos de datos en uso. El modelo semántico de datos forma parte de este tipo de modelos físicos. Además, dentro de estos modelos se encuentran los llamados modelos primitivos (orientados al archivo), que están formados por el modelo unificador y el modelo de elementos, los cuales están completamente en desuso.

3.3.3.1 MODELO SEMÁNTICO

El modelo semántico de datos fue originalmente desarrollado con el propósito de incrementar la efectividad y la precisión del diseño de bases de datos. Los métodos de modelado semántico fueron considerados apropiados para muchos problemas de usuario, y podrían ser convertidos con facilidad a modelos, con realizaciones basadas en registros, tales como los modelos jerárquico, de redes y funcional. Abrial introdujo el modelo binario semántico de datos en 1974, y éste fue continuado durante los años siguientes por el modelo entidad/relación de Chen, el modelo semántico de datos de Hammer y McLeod y el modelo de datos funcional. Este modelo semántico, recoge los datos según el punto de vista del usuario y la información provista por informes, pantallas existentes, entrevistas con el usuario, etc. [Hansen, 1997].

El modelo consiste en una estructura, hecha de abstracciones para capturar las características semánticas esenciales de la empresa. Estas abstracciones se combinan de modo consistente con las reglas de estructura del modelo semántico, para formar el modelo de la empresa [Hawryszkiewycz, 1994].

Las abstracciones de modelo semántico están en un nivel más alto que los elementos de datos; por lo general, permiten el modelado a nivel objeto y la formación de clases, y asociaciones de objetos de las mismas clases. Los elementos de datos se pueden agregar al modelo una vez que se ha acordado la estructura semántica esencial. Sin embargo, los modelos semánticos en general no poseen ningún criterio que se pueda usar para evitar redundancias.

La disponibilidad de los modelos semánticos abstractos sugiere tres métodos más. El primer paso en todos éstos es desarrollar un modelo del usuario, basado en las abstracciones del modelo semántico [Hawryszkiewycz, 1994]. En ese momento, se dan los tres métodos [Hawryszkiewycz, 1994]:

- Transferir el modelo de empresa directamente al modelo de implementación.
- Normalizar el modelo de empresa, usando el criterio relacional y luego convertirlo en una implementación.
- Transferir el modelo de empresa a relaciones y luego convertir las relaciones en una implementación.

Las dos últimas opciones ofrecen tres ventajas para los diseñadores:

- Al inicio del diseño, es posible usar las abstracciones semánticas en un nivel de diseño más alto, después, emplear un nivel de elementos de datos en las etapas posteriores del diseño.
- Un problema de uso se puede modelar con conceptos semánticos, especialmente seleccionados para capturar características esenciales.
- Se puede usar la teoría relacional para proporcionar el criterio de la estructura de datos.

El objetivo de los modelos semánticos, es proporcionar abstracciones que se adapten con naturalidad a la forma como los usuarios describen sus empresas. Para hacer esto, los modelos semánticos deben:

- Soportar los procesos usados en el análisis.
- Usar los términos naturales para el análisis.
- Proporcionar la estructura del modelo.

Los modelos semánticos recientes se orientan a lo práctico y no necesitan una notación matemática excesiva. La idea que los sustenta, es permitir a los diseñadores especificar sus propios requerimientos, por medio de una estructura orientada a usuarios, con la mínima notación matemática [Hawryszkiewycz, 1994].

Estos modelos proporcionan varias abstracciones para definir, nombrar y clasificar conjuntos de objetos y asociaciones. La mayoría de los modelos permite a los diseñadores definir diferentes tipos de conjuntos de objetos y diversos tipos de asociación. Estos tipos se pueden ver como representaciones completas de abstracciones de modelado, y son análogos a los tipos de variables en lenguajes de programación. Por tanto, igual que con los tipos reales o enteros, un modelo semántico proporciona, por ejemplo, un tipo de conjunto de entidad, un tipo de conjunto de relación, etc.

Para definir y hacer crecer el modelo de empresa, es necesario considerar el modelado semántico en tres niveles [Hawryszkiewycz, 1994]:

- El nivel de modelo semántico o metamodelo, el cual define las abstracciones y las reglas para clasificar conjuntos de objetos.
- El nivel de empresa, que define una empresa de manera consistente con las reglas del modelo semántico.
- El nivel de objeto, que crea objetos en el nivel de empresa de manera consistente con las reglas del modelo semántico.

El modelo semántico se puede ver como un lenguaje de programación, en donde las declaraciones corresponden al nivel del modelo de empresa. El modelo se puede hacer con base en algunas declaraciones de instancias (o variables) de los tipos permitidos.

Las notaciones de nombramiento y clasificación proporcionan bases aceptables para definir modelos semánticos. Un modelo semántico proporciona al usuario abstracciones a fin de clasificar objetos en la empresa. Muchas de estas abstracciones están disponibles y también incluyen habilidades para especificar distintas clases de objetos, interacciones entre clases de objetos y asociaciones de subclases de objetos. Así como dependencias y otras categorías de conjuntos de objetos. De igual manera, el modelo semántico contiene reglas que se deben satisfacer con objetos en conjuntos y asociaciones [Hawryszkiewycz, 1994].

Debido a la variedad de clases de objetos y asociaciones, y las formas diferentes de estructurarlas, hay muchos modelos semánticos. Entre los más conocidos están:

- El modelo Entidad-Relación que comprende la entidad y la relación de los conceptos semánticos.
- Los modelos semánticos basados en las abstracciones de asociación y generalización.
- Los modelos semánticos que tienen por objeto enfatizar los subtipos como extensiones de los modelos semánticos o como modelos especiales.

Estos modelos no se excluyen mutuamente; se pueden usar las mismas abstracciones pero diferentes términos para nombrarlos y distintas estructuras de modelo. Así, la misma empresa del usuario se puede describir con cada modelo, aunque en diferentes formas.

3.3.3.1.1 REPRESENTACIÓN DE LOS MODELOS SEMÁNTICOS

Las estructuras semánticas se muestran mediante redes semánticas. La red semántica (figura 3.30) es una gráfica formada por:

- Un conjunto de puntos o nodos para describir objetos semánticos.
- Un conjunto de enlaces para representar las asociaciones entre esos objetos.

Una red semántica puede describir una estructura semántica a cualquier nivel. Si se modela una estructura en el modelo semántico o a un nivel de metamodelo, los puntos representarán abstracciones semánticas y los enlaces serán las asociaciones permitidas, entre tales abstracciones. Si la red semántica genera un modelo de empresa, los puntos significarán conjuntos de objetos o clases y los enlaces serán las asociaciones entre esas clases [Hawryszkiewycz, 1994].

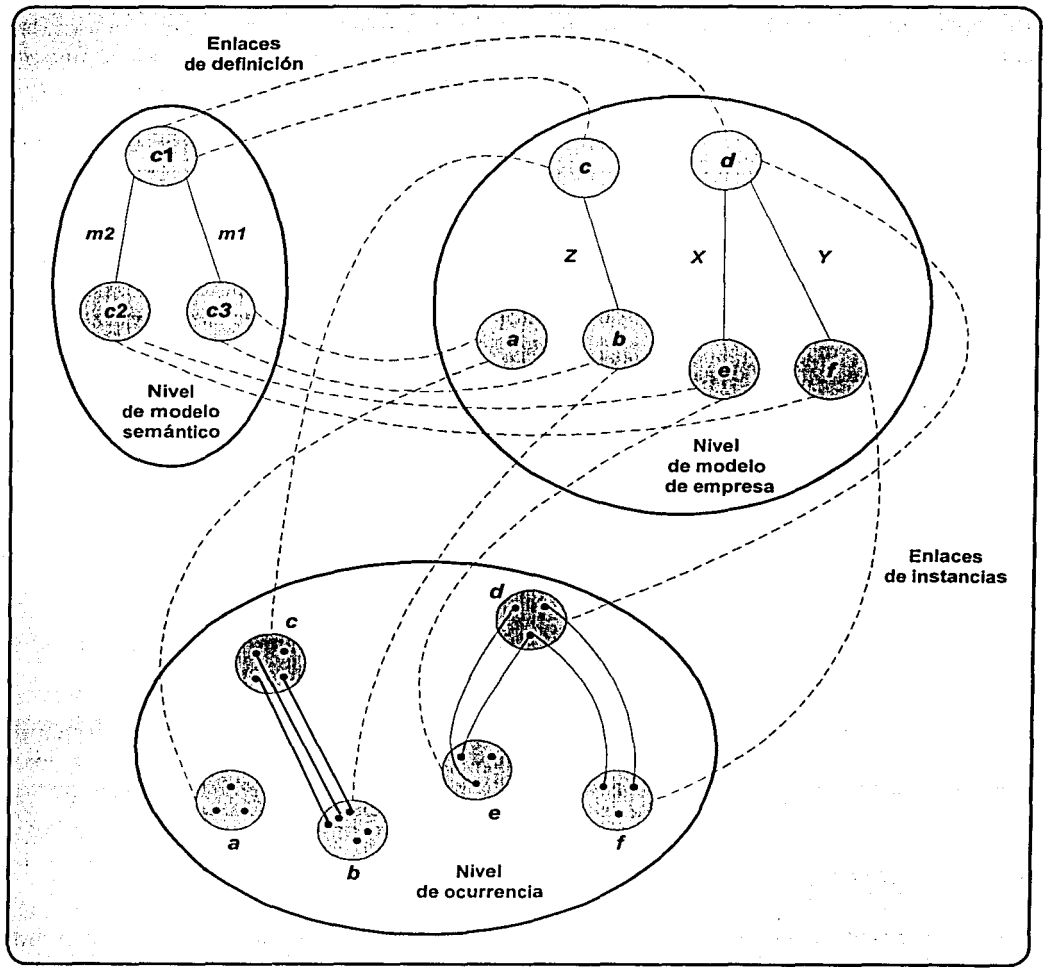


Figura 3.30 Redes Semánticas

Las asociaciones deben ser consistentes con las asociaciones en el nivel de modelo semántico. La consistencia se mantiene con los enlaces de la definición. Si la red semántica modela el nivel de ocurrencia, los puntos representarán los objetos de la empresa real. Los objetos se pueden ligar de modo consistente con las asociaciones en el modelo de empresa. Esta consistencia se mantiene con los enlaces en la instancia. Los niveles son genéricos y el modelo de empresa se genera a partir del modelo semántico; la clasificación de los objetos de empresa, a su vez, corresponde al modelo de empresa [Hawryszkiewycz, 1994].

CAPÍTULO 4

METODOLOGÍAS DE INGENIERÍA DE SOFTWARE

4.1 INTRODUCCIÓN

Dentro del contexto de la ingeniería de software en la década de los ochenta y, sobre todo, en sus últimos tres años, vuelven a estar de moda algunas metodologías para el desarrollo de proyectos informáticos, por lo cual nuevamente empiezan a utilizarse.

El empleo de técnicas y teorías que hasta ahora parecían difíciles de implantar en una empresa, por el costo añadido que suponían y por el retraso en la entrega de los proyectos, empieza a ser posible a medida que dichas técnicas son soportadas por estaciones de trabajo que han hecho más fácil su uso. Así, la Ingeniería del Software Asistida por Computadora, que podría traducirse como el acrónimo que proviene de la expresión en inglés "Computer Aided Software Engineering" mejor conocido por CASE, como el terreno con mayores innovaciones en los últimos dos años, y de las metodologías soportadas por herramientas CASE como el entorno de trabajo del ingeniero de sistemas en los próximos tiempos [López, 1991].

La eliminación de los inconvenientes que tenían las metodologías, la facilidad de uso dentro de un ambiente de producción integrado e informatizado, han hecho y van a hacer estos años que metodologías como MERISSE (1977) o SSADM (1981) vuelvan a estar de actualidad. Técnicas de representación como las de YOURDON, MÉTRICA y JAMES MARTIN, pasan a emplearse de modo sencillo en una computadora personal (PC). El modelado de datos y estructuras según diversas metodologías empieza a ser posible sin un gran esfuerzo, así como, el mantenimiento de un diccionario de datos (DD) y de la generación de la documentación de una aplicación [López, 1991].

Este capítulo pretende, aportar información sobre las metodologías y técnicas más empleadas en cada una de las fases del desarrollo de aplicaciones.

4.2 METODOLOGÍA DE EDWARD YOURDON.

Edward Yourdon es el representante de la corriente metodológica más importante de Estados Unidos, aunque hay numerosos autores que aportan variantes, matices y formalismos de representación al método de Yourdon.

A lo largo de sus obras Yourdon describe técnicas para la realización de análisis estructurado de sistemas basado principalmente en los siguientes conceptos:

b) Diagramas de flujo de datos para la representación de procesos.

c) Diagramas de transición de estados para la representación estructurada de las funciones a realizar en los procesos.

d) Modelo entidad/relación para la representación conceptual de datos.

e) Diccionario de datos como base o soporte de información del sistema.

f) Diagramas o mapas de estructura para la representación modular de los procesos y las variables intercambiadas entre ellos.

g) Especificaciones de programas basadas en lenguaje estructurado y tablas de decisión.

Yourdon define las siguientes etapas y niveles en el ciclo de vida de los sistemas informáticos (ver figura 4.1) [López, 1991].

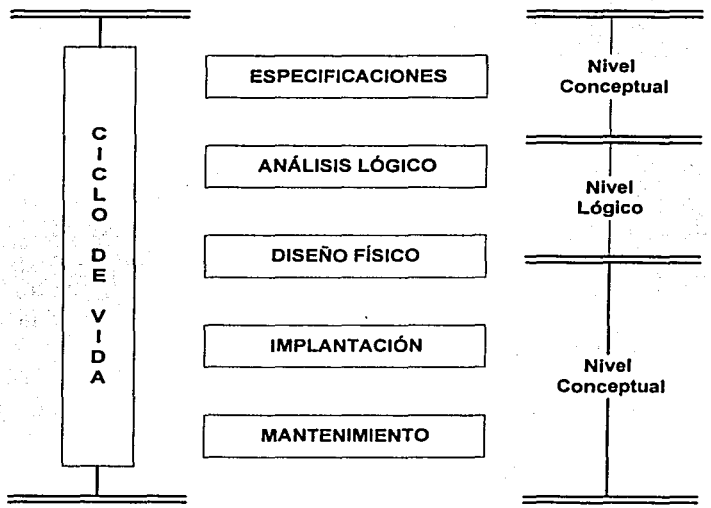


Figura 4.1 Etapas y niveles en el ciclo de vida de los sistemas.

4.2.1 ESTUDIO DE VIABILIDAD

En este punto se debe identificar el proyecto a realizar, los usuarios responsables y se debe hacer un estudio de la situación actual, representando la misma a través de un diagrama de flujo de datos (D.F.D.) de primer nivel o diagramas de contexto en los que, de forma simple, se indiquen los procesos simples más relevantes [López, 1991].

La viabilidad y el análisis de riesgo están relacionadas de muchas maneras. Si el riesgo del proyecto es alto, la viabilidad de producir software de calidad se reduce. Durante la ingeniería industrial, concentramos nuestra atención en cuatro áreas principales de interés [Pressman, 1998]:

Viabilidad económica. Una evaluación del costo de desarrollo comparado con los ingresos netos o beneficios obtenidos del sistema o producto desarrollado.

Viabilidad técnica. Un estudio de función, rendimiento y restricciones que puedan afectar a la consecución de un sistema aceptable.

Viabilidad legal. Determinar cualquier infracción, violación o responsabilidad legal en que se podría incurrir por el desarrollo del sistema.

Alternativas. Una evaluación de los enfoques alternativos al desarrollo del sistema o producto.

4.2.2 ANÁLISIS DEL SISTEMA

En esta fase se debe representar mediante las técnicas de diagramas de flujo, modelo entidad - relación, diagramas de transición de estado, etc., del sistema a desarrollar.

Se lleva a cabo con los siguientes objetivos en mente:

- Identificar la necesidad del cliente
- Evaluar el concepto del sistema para establecer la viabilidad
- Realizar un análisis técnico y económico
- Asignar funciones al hardware, software, personal, bases de datos y otros elementos del sistema.
- Establecer las restricciones de presupuesto y planificación temporal.
- Crear una definición de sistema que forme el fundamento de todo el trabajo de ingeniería subsiguiente.

Se requiere un gran dominio del hardware y del software (así como de la ingeniería humana y de base de datos) para conseguir con éxito los objetivos mencionados anteriormente [Pressman, 1998].

4.2.3 DISEÑO

En esta tercera fase, se pasa del nivel conceptual, a un nivel de representación lógica de los datos mediante la etapa de diseño dependiente del modelo de base de datos elegida y una estructuración de los procesos utilizando diagramas de estructura de los mismos y generando las especificaciones de programa correspondientes [López, 1991].

Es un proceso y un modelo a la vez. El proceso de diseño es un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos del software a construir. La capacidad creativa, la experiencia acumulada, el sentido del "buen" software y un empeño global en la calidad son factores críticos del éxito del diseño.
El modelo del diseño es el equivalente a los planos de una casa para un arquitecto. Empieza representando la totalidad de lo que se va a construir. Y lentamente se va refinando para proporcionar un directriz para construir cada detalle. Similarmente, el modelo de diseño creado para el software proporciona visiones diferentes del programa de computadora [Pressman, 1998].

4.2.4 IMPLEMENTACIÓN O PRODUCCIÓN

Comprende la generación de código y el ensamblaje, así cómo, la integración de todos los módulos [López, 1991].

4.2.5 PRUEBAS Y TEST DEL SISTEMA

Al llegar a la totalidad del sistema hasta llegar a la aceptación del mismo por parte del usuario. En esta fase se harán pruebas de integración y de funcionamiento al conjunto de programas y cadenas [López, 1991].

Durante las fases anteriores de definición y de desarrollo, el ingeniero intenta construir el software partiendo de un concepto abstracto y llegando a una implementación tangible. A continuación, llega la prueba. El ingeniero crea una serie de casos de prueba que intentan demoler el software construido. De hecho, la prueba es uno de los pasos de la ingeniería del software que se puede ver (por lo menos, psicológicamente) como destructivo en lugar de constructivo [Pressman, 1998]. Características de las pruebas:

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

4.2.6 CONTROL DE CALIDAD

El objeto de esta actividad es garantizar los controles de calidad del software que puedan estar definidos para la empresa. Esta fase complementa la anterior, de forma que el producto final sea de un buen nivel de calidad y cumpla los estándares fijados [López, 1991].

4.2.7 DOCUMENTACIÓN

En este apartado se generará toda la documentación necesaria para la instalación del sistema: manuales de usuario, de operación, etc. La documentación interna, o sea, las especificaciones de programas, habrán sido creadas ya anteriormente y utilizadas por los programadores. Con ello el nuevo sistema queda completamente documentado, interna y externamente [López, 1991].

4.2.8 CONVERSIÓN DE LOS DATOS DEL SISTEMA ANTERIOR

La ejecución de esta fase depende, evidentemente, del estado anterior a la mecanización del entorno afectado por el proyecto. Si existía ya un sistema informatizado, se deben realizar los programas de conversión de datos al nuevo sistema, y si anteriormente los archivos eran manuales puede requerirse una grabación y carga previa a la puesta en marcha del sistema [López, 1991].

4.2.9 INSTALACIÓN

Comprende la puesta en marcha del sistema y en esta fase son de aplicación las consideraciones hachas de aspectos tales como formación y entrenamiento del usuario, entrega de manuales, procesos paralelos, etc. [López, 1991].

4.3 METODOLOGÍA MERISE

A principios de los años setenta se comenzó a constatar la importancia que iba adquiriendo la información que manejaban las empresas a la hora de analizar sus resultados, para tomar decisiones con rapidez y orientar sus estrategias convenientemente. A continuación se presenta la época del dominio de los tratamientos, considerándose los datos exclusivamente como ficheros que tenían sentido sólo en función de los programas que los trataban.

Posteriormente, con el advenimiento de las bases de datos, se dio prioridad a los datos, estudiándolos de manera independiente de los tratamientos que los usaban y dando lugar al nacimiento del ciclo de abstracción, con sus tres conocidos niveles: conceptual, organizativo y físico [Piattini, 1995].

A mediados de los años setenta el Ministerio de Industria francés, dándose cuenta de las habilidades que presentaban estos métodos en relación con las nuevas necesidades de enfoque global, se propuso confrontar los diversos puntos de vista de una selección de profesionales de la ingeniaría informática. Para normalizar los elementos de un método informático nacional que fuera la síntesis enriquecida de los métodos existentes, y que abandonase la idea de creación de simples aplicaciones informáticas y asumiese la concepción de sistemas de información (SI) completos.

Esta aportación proporciona:

a) Un marco de análisis global que tiene en cuenta las perspectivas de evolución, la coherencia total del SI, el papel de la información en la empresa y el estado actual de las tecnologías.
b) Una herramienta de planificación del desarrollo.
c) Una progresión por etapas, implicando controles, validaciones, elaboración de documentos y obtención de resultados.
d) Una estructura de diálogo entre los participantes en el sistema de información (dirección, informáticos y usuarios).

En 1979, y a partir de una agrupación de expertos en ingeniería del software pertenecientes a diversas empresas de servicios coordinados por el Ministerio de Industria francés, nació MERISE como metodología de análisis y diseño de sistemas de información, aportando un plan de trabajo y técnicas de modelado para la concepción de aplicaciones coherentes para el área de la gestión de las empresas, pasando por lo que hasta entonces se había considerado como análisis, a lo que a partir de entonces se denomina concepción de sistemas de información, suponiendo dicha concepción una verdadera intersección entre informática y organización. Además, integrando los sistemas en el marco común diseñado por RACINES. En 1982 y también bajo el patrocinio del Ministerio de Industria francés, se procedió a actualizar RACINES y a definir sus interfaces con MERISE (ver figura 4.2). A partir de ese momento, RACINES y MERISE forman un cuerpo metodológico completo agrupado bajo la denominación de metodología MERISE, quedando como etapas del ciclo de vida [Piattini, 1995].

4.3.1 ETAPAS

I. *Estudio previo*. Tiene por misión definir de manera global las soluciones conceptuales, organizativas y técnicas del futuro SI en relación con el dominio objeto del estudio.

II. *Estudio detallado*. Permite definir explícitamente las especificaciones funcionales de la aplicación objeto del estudio.

III. *Estudio técnico*. Realiza la distribución de datos en ficheros físicos, y el tratamiento en módulos de programas en función del sistema informático a usar.

IV. *Realización y puesta en marcha.* Producción de los módulos de programación e implantación de los medios técnicos y organizativos necesarios, así como, formación del personal, lanzamiento de la aplicación, así como, la recepción definitiva por parte del usuario.

V. *Mantenimiento.* Etapa que abarca el resto de la vida del sistema, hasta que sea sustituido por otro nuevo.

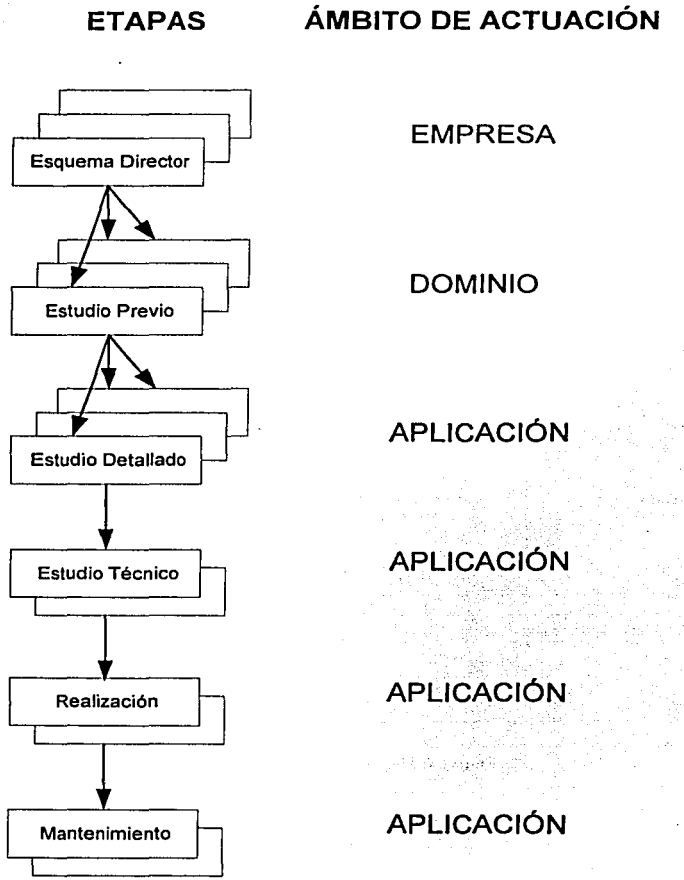


Figura 4.2 Etapas del ciclo de vida en la metodología MERISE.

4.3.2 CARACTERÍSTICAS IMPORTANTES DE MERISE

a) Aproximación racional para la resolución de problemas mediante la combinación de los diferentes niveles de abstracción considerados, las etapas de concepción y desarrollo definidas. Lo que permite jerarquizar los problemas y aportar soluciones construidas apoyándose en técnicas de modelado.

b) El modelo de tratamientos basado en las redes de Petri, que proporciona una visión cronológica y completa de las relaciones entre procesos [Piattini, 1995].

c) La coherencia de los sistemas resultantes, obtenida gracias a la construcción de modelos de acuerdo con la empresa y el dominio.

d) El amplio grado de difusión alcanzado por el método, lo que garantiza las inversiones realizadas y facilita el hallazgo de expertos en el mismo.

4.3.3 ETAPAS DE DESARROLLO

Por otra parte, MERISE al igual que otras metodologías, presenta esquemáticamente las fases antes mencionadas, en la siguiente figura [López, 1991].

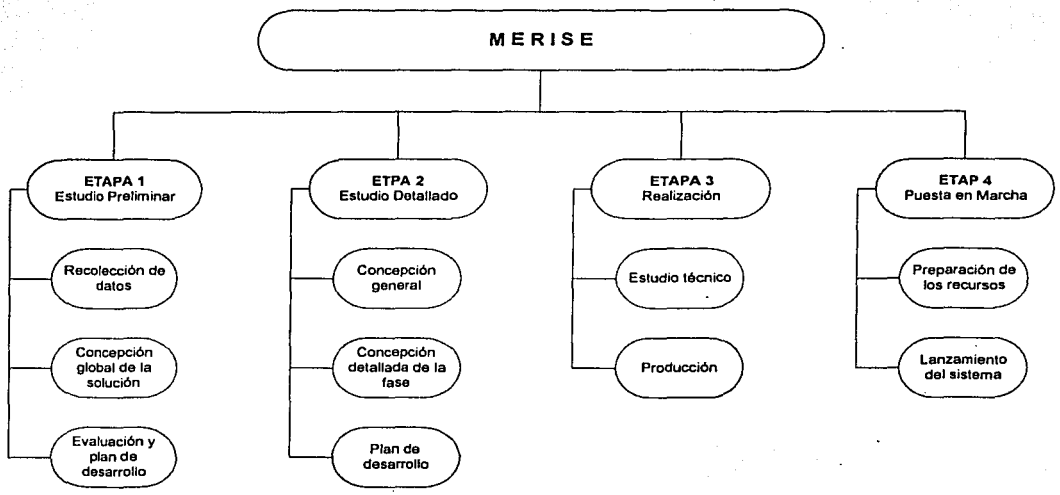


Figura 4.3 Fases de la metodología MERISE.

4.3.3.1 ETAPA 1. ESTUDIO PRELIMINAR

El objetivo de esta recolección de la información inicial es describir el objeto general del proyecto, es decir, qué se trata de conseguir, que áreas se verán afectadas por el mismo. Después de esta toma de datos inicial el objetivo siguiente es la descripción de la situación actual en las áreas afectadas, determinando de la forma más precisa las formas actuales de funcionamiento, flujos de información entre las diferentes unidades, así mismo, se debe realizar el modelo organizativo de los procesos realizados en el entorno estudiado (M.O.T). Ya que una vez descrita la situación actual, se hace un análisis de la misma, una crítica de puntos débiles y una propuesta de mejoras y soluciones a lo anterior.

Todo esto con el fin de encontrar una nueva solución a un nivel muy general, en el cual se plasmarán modelos de datos y de procesos con las soluciones propuestas en el modelo conceptual de datos (M.C.D), y el modelo conceptual de tratamientos (M.C.T) [López, 1991].

Mediante el M.C.D se identificarán las entidades del sistema y las relaciones entre ellas. Este modelo podrá ser modificado y afinado en etapas posteriores. En cuanto a los procesos o tratamientos, se hará el M.C.T para identificar las funciones a realizar por el sistema.

Aportando información sobre el coste de la nueva solución en cuanto a recursos materiales y humanos, haciendo al final un balance económico del proyecto, hasta llegar a una propuesta formal definitiva. En la cuál se abarcan puntos cómo la estructuración de sistemas así como subsistemas, plan de desarrollo, resumen del estudio preliminar, lo cual esquemáticamente se presenta en la figura 4.4 [López, 1991].

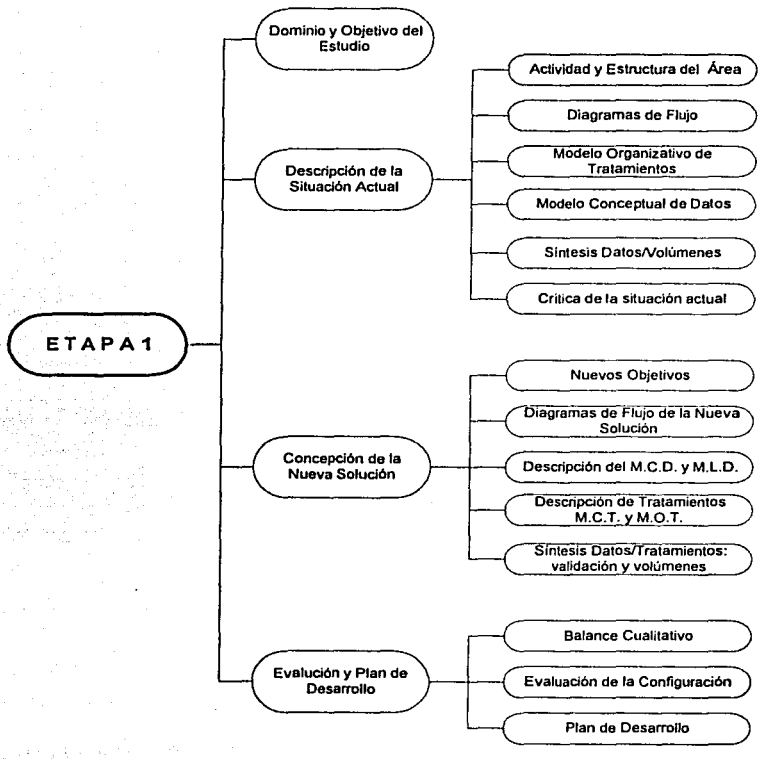


Figura 4.4 Fases y Subfaces en la ETAPA de Estudio Preliminar.

4.3.3.2 ETAPA 2. ESTUDIO DETALLADO

El objetivo de esta fase es la redacción de un informe general sobre la solución propuesta, indicando todos los aspectos generales que van a rodear al desarrollo en el cual se describen los diferentes procesos, con sus correspondientes entradas y salidas: pantallas, listados, etc. Así mismo con el plan de desarrollo, en donde se realiza una planificación del desarrollo y la puesta en marcha del nuevo sistema a implantar.
Para elaborar este plan se partirá de las especificaciones detalladas de las unidades del sistema que ya se han realizado. Para evaluar los tiempos de realización de los diferentes programas, habrá que partir de la complejidad y número de los mismos, de estándares de programación sobre los soportes utilizados (lenguajes, ayudas a la programación, herramientas CASE de generación de código, etc.) y, sobre todo, de la experiencia real adquirida sobre la capacidad de producción en el entorno elegido (ver figura 4.5) [López, 1991].

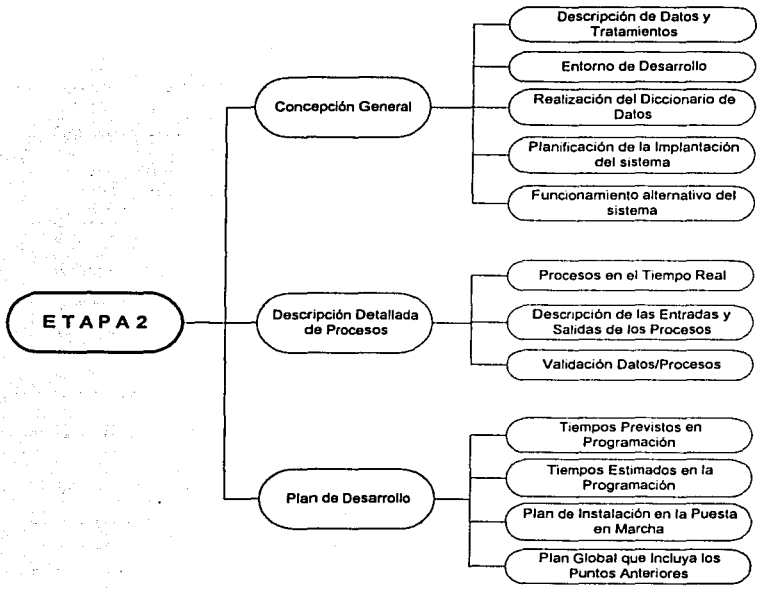


Figura 4.5 Fases y Subfaces en la ETAPA de Estudio Detallado.

4.3.3.3 ETAPA 3. REALIZACIÓN

El objeto de esta fase es, evidentemente, la producción y prueba de los programas que componen el sistema de acuerdo a las normas dadas y a las especificaciones detalladas (ver figura 4.6). Así como, fijar el entorno de desarrollo, formas de trabajo, especificar en forma general un informe técnico y las bases sobre las que van a funcionar las personas que van a intervenir en el desarrollo [López, 1991].

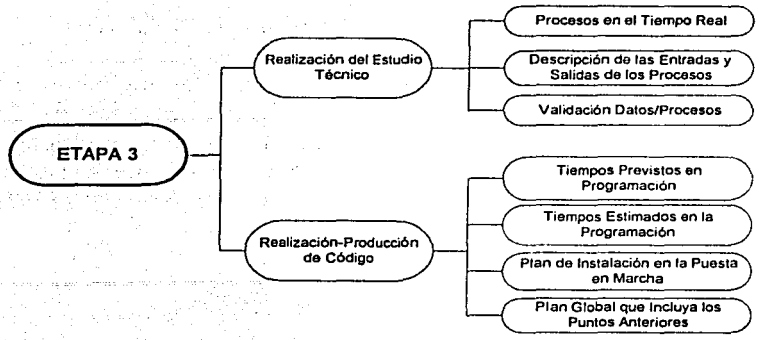


Figura 4.6 Fases y Subfaces en la ETAPA de Realización.

4.3.3.4 ETAPA 4. PUESTA EN MARCHA

El objetivo a cubrir por esta fase es la presentación, según la planificación realizada, de todos los recursos para la puesta en marcha del sistema. Estos recursos van, desde los elementos hardware o máquinas, hasta la revisión de la documentación necesaria para el usuario. Una vez preparados estos recursos, se producirá la recepción de la aplicación por parte del usuario y la puesta en marcha del mismo.

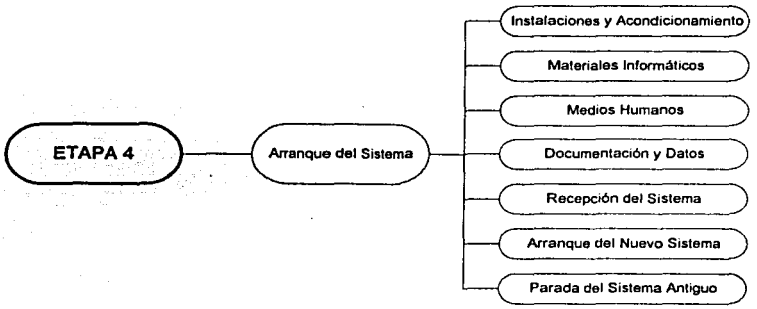


Figura 4.7 Fase y Subfaces de la ETAPA Puesta en Marcha.

Esta puesta en marcha será en la forma en que se haya previsto, teniendo en cuenta que podrá o no haber procesos paralelos del sistema antiguo y el nuevo. En estos periodos de trabajo en paralelo el nuevo sistema estará sujeto a una intensa observación con objeto de destacar cuanto antes todos los problemas y afinar el sistema. Figura 4.7 [López, 1991].

4.3.4 DOCUMENTACIÓN

Al realizar cada una de las etapas anteriores se obtienen diferentes documentos que se muestran en la figura 4.8 [López, 1991].

4.3.5 EVOLUCIÓN, TENDENCIAS Y FUTURO DEL MÉTODO

Desde su nacimiento hasta los nuevos días, la metodología MERISE ha experimentado una amplia difusión, que puede cifrarse en más de 35,000 usuarios solamente en Francia, en donde aproximadamente el 55% de las empresas que utilizan alguna metodología comercial han escogido MERISE. Igualmente, existe un gran número de usuarios en España, Bélgica, Suiza, Italia, América del Norte y en Magreb [Piattini, 1995].

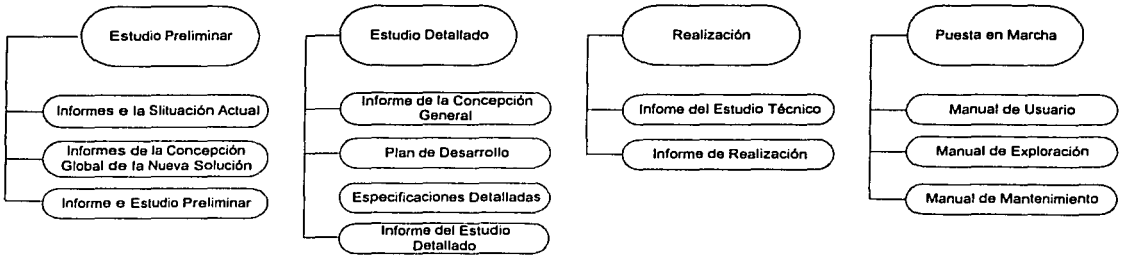


Figura 4.8 Resumen de la documentación generada.

De todas maneras, el nacimiento de una serie de nuevas tecnologías, como por ejemplo:

- Arquitecturas cliente-servidor.
- Bases de datos distribuidas.
- Interfaces gráficas para puestos de trabajo.
- Orientación al objeto.

Y la constancia de algunos puntos débiles surgidos de su utilización:

- Utilización algo pesada y monolítica.
- Mayor adaptación al desarrollo de medianos y grandes sistemas.

MERISE aconsejó la conclusión de nuevas técnicas en el seno de la metodología con objeto de, sin perder sus puntos fuertes, adaptarse mejor a las nuevas arquitecturas informáticas emergentes. Se han incorporado técnicas de composición/descomposición, como los diagramas de flujo de datos, los cuales se incorporan en el nivel conceptual del método [Piattini, 1995].

Desde el nivel conceptual, se tienen en cuenta las interacciones datos-tratamientos a fin de preparar la anterior distribución de unos y otros y de mejorar las posibilidades de definición de módulos reutilizables, incorporando los mecanismos de herencia y generalización propios de la orientación a objetos. Se extiende el formalismo de los datos realizando una mejor distinción entre los niveles organizativo y lógico para tener en cuenta las arquitecturas cliente-servidor. Asimismo, se diferencia entre el nivel lógico y el lógico repartido, separando la interfaz del corazón del aplicativo [Piattini, 1995].

4.4 METODOLOGÍA SSADM

En 1980, el Gobierno Británico se planteó la necesidad de adoptar una metodología de desarrollo que aminorara los típicos problemas que, de una u otra forma, afectaban a casi todos los departamentos informáticos en aquellos años: rotación de personal acelerada, falta de documentación de las aplicaciones, métodos triviales, sobrecarga de mantenimiento, etc., y decidió elaborar una metodología propia [Piattini, 1995].

El proyecto se realizó conjuntamente entre Central Computer and Telecommunications Agency (CCTA) y Learmonth and Burchett Management Systems (LBMS). El resultado fue SSADM (Structured Systems Analysis and Design Method). Al cabo de dos años, el uso de la metodología SSADM se hizo obligatorio en todos los proyectos informáticos del Gobierno Británico.

Cuando nació esta metodología, lo hizo sólo con las fases de análisis y diseño. La fase de estudio de viabilidad se incorporó en la versión 3.0, en julio de 1988. En el verano de ese mismo año, había ya 600 proyectos para SSADM en el Gobierno Británico, y poco después se informó el grupo de usuarios de SSADM en el sector privado. La estructura de la metodología SSADM versión 3, comprendía 8 etapas agrupadas en 3 fases: estudio de viabilidad, análisis y diseño. En 1990 apareció la versión 4, que supuso una remodelación importante en la estructura del método, haciéndola más modular, menos interactiva e incorporando nuevas técnicas fundamentalmente en el diseño de procesos (ver figura 4.9).

SSADM proporciona un conjunto de especificaciones y procedimientos para llevar a cabo las tareas de análisis y diseño de sistemas. No cubre la planificación estratégica ni el control de proyectos, y el diseño físico llega hasta el máximo grado de detalle en especificaciones, pero sin entrar en la construcción del código [Piattini, 1995].

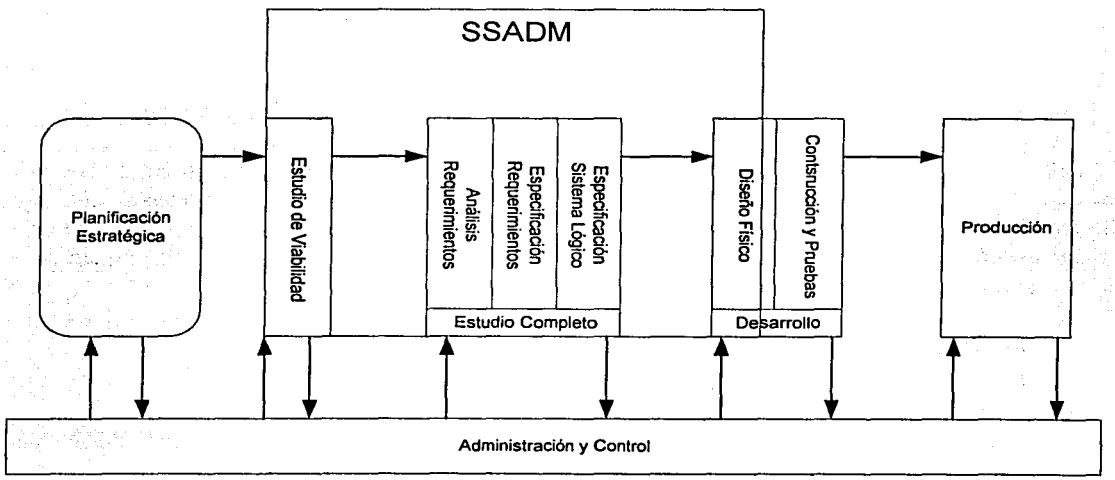


Figura 4.9 Posición de SSADM en el ciclo de vida.

El método contempla, en principio, las tres primeras fases del desarrollo (estudio de viabilidad, análisis y diseño), divididas a su vez en una serie de etapas cómo se muestra en la figura 4.10 [López, 1991].

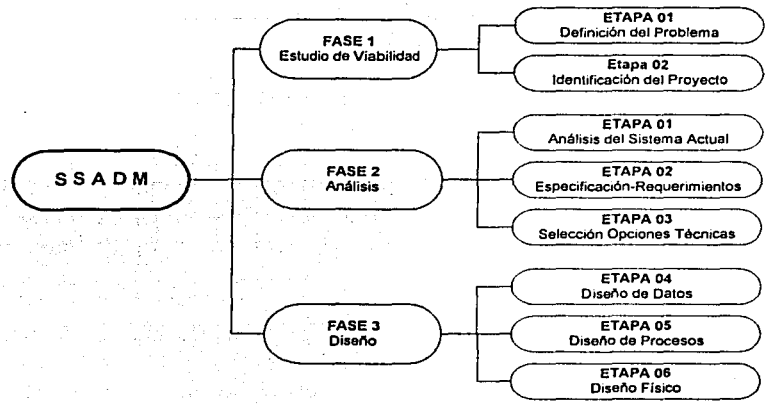


Figura 4.10 Metodología SSADM

En la fase de estudio de viabilidad se realizan actividades cómo las siguientes:

- Una evaluación del costo de desarrollo comparado con los ingresos netos o beneficios obtenidos del sistema o producto desarrollado.
- Un estudio de función, rendimiento y restricciones que puedan afectar a la consecución de un sistema aceptable.
- Determinar cualquier infracción, violación o responsabilidad legal en que se podría incurrir por el desarrollo del sistema.
- Una evaluación de los enfoques alternativos al desarrollo del sistema o producto.

No es necesario un estudio de viabilidad para sistemas en que la justificación económica es obvia, el riesgo técnico es bajo, se esperan pocos problemas legales y no existe ninguna alternativa razonable. Sin embargo, si falla alguna de las condiciones anteriores, se debería hacer un estudio del área en cuestión [Pressman, 1998]. La fase de análisis del sistema se llevan acabo los siguientes objetivos:

- Identificar la necesidad del cliente.
- Evaluar el concepto del sistema para establecer la viabilidad.
- Realizar un análisis técnico y económico.
- Asignar funciones al Hardware, software, personal, bases de datos y otros elementos del sistema.
- Establecer las restricciones de presupuesto y planificación temporal .

- Crear una definición de sistema que forme el fundamento de todo el trabajo siguiente.

En la fase de diseño se lleva a cabo el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositiva, un proceso o un sistema con suficiente detalle como para permitir su realización física. Se transforma el modelo de dominio de la información creado durante el análisis, en las estructuras de datos necesarias para implantar el software [Pressman, 1998]. El diseño es un proceso iterativo a través del cual se traducen los requisitos en una representación del software. Inicialmente, el anteproyecto muestra una visión holística del software. Es decir, el diseño se representa a un alto nivel e abstracción, un nivel que se puede seguir hasta requisitos específicos de datos, funcionales y de comportamiento. A medida que ocurren iteraciones del diseño, el refinamiento siguiente lleva a representaciones del diseño de mucho menor nivel de abstracción. Estos todavía pueden ser seguidos hasta los requisitos, pero la conexión es mucho más sutil [Piattini, 1995]. Cronológicamente, las seis etapas, una vez comprobada la etapa cero que es la de viabilidad del proyecto, se desarrollarán en el siguiente orden (ver figura 4.11) [López, 1991].

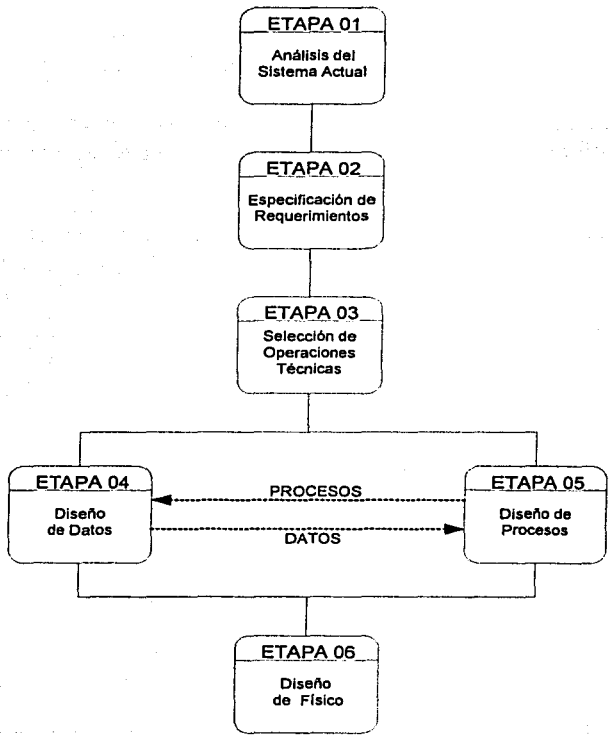


Figura 4.11 Etapas de la Metodología SSADM

Al utilizar abreviaturas de estas técnicas y de otros conceptos manejados por SSADM, para no prestarse a confusión se citará el concepto en castellano y a nivel de siglas, se usarán las mismas que las utilizadas para la metodología original [López, 1991].
Entre estas técnicas están:

- DIAGRAMAS DE FLUJO DE DATOS (DFD).
- ESTRUCTURA LÓGICA DE DATOS (LDS).
- HISTORIA DE LA VIDA DE LA ENTIDAD (ELH).
- TERCERA FORMA NORMAL (TNF).

DIAGRAMAS DE FLUJO DE DATOS (DFD), tienen por objeto la representación de los flujos de información en el sistema considerado, así como, los flujos de información con el exterior, es decir, con otros sistemas o subsistemas. Es una herramienta importante y útil ya que ayuda a describir posibles fallos de funcionamiento u organización. Los DFD en los diferentes niveles nos darán diferentes visiones y cada una de ellas nos servirá en una fase del desarrollo. Así, en la primera fase de análisis, buscaremos una representación global del sistema hasta elegir la opción más conveniente. Es posible que, después de la representación del sistema a través de los DFD se detecten errores o duplicidades en cuento al almacenamiento de datos por lo que habría que agruparlos de forma lógica para optimizar los procesos y la organización de los mismos [López, 1991].

ESTRUCTURA LÓGICA DE DATOS (LDS), el análisis de la estructura lógica de los datos en el modelo entidad/relación. Para realizar esta tarea se deben seguir los siguientes pasos:

1. Reconocer y seleccionar un conjunto inicial de entidades del sistema.
2. Analizar las relaciones entre ellas reflejándolo en una rejilla o tabla de doble entrada de entidades y relaciones.
3. Convertir la rejilla en la LDS.
4. Validar la LDS con la lista de requerimientos del sistema.
5. Racionalizar la estructura simplificando enlaces entre entidades.
6. Revisar que con el modelo simplificado se siguen satisfaciendo los requisitos.

En la siguiente tabla se muestra la correspondencia entre los términos utilizados por el modelo E/R y los usados en ficheros o bases de datos [López, 1991].

MODELO E/R	BASES DE DATOS
TIPO DE ENTIDAD	FICHERO
ENTIDAD	REGISTRO
ATRIBUTO	CAMPO DEL FICHERO

Tabla 4.1 Correspondencia entre los términos utilizados por el modelo E/R

HISTORIA DE LA VIDA DE LA ENTIDAD (ELH), representa la descripción de cómo las entidades descritas son afectadas por los procesos que ocurren en el sistema; Cómo son los de creación, actualización y descripción de una entidad a lo largo del tiempo [López, 1991]. La representación de la realidad dará lugar a diferentes estructuras de ELH de una entidad cómo son:

- Estructuras selectivas de sucesos.
- Estructuras de ocurrencia continua en un solo suceso.
- Estructuras de ocurrencia continua en varios sucesos.
- Estructura paralela de sucesos.

Si nos fijamos, estamos utilizando los mismos formalismos que al representar las acciones a realizar en un programa cómo son ITERACIÓN, SECUENCIA y SELECCIÓN [López, 1991].

TERCERA FORMA NORMAL (TNF), en la descripción de datos. Es un método matemático parta la definición de datos que ayuda a evitar inconsistencia y ambigüedades en la estructura de los mismos.

Especificaciones de Programa. Dentro de la metodología SSADM, de forma general, un programa es considerado como una función, que de forma estructurada da lugar a la figura 4.12 [López, 1991].

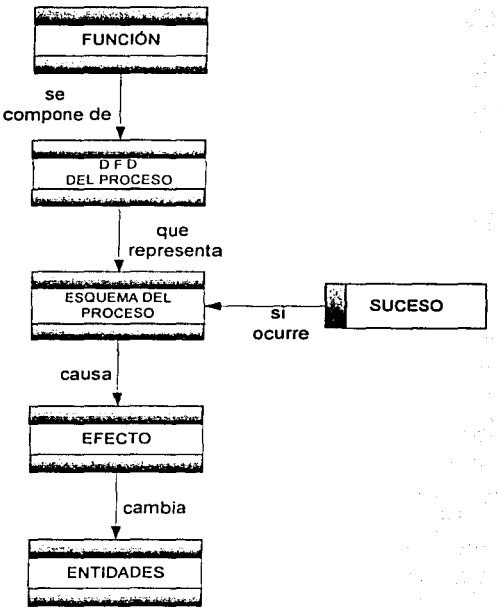


Figura 4.12 Estructura general de un programa

La descripción de cada programa, considerando como una función a aplicar sobre los datos, partiría de un DFD del proceso, dando lugar cada DFD a uno o varios esquemas de procesos que representará los efectos que producen cada uno de los recursos que pueden ocurrir. Estos efectos podrán dar lugar a determinados cálculos u operaciones con los datos o a la actualización de algunas entidades. Hay que tener en cuenta el tipo de relación entre los elementos que componen el programa o función. Así, observamos que cada función podrá dar lugar a varios DFD, cada DFD a varios esquemas de procesos, etc..

Ahora pasemos a clarificar los pasos a seguir con esquemas en los que se pueden ver la secuencia de los pasos y las técnicas utilizadas en cada uno de ellos. Hay que destacar la codificación de la numeración de las tareas a realizar de cada etapa y paso. De esta forma cualquier tarea de cualquier etapa queda completamente definida por su código [López, 1991]. Lo que viene a continuación debe considerarse como un recetario de puntos a realizar, algunos de los pasos pueden ser eliminados. Lo importante es que la eliminación sea con responsabilidad. Aparecerán abreviaturas de las diferentes técnicas cómo sigue [López, 1991].

DFD	Diagrama de flujo de datos
LDS	Estructura lógica de datos
ELH	Historia de vida de la entidad
LDD	Diseño de diálogos lógicos de pantallas
RDA	Análisis relacional de datos
CLDD	Diseño lógico completo de datos
PS	especificaciones de programa
PO	especificaciones de salidas del sistema

Pasemos a describir las diferentes fases y etapas que componen a dicha metodología, así cómo la esquematización de cada una de las etapas.

4.4.1 FASE 1. ESTUDIO DE VIABILIDAD
ETAPA 01. DEFINICIÓN DEL PROBLEMA

Su objetivo es analizar el sistema actual, establecer una lista de requerimientos del nuevo sistema y una vez que se tiene claro lo que se requiere conseguir, evaluar la complejidad del proyecto. A continuación se describen brevemente los pasos que conforman a la etapa 01 (ver figura 4.13) [López, 1991].

- INICIAR EL ESTUDIO DE VIABILIDAD [1], tiene cómo objetivo establecer la complejidad del proyecto, la duración aproximada de cada uno de los pasos y si existe, en algún plan estratégico.
- ANÁLISIS DEL SISTEMA [2], su objetivo es reflejar el funcionamiento del sistema actual a través de Diagramas de Flujo de Datos.
- CREACIÓN DE LA ESTRUCTURA LÓGICA DE DATOS [3], su objetivo es reflejar la estructura de datos actual en el sistema después de un estudio inicial.

- DESARROLLO DEL SISTEMA LÓGICO [4], convierte la visión física actual reflejada en los DFD de los pasos anteriores en una visión lógica del mismo.

- CONSIDERACIÓN DE LA LISTA DE PROBLEMAS Y REQUERIMIENTOS [5], se revisa la lista después de las actualizaciones hechas, eliminando duplicidades e impresiones.

- REVISIÓN DE LA DEFINICIÓN DEL PROBLEMA [6], en este punto el grupo de revisión analiza la documentación aportada y autoriza el paso a la siguiente fase. Esta documentación debe cumplir las normas especificadas en la metodología a seguir en el proyecto o estándares de la empresa.

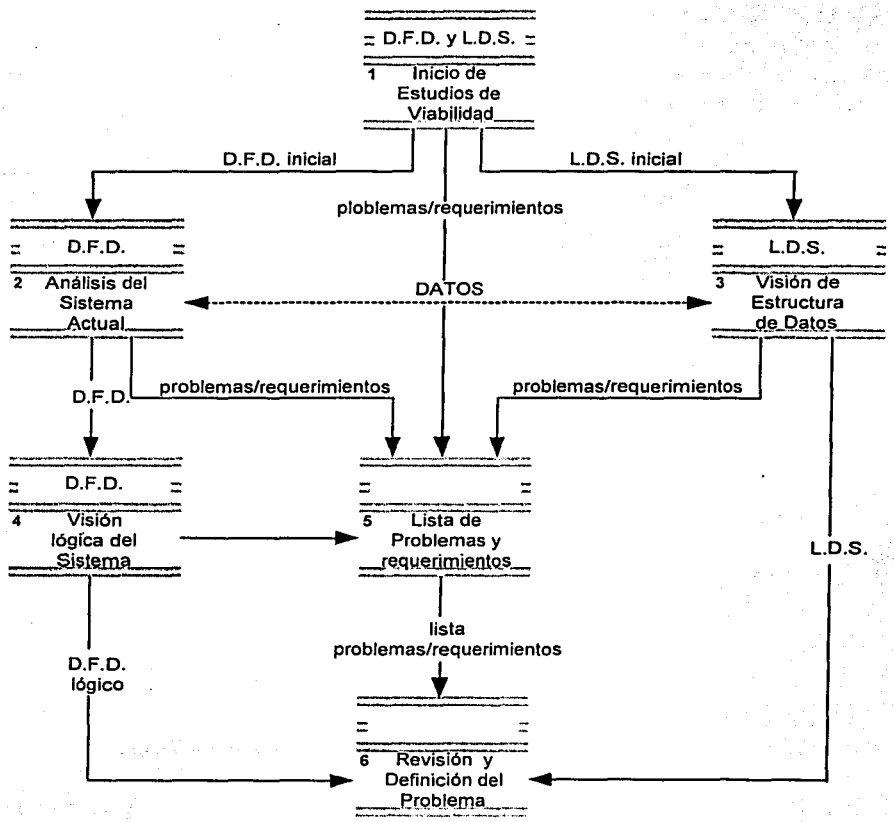


Figura 4.13 Definición del problema

FASE 1. ESTUDIO DE VIABILIDAD
ETAPA 02. IDENTIFICACIÓN DEL PROYECTO

En el resultado final de esta fase, puede ser que el proyecto y su desarrollo no sean aconsejables. Es por eso que esta fase se considera previa al inicio del desarrollo, es decir, a la fase de análisis y en la numeración de etapas, se considera siempre dividida en dos etapas. A continuación se describen brevemente los pasos que conforman a la etapa 02 (ver figura 4.14) [López, 1991].

- IDENTIFICACIÓN DE OPCIONES DEL PROYECTO [7], define las diferentes opciones para la resolución del problema. Se trabajará con un número de opciones que deben estar entre 3 y 6.

- CREAR UN ESQUEMA DE LAS ESPECIFICACIONES DEL PROYECTO [8], con el fin de ampliar el esquema que se tiene actualmente del proyecto con especificaciones lo suficientemente detalladas como para poder evaluarlo. Esto habrá que hacerlo para cada una de las opciones que se consideran como posibles.

- EVALUACIÓN DE LAS OPCIONES DEL PROYECTO [9], su objeto es la elección de una de las posibles opciones a desarrollar. De esta evaluación puede surgir el hecho de que ninguna de las opciones sea aceptada y por lo tanto el proyecto será considerado como viable.

- INFORME DEL ESTUDIO DE VIABILIDAD [10], tiene cómo objetivo dar forma a un informe en el cual figure toda la documentación de los trabajos realizados durante la fase previa de Estudio de Viabilidad.

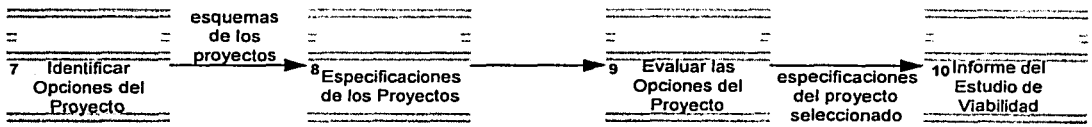


Figura 4.14 Identificación del proyecto

4.4.2 FASE 2. ANÁLISIS
ETAPA 01. ANÁLISIS DE LA SITUACIÓN ACTUAL

En esta fase se hace la revisión del análisis de requerimientos y comprobación de que se refleje la situación actual del sistema. Identificar las áreas de análisis y preparar un plan de entrevistas, es muy importante hacer una lista de requerimientos y recibir la conformidad con el plan por parte de la Dirección del Proyecto, juntamente dar un nivel de prioridad a cada elemento de la lista para discutirlos con la Dirección.

Recordar que se tiene que actualizar la lista de requerimientos para ver si ha surgido uno nuevo y compararla periódicamente con la lista inicial. Así mismo, se tiene que ir validando los DFD que existan más los que se generen posteriormente. También se tiene que anotar las entidades y sus atributos más importantes en el Diccionario de Datos (DD). A continuación se describen brevemente los pasos que conforman a la etapa 01 (ver figura 4.15) [López, 1991].

- INICIO DEL ANÁLISIS [11], establece el marco en que se va a desarrollar esta fase con sus tres etapas asociadas: análisis del sistema actual, especificación de requerimientos y selección de opciones técnicas.

- INVESTIGAR EL SISTEMA ACTUAL [12], su objeto es analizar el sistema actual, identificar sus procesos y realizar los diagramas de flujo correspondientes al mismo.

- INVESTIGAR LA ESTRUCTURA DE DATOS DEL SISTEMA [13], su objetivo es identificar las entidades mayores y sus relaciones en el sistema actual.

- DESARROLLO DE LISTAS DE PROBLEMAS Y REQUERIMIENTOS [14], en este punto se debe revisar la lista y las actualizaciones hechas durante la fase de análisis del sistema actual.

- REVISAR LOS RESULTADOS DEL ANÁLISIS [15], su objetivo es revisar que la documentación de toda la fase está realizada de acuerdo a los estándares y normas establecidos.

FASE 2. ANÁLISIS
ETAPA 02. ESPECIFICACIÓN DE REQUERIMIENTOS

En esta etapa se optimizan las descripciones de funciones elementales para conseguir su consistencia con los modelos de datos realizados. A continuación se describen brevemente los pasos que conforman a la etapa 02 (ver figura 4.16) [López, 1991].

- DEFINIR EL SISTEMA LÓGICO [16], su objetivo es producir una visión lógica del sistema actual por medio de DFD, simplificando flujos y procedimientos.

- DEFINICIÓN DE REQUERIMIENTOS DE SEGURIDAD Y CONTROL [17], su objetivo es hacer una lista de requerimientos en estos aspectos:

  - Identificar los requerimientos de Confidencialidad
  - Identificar los requerimientos de Integridad
  - Identificar los requerimientos en cuanto a aspectos de retención de datos en actualización
  - Identificar requerimientos de Control

- Identificar requerimientos de vuelta atrás y recuperación

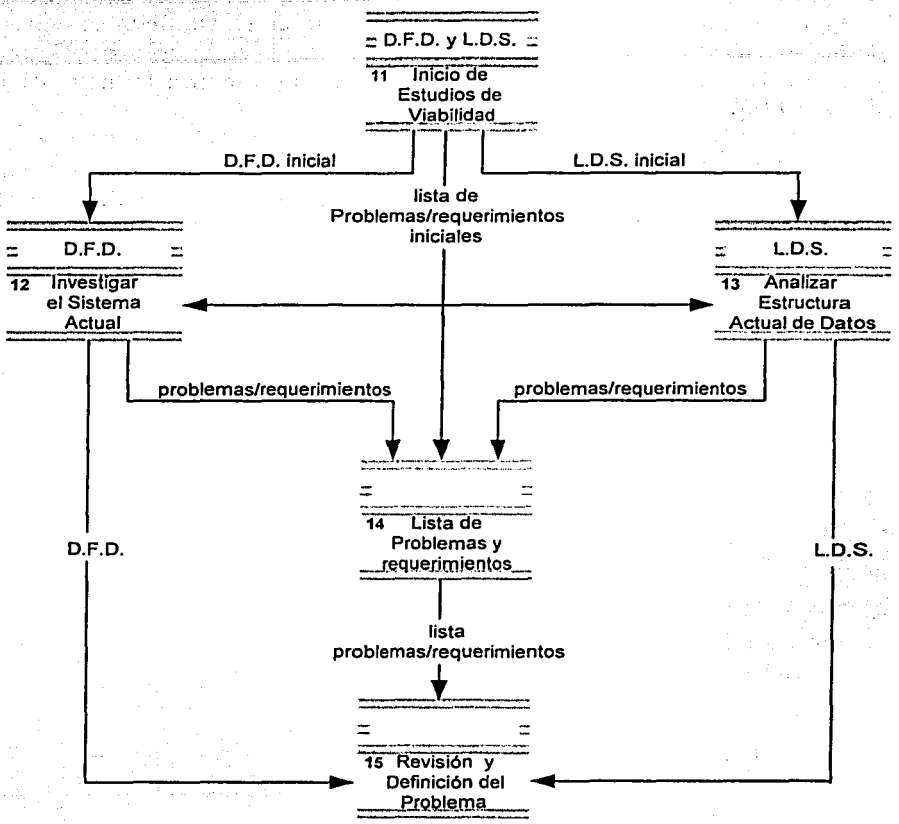


Figura 4.15 Análisis de la situación actual

- IDENTIFICAR Y SELECCIONAR LAS OPCIONES DEL SISTEMA [18], su objetivo es especificar el perfil del sistema deseado. Es uno de los pasos que se siguen hasta seleccionar cual de las opciones es la que mejor se ajusta a las características de la empresa, considerando aspectos como funcionalidad e impacto en la organización, análisis costo/beneficio, etc..

- DEFINIR LA OPCIÓN ELEGIDA EN DETALLE [19] y [20], su objeto es hacer las especificaciones detalladas del sistema usando las técnicas de los flujos de datos.

- CREAR LA ESTRUCTURA DE DATOS REQUERIDA [21], su objeto es crear la estructura lógica de datos necesaria para cumplir los requerimientos del sistema elegido. Debe cumplir por completo los requerimientos del sistema seleccionado. Se completarán las descripciones de las entidades.

- Revisar cada entrada de la lista de problemas y requerimientos, viendo si se resuelve en el sistema propuesto modificando, si es preciso: LDS y Entidades
- Completar la descripción de entidades con todos sus atributos, especificando el formato de cada uno de ellos
- Introducir todas las entidades y atributos en el DD

- INVESTIGAR LOS DETALLES DEL SISTEMA LÓGICO [22], su objetivo es analizar la vida de cada entidad del sistema, validando DFD y LDS, para sentar las bases para el diseño de procesos. En este paso de harán la ELH de las entidades, es decir, se analizará la historia de la vida de las entidades en el sistema.

- REVISAR LAS ESPECIFICACIONES DEL SISTEMA REQUERIDO [23], su objetivo es formalizar una revisión del material obtenido y conseguir una autorización por parte de usuarios y dirección para seguir con la siguiente fase.

FASE 2. ANÁLISIS
ETAPA 03. SELECCIÓN DE OPCIONES TÉCNICAS

En esta etapa se revisa la lista de requerimientos con objeto de comprobar si las opciones posibles cumplen dichos requerimientos. Preparar el plan de presentación, es decir, una serie de reuniones destinadas a informar a los implicados. Distribuir copias de las especificaciones técnicas de la opción elegida. Ampliación de información por parte de los analistas en cuento a las especificaciones o presentaciones; así como, retocar las opciones según criterio de los usuarios acerca de la recopilación de información. A continuación se describen brevemente los pasos que conforman a la etapa 03 (ver figura 4.17) [López, 1991].

- CREAR OPCIONES TÉCNICAS [24], su objeto es identificar y especificar las posibles implementaciones físicas del sistema. Las opciones tanto de hardware como de software ha debieron ya ser contempladas en el estudio de viabilidad, pero en este paso debe hacer un menú de los diferentes enfoques documentando cada uno de ellos.

- SELECCIÓN POR LOS USUARIOS DE LA OPCIÓN TÉCNICA [25], su finalidad es la de elección de una de las opciones documentadas en el paso anterior. Para realizar esta elección se valorarán:

  - Coste de desarrollo y explotación
  - Tiempo de desarrollo
  - Facilidad de implantación
  - Beneficios obtenidos
  - Impacto en la organización

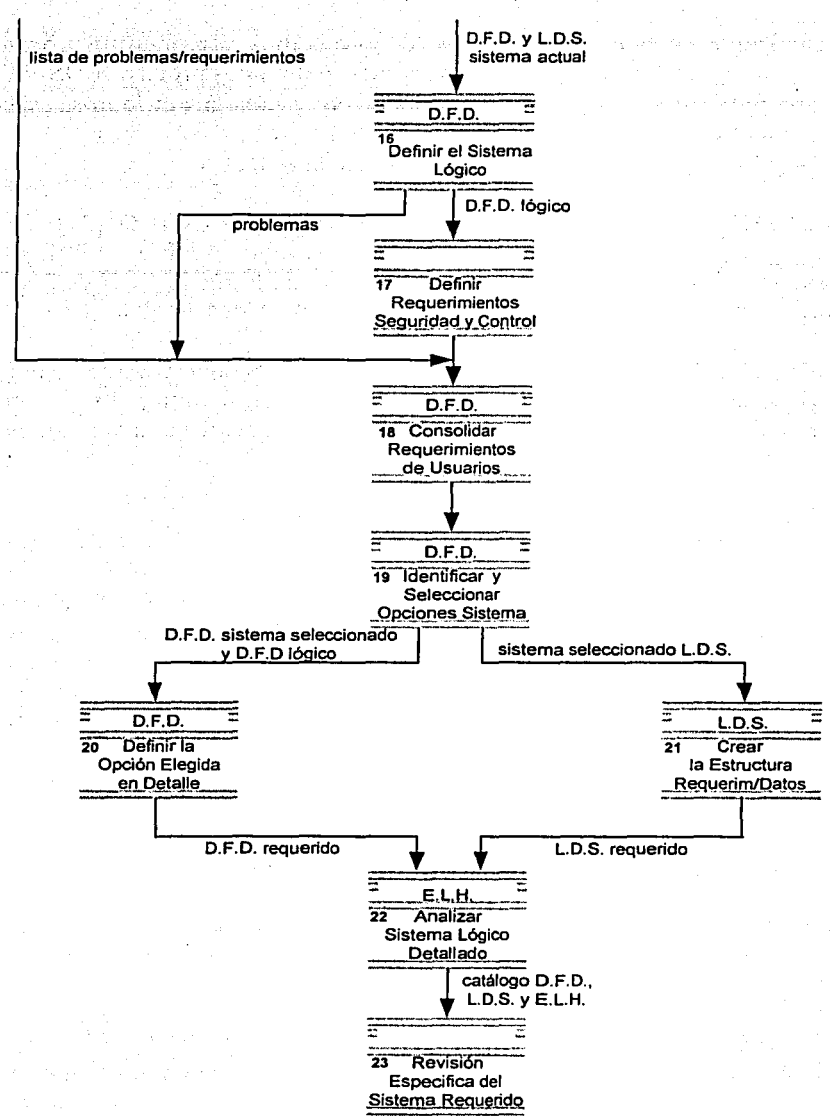


Figura 4.16 Especificación de requerimientos

- CONTEMPLAR Y REVISAR LAS ESPECIFICACIONES DEL SISTEMA REQUERIDO [26], debido a los cambios que han podido surgir al elegir la opción deseada, hay que fijar las especialidades técnicas de esa opción final de cara a preparar el diseño físico.

- DEFINIR LOS OBJETIVOS DEL DISEÑO [27], fijando criterios de rendimiento del sistema para hacerlo efectivo. Serán requerimientos fijados por los usuarios y condicionarán el diseño físico.

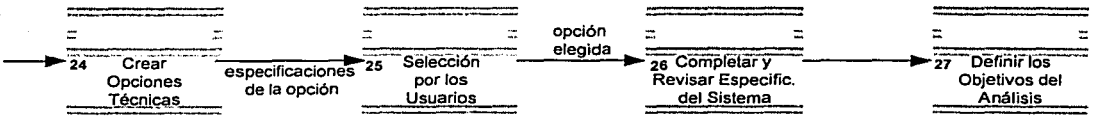


Figura 4.17 Selección de opciones técnicas

4.4.3 FASE 3. DISEÑO
ETAPA 04. DISEÑO DE DATOS
ETAPA 05. DISEÑO DE PROCESOS

Las siguientes dos etapas tratan de la selección de las fuentes de información, es decir, las entradas del sistema, anotando las especificaciones de las mismas. Optimizar las relaciones que se hagan y pasar a un conjunto de relaciones en TFN; A partir de ello construir una estructura de datos en TFN y actualizar los DFD necesarios, así como, el DD. A continuación se describen brevemente los pasos que conforman las etapas 04 y 05 (ver figura 4.18) [López, 1991].

- HACER UN ANÁLISIS RELACIONAL DE DATOS (RDA) [28], su objetivo es pasar del modelo de datos del sistema requerido a un conjunto de relaciones en tercera forma normal. Se partirá del estructura lógica de datos (LDS) se analizarán las fuentes de información del sistema tratando de hacer agrupaciones de datos e identificando las relaciones entre ellos.

  - Seleccionar las fuentes de información, es decir, las entradas del sistema
  - Anotar especificaciones de las mismas
  - Hacer un plan para llevar a cabo el RDA
  - Llevar a cabo el RDA con cada fuente
  - Optimizar el resultado del RDA y pasar a un conjunto de relaciones en tercera forma normal
  - Cargar el resultado del RDA en el diccionario de Datos

- HACER EL DISEÑO LÓGICO DE DATOS [29], su objetivo es optimizar un diseño lógico de datos que servirá como base para el diseño físico, es decir, realizar el CLDD.

  - A partir de las relaciones optimizadas en 3FN, construir una estructura de datos en 3FN
  - Comprobar esta estructura con la estructura lógica de datos requerida (LDS) y con la lista de funciones para ver requerimientos de acceso a los datos.

- Actualizar la descripción de entidades
- Actualizar el Diccionario de Datos

- DEFINIR LOS PROCESOS DE DIÁLOGO [30], a partir del catálogo de lecturas y actualizaciones se comprobarán los formatos de entradas/salidas, diálogos lógicos y controles lógicos de diálogos.

  - Comprobar los formatos de entradas/salidas
  - Actualizar el diccionario de Datos
  - Para los procesos en tiempo real, comprobar el diálogo lógico y sus controles

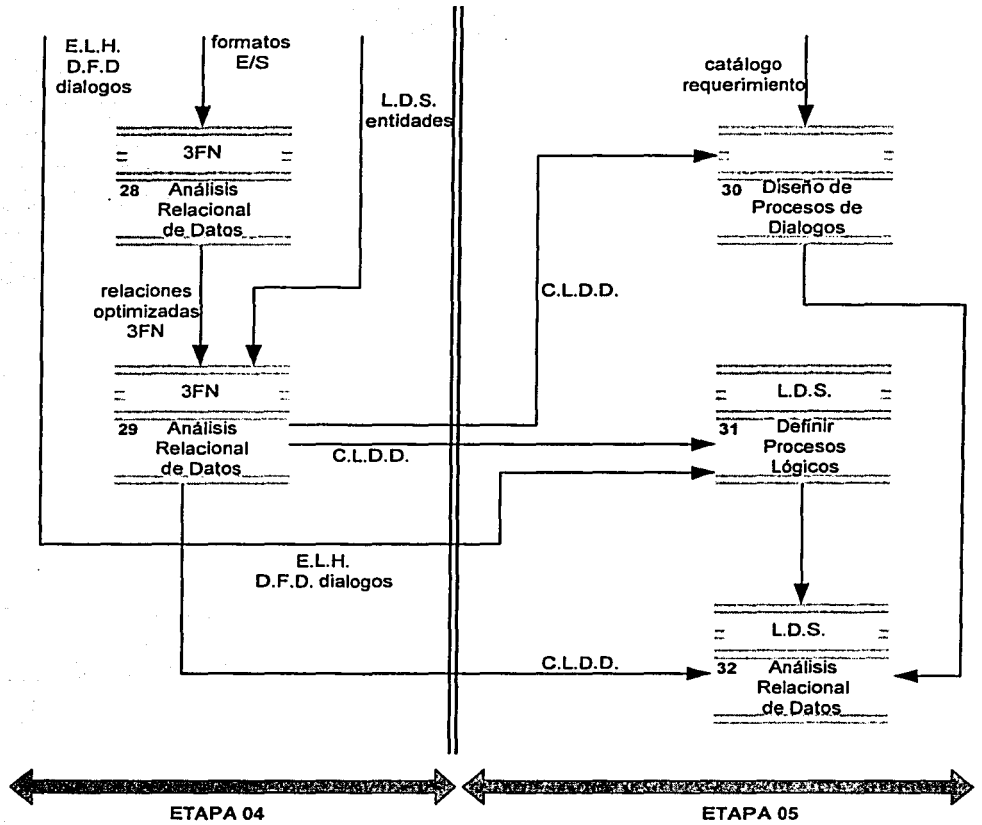


Figura 4.18 Diseño de datos y procesos

- DEFINIR LOS PROCESOS LÓGICOS DE ACTUALIZACIÓN [31], describir los procesos asociados a cada suceso del sistema.

- REVISAR Y VALIDAR EL DISEÑO LÓGICO DEL SISTEMA [32], para comprobar la integridad del sistema antes de pasar al diseño físico. Haciendo revisión lógico del sistema.

  - Diseño Lógico de Datos
  - Descripción de entidades
  - Catálogo de funciones
  - Esquemas de procesos
  - Manejo de errores
  - Formatos de entrada/salida
  - Esquemas lógicos de diálogos
  - Controles de diálogo

FASE 3. DISEÑO
ETAPA 06. DISEÑO FÍSICO DE DATOS Y PROCESOS

En esta etapa se describirán tanto las pantallas de menú como las de diálogo. Verificar la secuencia con los DFD del sistema y crear las especificaciones físicas de los procesos. Es importante recordar en esta etapa que a partir de la primera aproximación al diseño físico, hacer una planificación del espacio de almacenamiento necesario, de acuerdo con el sistema gestor de bases de datos empleado y teniendo en cuenta el espacio para índices, punteros, "overflow" y el reservado para sucesivas actualizaciones.

Modificar el diseño físico para recoger objetivos no contemplados en el diseño inicial, actualizando también la descripción de entidades y relacionar los objetivos de diseño no cubiertos con el diseño físico.

Así mismo, se tienen que consultar los cambios previstos con los responsables de los usuarios, modificando el diseño físico de acuerdo a esto. Asegurarse de que los objetivos son cubiertos con los cambios realizados, por lo cual hay que revisar programas y transacciones (recordar que todos estos cambios de diseño afectan también al DD).
A continuación se describe brevemente los pasos que conforman la etapa 06 (ver figura 4.19) [López, 1991].

- CREAR UNA PRIMERA VISIÓN DEL DISEÑO FÍSICO DE DATOS [33], su objetivo es convertir el modelo lógico de datos en un modelo físico.

  - Fijar las reglas de conversión para una primera aproximación al diseño físico teniendo en cuenta el gestor de base de datos a utilizar (DBMS)
  - Aplicar estas reglas al diseño lógico de datos para producir un primer diseño físico
  - Actualizar el Diccionario de Datos para reflejar esta primera aproximación

- CREAR LAS DESCRIPCIONES DEL PROGRAMA PARA LAS TRANSACCIONES PRINCIPALES [34], se trata de ir clasificando los esquemas de procesos a realizar según vayan a ser procesos "batch" o en tiempo real [López, 1991].

  - Especificar en el catálogo de funciones si van a suponer procesos batch o de tiempo real. Detectando procesos críticos para estudiar la forma especial los tiempos de respuesta
  - Definir los programas en tiempo real o transacciones. Se realizan las especificaciones físicas del proceso, detallado.
    - Secuencia de operaciones
    - Lecturas lógicas
    - Manipulaciones
    - Almacenamientos

- HACER PREDICCIONES DE RENDIMIENTO Y AJUSTAR EL DISEÑO [35], afinar el primer diseño físico según requerimientos de acceso y rendimiento. En esta fase, consultando con los usuarios se detectarán puntos críticos de proceso que pueden provocar variaciones en el diseño físico para mejorar tiempos de respuesta o proceso.

- CREAR LAS DEFINICIONES DEL FICHERO Y BASES DE DATOS [36], esto servirá de base a las siguientes fases de desarrollo. Hay que crear en el Diccionario las entradas correspondientes a la descripción de ficheros y bases de datos.

- COMPLETAR LAS ESPECIFICACIONES DE PROGRAMAS [37], las especificaciones serán diferentes dependiendo del entorno en el que se vayan a desarrollar los programas (lenguaje, máquina, etc). Hacer un esquema o plano de la secuencia de programas en cada ciclo de proceso indicando su periodicidad. Actualizar el DD.

- HACER UN PLAN DE PRUEBAS DEL SISTEMA [38], su objetivo es contemplar un conjunto de pruebas a realizar al sistema globalmente, comprobando la consistencia del sistema a través de unas pruebas de integración. Hacer una serie de pruebas para comprobar que se resuelven los problemas de la lista de requerimientos y de los DFD.

- REALIZAR LAS INSTRUCCIONES DE OPERACIÓN [39], su objetivo es realizar informes o manuales con las instrucciones de operación para cada ciclo de explotación del sistema, definiendo las necesidades de operación para cada uno de los programas en tiempo real y definiendo las instrucciones de operación para cada una de las cadenas de explotación del sistema (diarias, semanales, etc.).

- PLANIFICAR LA FASE DE IMPLANTACIÓN [40], teniendo en cuenta todas las tareas a realizar para la puesta en funcionamiento del sistema, planificar la

conversión de datos y hacer un plan combinado de implantación teniendo en cuenta los planes de pruebas de integración del sistema y pruebas de volumen .

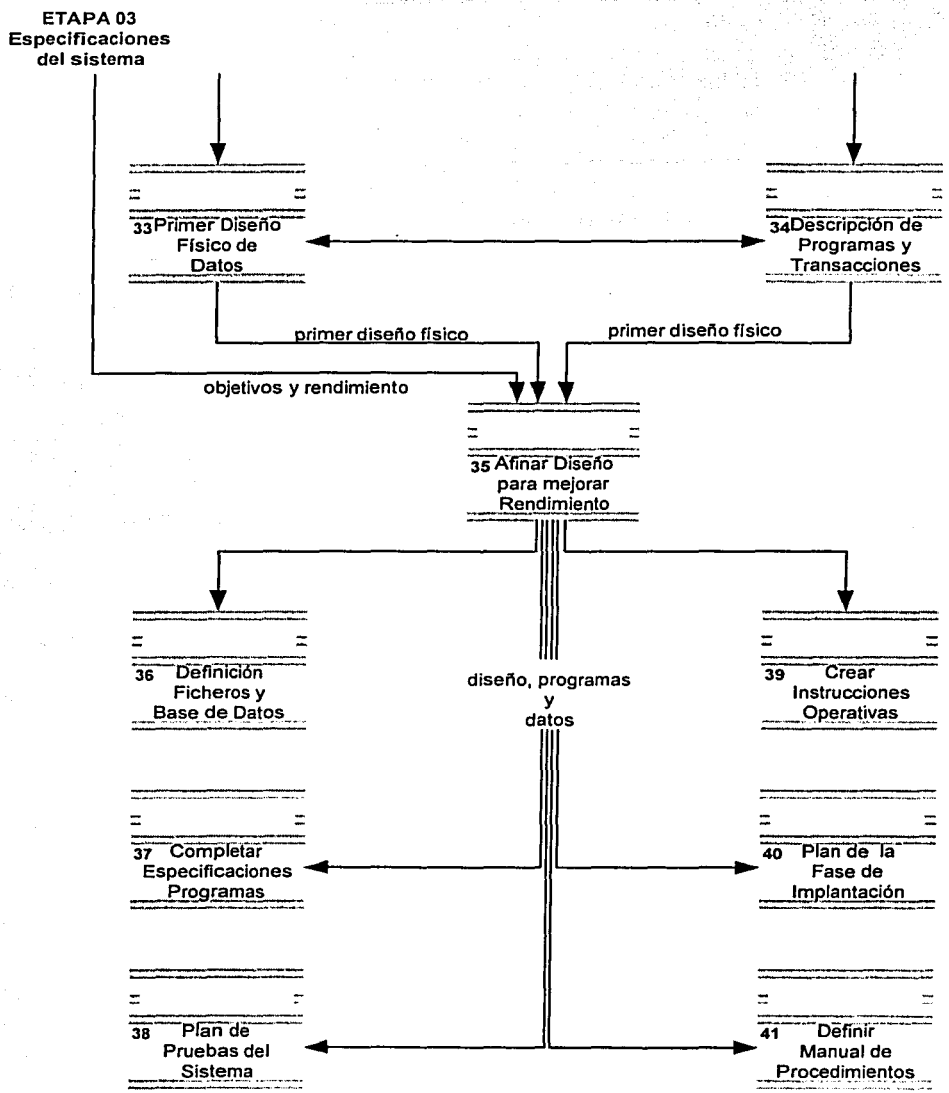


Figura 4.19 Diseño físico de datos y procesos

- REALIZAR EL MANUAL DE PROCEDIMIENTOS [41], definir en los DFD las entidades externas que supongan intercambio de información del sistema con el exterior. Definir los formatos finales de entradas y salidas, es decir, de pantallas y listados incluyendo las instrucciones para su utilización e interpretación por el usuario. Hacer una descripción de los diálogos o secuencias de pantallas de acuerdo a los informes de diálogos lógicos, controles de los mismos y descripción de procesos. Recordar que se tiene que definir a detalle los métodos de entrada, las instrucciones para el uso de operaciones de vuelta atrás y recuperación, así como, las instrucciones de operación a realizar por los usuarios.

4.4.4 VENTAJAS E INCONVENIENTES

De lo ventajoso que resulta el uso de una metodología (cualquiera), se puede decir que con SSADM nos encontramos ante algunas ventajas importantes:

- La alta participación del usuario en la definición de los modelos asegura que el sistema diseñado coincida con el sistema requerido.

- Nos proporciona tres vistas independientes del sistema:

  1. La funcionalidad, a través de los diagramas de flujo de datos (DFD), donde vemos los procesos que se desarrollan en nuestro sistema.
  2. Los datos con un estudio exhaustivo del modelo (técnicas "top-down" y "bottom-up").
  3. La lógica de nuestro sistema con una técnica que prácticamente es utilizada sólo por esta metodología: la historia de vida de la entidad.

- La separación de conceptos lógicos y físicos facilita en cierto modo la implantación del sistema analizado en distintos entornos.

- SSADM produce gran cantidad de documentación a lo largo de todas sus etapas, lo que supone una importante ayuda para el mantenimiento y producción del sistema diseñado.

Si hay que señalar algún inconveniente en la utilización de la metodología SSADM, quizá sea el carácter rígido y exigente de su origen británico. Cada paso de cada etapa está rigurosamente definido, así como, los subproductos que se van a ir obteniendo en el camino. Las técnicas de estudio de datos y procesos son similares a las de otras metodologías, pero aporta las suyas propias, lo que requiere una formación y entrenamiento adicionales si se pretende seguir el método al pie de la letra. Habitualmente se recomienda una adaptación de la metodología de acuerdo a las necesidades de la organización, de forma que resulte algo más ligera [Piattini, 1995].

4.5 METODOLOGÍA MÉTRICA

El consejo superior de información de Madrid, creado en 1983, encargado de elaborar y desarrollar las directrices en materia de tecnologías de la información y comunicaciones para la Administración, acordó en 1989 la realización del proyecto MÉTRICA. Se trataba de obtener un marco común de planificación y desarrollo de sistemas de información en la Administración del Estado. Cómo complemento, se elaboraron, además, cinco guías para su utilización en la planificación y desarrollo de sistemas de información de la Administración [Piattini, 1995].

- Plan director de sistemas de información.
- Plan de sistemas de información.
- Análisis de sistemas de información.
- Diseño de sistemas de información.
- Gestión de proyectos.

Y una sexta que incluía técnicas de planificación, análisis y diseño. Al no ser el objetivo principal del proyecto la obtención de una metodología completa, las guías carecían de algunos aspectos a la hora de ser aplicadas a la realización de sistemas de información. No obstante, dada la carencia de otros manuales para el desarrollo de sistemas de información, las guías MÉTRICA gozaron de cierta aceptación, a pesar de ser precisa la realización de una adaptación adicional para su utilización práctica [Piattini, 1995].

En diciembre de 1991, el Consejo Superior de Informática decidió revisar el proyecto MÉTRICA, realizando la segunda versión de la metodología de planificación y desarrollo de sistemas de información. Entre las razones que movieron a la elaboración MÉTRICA Versión 2, cabe citar las siguientes:

- Responder a la demanda por parte de los centros informáticos de una referencia para el desarrollo de sistemas de información.
- Dar continuidad a otras iniciativas del consejo referentes al desarrollo de sistemas de información realizadas con anterioridad, como el Plan General de Garantía de Calidad, aplicable al desarrollo de equipos lógicos.
- Mejorar la versión anterior de MÉTRICA, realizada en 1989.
- Contar con una especificación compatible con EUROMETODO, cuya versión definitiva no estaría disponible hasta finales de 1995.
- Aprovechar la situación del mercado de herramientas de ayuda al desarrollo de sistemas de información, cuya oferta en 1992 era mucho más amplia que cuando se realizo la versión anterior de MÉTRICA.

El proyecto MÉTRICA Versión 2 se realizó en el primer semestre de 1992, y constaba de tres partes:

- Desarrollo de la metodología.
- Formación.

- Proyectos piloto.

La metodología MÉTRICA Versión 2 se compone de tres guías [Piattini, 1995].

Guía de referencia.

Contiene los elementos fundamentales de la metodología. Explica qué hay que hacer al desarrollar un sistema y en que momento se deben realizar las tareas previstas. Indica quienes son los responsables de las tareas y sus funciones, así como, las técnicas a utilizar para realizar las tareas y los productos a obtener como resultado.

Los pasos a seguir para el desarrollo de sistemas de información se han estructurado en fases, módulos, actividades y tareas, facilitando así la planificación, gestión y control del proyecto, ya que se establecen conjuntos formales de productos e hitos de aceptación de los resultados obtenidos.

Guía de técnicas.

Recoge en su mayoría técnicas estructuradas para el desarrollo de sistemas de información ya consolidadas. En esta guía se describen con detalle el funcionamiento y objetivo de las técnicas, sus reglas y restricciones. Además, del significado semántico de los símbolos gráficos utilizados. Se incluyen consejos prácticos para su utilización ejemplos y la referencia a las actividades de la metodología en las que se utiliza la técnica correspondiente.

Guía de usuario

Es un manual de consulta rápida que resume el contenido de la Guía de referencia, y será utilizado como recordatorio por los miembros del equipo del proyecto una vez se hayan estudiado los conceptos y las técnicas que constituyen MÉTRICA Versión 2.

4.5.1 TÉCNICAS

En MÉTRICA Versión 2 se hace énfasis en la utilización de técnicas gráficas con el fin de describir los sistemas de forma sencilla y mejorar la comunicación entre los desarrolladores y los usuarios. Las imágenes a utilizar en las técnicas pueden ser alteradas, siempre y cuando sigan describiendo su contenido. Se facilita, por tanto, la implantación de las técnicas en cualquier organización, ya que se podrá utilizar cualquier herramienta CASE que las soporte.

Para cada técnica se definen los procedimientos que permiten utilizar la herramienta correctamente y se indica en qué fase, actividad o tarea se ha de utilizar [Piattini, 1995].

La técnicas incluidas son:

- Diagramas de flujo de datos.
- Modelado de datos.
- Modelo conceptual: Diagrama de estructura de datos.
- Modelo lógico: Normalización.
- Historia de la vida de las entidades.
- Diseño estructurado.
- Diagramas de estructura de cuadros de Constantine.
- Entrevista.
- Análisis Coste-Beneficio.
- Pruebas.
- De caja blanca y de caja negra.
- Unitarias y de integración.
- Del sistema y de aceptación.
- Factores críticos de éxito.
- Técnicas matriciales.

Diagrama de flujo de datos. Delimita el sistema en estudio, describe el movimiento y transformación de los datos a través del sistema. Los niveles considerados son: diagrama de contexto, subsistemas, funciones, subfunciones asociadas a eventos y procesos. Los elementos del diagrama son: entidades externas procesos, almacenes de datos y flujos de datos [Piattini, 1995].

Modelado de datos. Como representación del modelo conceptual, se utiliza el modelo entidad/relación de Chen, que, por ser más cercano al mundo real y admitir relaciones n-arias, se utiliza en la representación del sistema de alto nivel en la fase de planificación, y para la fase de análisis se utiliza el diagrama de estructura de datos. Para representar el modelo lógico se normalizan los datos hasta la tercera forma normal.

Historia de la vida de las entidades. Complementa la representación dinámica y estática de los procesos y los datos, sirviendo para verificar su exactitud y garantizando la coherencia a través del estudio de los eventos siguiendo la evolución de las entidades de datos a lo largo del tiempo.

Diagrama de estructura de cuadros. A partir de los diagramas de flujo de datos, se indica cómo obtener el diseño de la arquitectura de componentes e interfaces y el diseño detallado. En la fase de diseño, para optimizar el modelo de datos quizá será necesario desnormalizar dicho modelo.

Se han incluido también técnicas de pruebas (de caja blanca, de caja negra, unitarias, de integración de componentes, del sistema y de aceptación), entrevistas, análisis coste-beneficio, factores críticos de éxito y técnicas matriciales [Piattini, 1995].

4.5.2 FASES

La metodología MÉTRICA Versión 2 cubre el ciclo de vida completo de los sistemas de información, para lo cual se han incluido las fases de planificación, análisis, diseño, construcción e implantación. Por su complejidad, y para agrupar tareas semejantes, las fases de análisis y construcción constan de dos módulos (ver tabla 4.2).

FASES	MODULOS
Planificación	P.S.I. Plan de sistemas de información A.R.S. Análisis de requisitos del sistema
Diseño	E.F.S. Especificación funcional del sistema D.T.S. Diseño Técnico del sistema
Construcción	D.C.S. Desarrollo de componentes del sistema D.P.U. Desarrollo de procedimiento de usuario
Implantación	P.I.A. Pruebas, implantación y aceptación del sistema

Tabla 4.2 Fases y módulos de la metodología MÉTRICA V2

Los módulos se subdividen en actividades, y estás, en tareas. En cada tarea se especifican objetivos, los productos a obtener y la técnica a utilizar [Piattini, 1995].

Planificación. Se definirá qué información es precisa para conseguir los objetivos estratégicos de la organización, definiendo de manera general su arquitectura (procesos y datos) y los nuevos sistemas a desarrollar. En el caso de que sólo se tratara de un sistema, no se realizaría esta fase.

Análisis. Se definirá el alcance y objetivo del proyecto de realización del sistema y sus requisitos. En primer lugar, se realizará un estudio del modelo lógico del sistema existente para conocer sus deficiencias. Se dialogará con los usuarios para conocer las necesidades. Se obtendrán distintas alternativas que den solución al problema, recomendando una de ellas.

Una vez elegida la opción, se describirán con detalle las especificaciones formales para obtener la aprobación por parte de los usuarios. Todo ello desde el punto de vista lógico, sin tener en cuenta ninguna consideración física, gracias a lo cual, si en el futuro cambian las condiciones del entorno físico del sistema (por ejemplo hardware, base de datos, etc.) el modelo permanecerá invariante [Piattini, 1995].

Diseño. Se obtendrá el conjunto de componentes del sistema a construir y un plan de pruebas inicial del sistema mediante el diseño de la arquitectura del sistema (los módulos, las interfaces: entre módulos, con otros sistemas y de usuario) y del modelo físico de los datos, así como, la especificación del entorno tecnológico y los requisitos de comunicaciones, operación, seguridad y control entre otros.

Construcción. Se desarrollarán y probaran todos los componentes del sistema documentando todos los programas y sus pruebas. Una vez realizadas las pruebas unitarias, se efectuaran y documentarán las pruebas de integración. El otro bloque de tareas de está fase está encaminado a obtener los procedimientos de usuario para la instalación, conversión de datos, operación y puesta en producción del nuevo sistema, con los que se confeccionará el manual de usuarios en el nuevo sistema y se elaborarán los materiales necesarios para realizar dicha formación [Piattini, 1995].

Implantación. Se trata de obtener la aceptación del sistema por parte de los usuarios, para lo cual se efectuarán las correspondientes pruebas del sistema y de aceptación. Por último, se instalarán los procedimientos manuales y automáticos precisos para el funcionamiento en producción del nuevo sistema y comenzar su explotación.

MÉTRICA Versión 2 permite, seleccionar las actividades necesarias en cada caso, sin tener que realizar todas obligatoriamente, adaptándose a distintos proyectos en función de su duración, complejidad o ciclo de vida [Piattini, 1995].

4.5.3 PARTICIPANTES Y FUNCIONES

Para cada proyecto que se realice con MÉTRICA Versión 2, se creará una estructura de participantes dependiendo de las características particulares de dicho proyecto.

Un comité de dirección encargado de dotar de los recursos necesarios y realizar revisiones; este grupo podrá estar encabezado por el director del proyecto [Piattini, 1995].

Un equipo del proyecto, que constará de un jefe del proyecto y un número variable de componentes en función del proyecto.

Se requerirá también la participación activa de un grupo de usuarios, que suministrará la información al equipo de proyecto sobre los requisitos del sistema y sus necesidades y validará los resultados una vez revisados [Piattini, 1995].

- Un departamento de tecnologías de la información o equivalente.
- Un Grupo de calidad, cuya misión será verificar que los trabajos de las fases se realizan de acuerdo a la metodología y que los productos obtenidos son conformes a los preescrito.
- La configuración mínima estaría formada por el Equipo de proyecto, el Grupo de usuarios y el Grupo de calidad, además de un Director del proyecto.

4.5.4 PRODUCTOS

Como resultado de las distintas tareas realizadas, se van obteniendo productos que pueden ser finales o intermedios; en este último caso, servirán para su utilización en tareas posteriores y se complementarán en la medida en que se cuente con una información más profunda (ver tabla 4.3) [Piattini, 1995].

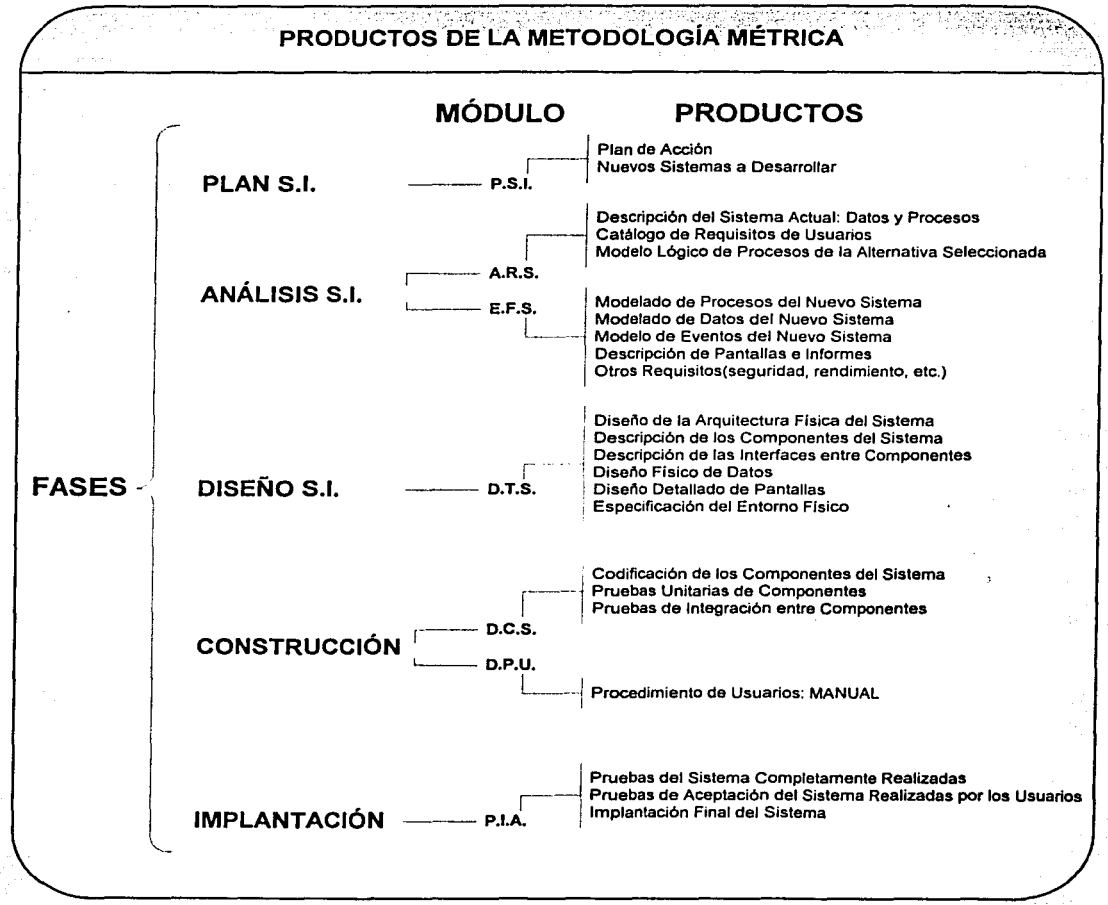


Tabla 4.3 Productos de la metodología MÉTRICA

4.5.5 METAS

MÉTRICA Versión 2 tiene por objetivo mejorar el proceso de producción de sistemas de información en los siguientes aspectos:

- Mayor respeto a plazos y costes previstos.
- Mayor satisfacción de los usuarios.
- Mayor calidad de los sistemas desarrollados.
- Ahorro posterior en concepto de mantenimiento.
- Procedimiento normalizado para el desarrollo.
- Ahorro en formación del personal.
- Aumento de la productividad.

Se planteó, por tanto, obtener una metodología que reuniera, entre otras, estas características:

- ABIERTA. Para utilizar por la Administración del Estado, otras administraciones y cualquier empresa o centro que la quiera adoptar.

- FORMAL. Que estableciera los pasos necesarios para la construcción de sistemas de información.

- CONSTRUIR CON CALIDAD. Mediante su utilización, se obtuvieron sistemas de mayor calidad.

- FLEXIBLE. Adaptable a distintos entornos, proyectos de tamaño y complejidad diversa y a varios ciclos de vida de sistemas de información de gestión.

- SENCILLA. Que fuera práctica y de fácil utilización.

4.6 METODOLOGÍA JAMES MARTIN

La ingeniería de información desarrollada inicialmente por James Martin, es una de las metodologías más populares y actualmente existen también otras metodologías inspiradas en ella. No obstante, muchas de las metodologías basadas en ingeniería de información han incorporado algunas variantes, ampliando nuevos conceptos según la empresa consultora que pretende ofrecer esta versión privada de la metodología. James Martin, aunque ofrece una metodología pública mediante los distintos libros publicados, también distribuye una versión de la metodología a través de algunas de sus empresas asociadas [Piattini, 1995].

En este apartado se explica la metodología de Martin según la versión pública, haciendo énfasis en los puntos más importantes de su evolución. MARTIN (1991) define la ingeniería de la información como la aplicación de un conjunto interrelacionado de técnicas formales para la planificación, análisis, diseño y construcción de sistemas de información en toda la organización o en un área de la misma [Piattini, 1995].

Por supuesto, Martin también comenta que la aplicación de la ingeniería de la información se basa hoy día en un repositorio de herramientas CASE. De lo contrario, el proceso puede llegar a ser muy complejo y pesado. La metodología pretende establecer desde la fase inicial de planificación un modelo de sistema de información basado en la estrategia de la organización, independientemente de la plataforma física. Esto permitirá establecer un sistema de información bastante estable a largo plazo (hasta cinco años) sin que los entornos físicos afecten al sistema [Piattini, 1995].

Las ideas de la metodología permiten una visión global de la organización, y por lo tanto contemplar con facilidad todas las mejoras que se pueden realizar a lo largo, medio y corto plazo [Piattini, 1995].

Las principales características de la metodología son:

I. Proporcionar un entorno metodológico para cada fase e integrar éstos con técnicas y herramientas dentro de una sola planificación del proyecto y una estructura de descomposición del trabajo de gestión.

II. Uso de técnicas que están integradas en cada fase de la metodología.

Con lo anterior, los objetivos generales de la ingeniería de la información dentro de una organización son [Piattini, 1995]:

a) Ayudar a conseguir y defender ventajas competitivas en su mercado mediante la identificación del uso estratégico de la tecnología de la información.

b) Proporcionar una arquitectura para el soporte integrado de la información ejecutiva, toma de decisiones y sistemas de información a nivel operativo o intermedio.

c) Soportar las necesidades de información de la gestión ejecutiva de una manera eficiente.

d) Enfocar los esfuerzos del grupo de sistemas de información de la empresa, relacionando el desarrollo del sistema de información con los objetivos del negocio de la empresa y los factores críticos de éxito.

e) Involucrar de una manera eficaz a los usuarios en la planificación, análisis, desarrollo e implantación de los sistemas de información mediante técnicas de entrevistas y diseño de prototipos.

f) Mejorar la calidad de los sistemas de información mediante el empleo de técnicas rigurosas a través de todo el ciclo de vida del desarrollo.

g) Disminuir el tiempo requerido para desarrollar nuevas aplicaciones, así como reducir los costes de mantenimiento de las ya existentes.

4.6.1 FASES

Estas fases son: planificación, análisis, diseño y construcción, que forman el ciclo de vida del desarrollo de los sistemas de información en la metodología. La figura 4.20 describe la típica pirámide de las fases [Piattini, 1995].

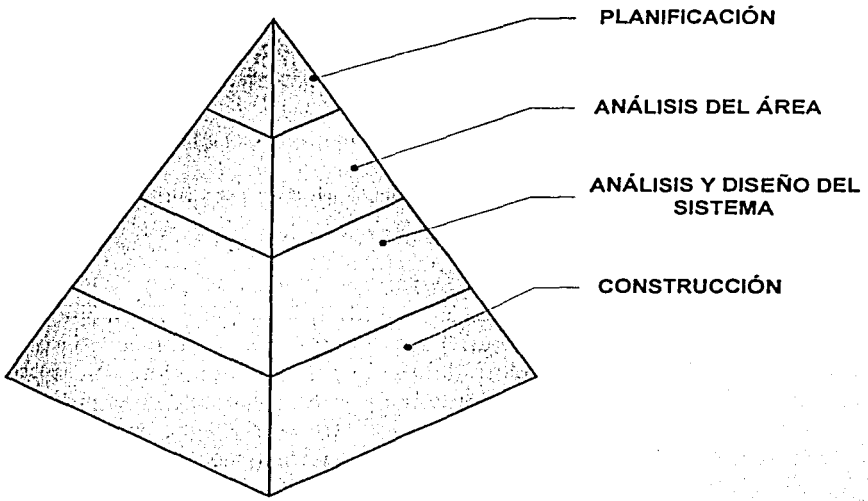


Figura 4.20 Pirámide de la ingeniería de la información

I. La fase de planificación consiste en la elaboración de un modelo de Empresa que soporte los objetivos del negocio y el desarrollo de planes de sistemas de información estratégico táctico de la empresa.

II. La fase de análisis consiste en la formalización de los requisitos de información de un área de negocio y realización de un modelo conceptual que los soporte. Define las aplicaciones del área del negocio [Piattini, 1995].

III. La fase de diseño consiste en la transformación del modelo conceptual de la aplicación en una especificación de datos y procesos. Define el entorno y las pruebas de la aplicación.

IV. La fase de construcción e implantación consiste en desarrollar la nueva aplicación, probarla e implantarla en la organización.

Se puede describir la metodología como una manera de trabajar que reúne el conjunto de información, datos o elementos en un repositorio. Todas las fases (modelos integrados) de la metodología acceden al tratamiento de esta información siempre a través del repositorio o diccionario, como se puede ver en la figura 4.21.

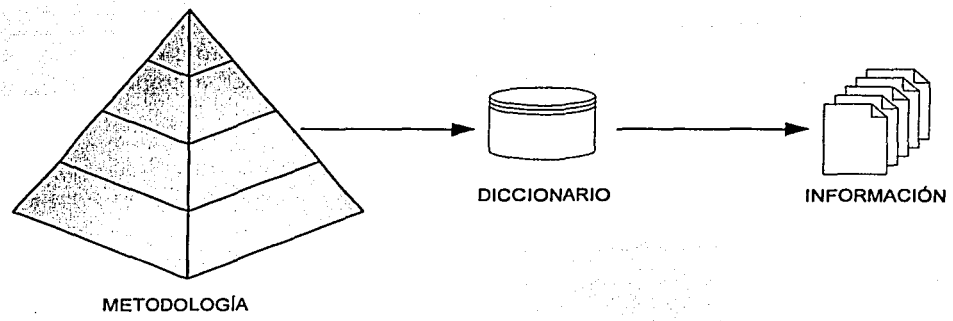


Figura 4.21 Papel del diccionario en la ingeniería de la información

4.6.2 OBJETIVOS DE LAS FASES

a) Los objetivos de la fase de implantación son: definición de la estrategia del sistema de información de acuerdo con los objetivos de la empresa, establecer los procedimientos de priorización, identificar las necesidades de información, identificar competidores, construir la arquitectura de información e incorporar aplicaciones existentes

b) Los objetivos de la fase de análisis son: entender las necesidades de información y consideraciones especiales del área de la empresa, generar informes relevantes, determinar si un paquete resolverá o soportará las necesidades y definir una solución que se pueda estimar e implantar [Piattini, 1995].

c) Los objetivos de la fase de diseño son: traducir los requerimientos del área de negocios en un diseño completo del sistema, diseñar una estructura de aplicación eficiente, determinar los requerimientos del paquete y producir una correcta planificación de la fase de construcción e implantación.

d) Los objetivos de la fase de construcción e implantación son: construir un sistema de alta calidad, asegurar una aceptación y transferencia satisfactoria del sistema al usuario y facilitar la evolución del sistema.

4.6.3 DESARROLLO

La ingeniería de la información utiliza una serie de técnicas. Los resultados son visibles mediante los productos finales que el método va produciendo a lo largo de las diferentes fases y productos a lo largo de las diferentes etapas [Piattini, 1995].

a) La fase de planificación utiliza las técnicas de diagrama de descomposición, diagrama de entidad/relación y la matriz de asociación.

b) La fase de análisis utiliza diagramas de entidad/relación, el diagrama de flujo de datos, el diagrama de descomposición y la matriz de asociación, especialmente sobre las entidades y procesos.

c) La fase de diseño utiliza el diagrama de estructura de datos, el diagrama de descomposición, y cómo técnica más importante, que es muy propia de la metodología, el diagrama de acción, que se explica a continuación [Piattini, 1995].

4.6.4 BENEFICIOS A CORTO Y LARGO PLAZO

Los beneficios a corto plazo son:

a) Las necesidades de una dirección para tener la información son normalmente muy rápidas de identificarlas [Piattini, 1995].

b) La atención primaria se concreta de forma ordenada sobre los objetivos, problemas y factores críticos de éxito.

Los beneficios a largo plazo son:

a) Se utiliza la información y la tecnología última para una mayor y mejor construcción de los sistemas y esto puede llegar a ser positivo para los sistemas [Piattini, 1995].

b) La dirección consigue una mejor comprensión del negocio mediante las técnicas utilizadas.

c) La misma información se utiliza en distintos lugares de manera integrada, y todo el mundo utiliza un mismo idioma para el trabajo.

El resultado final obtenido es el producto ideal. Para la obtención de este producto ideal, todas las fases controlan continuamente los niveles de calidad (no pasar de una fase a otra sin haber terminado la anterior de forma correcta) y los niveles de riesgo (tener identificados los problemas de información continuamente). Estos controles cumplen con la exigencia de calidad [Piattini, 1995].

4.6.5 METAS ALCANZADAS

a) Los resultados de la fase de planificación son: la definición formalizada de la empresa según su organización, estrategia, datos y procesos, obtener una arquitectura de sistemas de información necesaria para soportar las necesidades de la empresa, obtener un plan estratégico detallando la implantación de la nueva arquitectura del sistema de información y el compromiso de la dirección [Piattini, 1995].

b) Los resultados de la fase de análisis son: obtener los requerimientos del sistema de información para el área de la empresa, obtener el modelo de información del área, el diseño conceptual de las aplicaciones, la adquisición del paquete y el plan de desarrollo para el usuario.

c) Los resultados de la fase de diseño son: la obtención de algún prototipo del sistema; obtener el diseño del sistema y enfocarlo hacia la fase de construcción e implantación dependiendo de los factores externos (usuarios), internos (técnicos), el ambiente, las pruebas y las conversión de datos [Piattini, 1995].

d) Los resultados de la fase de construcción son: obtener un sistema correctamente instalado, obtener un plan de evolución del sistema y obtener la documentación completa del sistema.

e) Los objetivos de la fase de diseño son: traducir los requerimientos del área de negocios en un diseño completo del sistema, diseñar una estructura de aplicación eficiente, determinar los requerimientos del paquete y producir una correcta planificación de la fase de construcción e implantación.

f) Los resultados de la fase de diseño son: la obtención de algún prototipo del sistema; obtener el diseño del sistema y enfocarlo hacia la fase de construcción e implantación dependiendo de los factores externos (usuarios), internos (técnicos), el ambiente, las pruebas y las conversión de datos.

g) Los objetivos de la fase de construcción e implantación son: construir un sistema de alta calidad, asegurar una aceptación y transferencia satisfactoria del sistema al usuario y facilitar la evolución del sistema [Piattini, 1995].

h) Los resultados de la fase de construcción son: obtener un sistema correctamente instalado, obtener un plan de evolución del sistema y obtener la documentación completa del sistema.

4.7 METODOLOGÍA CASE

Las siglas CASE provienen del acrónimo de la expresión en inglés "Computer Aided Software Engineering", que podría traducirse como Ingeniería del Software Asistida por Computadora. Esta definición implica una forma concreta de entender la problemática del software: se trata de aplicar un enfoque de Ingeniería al desarrollo y mantenimiento de software contando con el apoyo de herramientas construidas, a su vez, con software. Estas herramientas permiten que las actividades propuestas por la Ingeniería del Software sean más eficaces y más fáciles de realizar.

La expresión herramienta CASE se ha convertido en un término bastante conocido por los profesionales dedicados al desarrollo y mantenimiento de software. Sin embargo, muchas de estas personas mantienen en su mente la imagen de una herramienta CASE como el clásico banco de trabajo de analista. Este tipo de herramientas también se conocen como CASE superior (upper CASE) o CASE frontal (front-end CASE) [Piattini, 1995].

CASE propone una nueva formulación del concepto de ciclo de vida del software, basada en la automatización. La idea básica que subyace en CASE es proporcionar un conjunto de herramientas bien integradas y que ahorren trabajo, enlazando y automatizando todas las fases del ciclo de vida del software.

La tecnología es una combinación de herramientas de software y de metodologías. Más aún, es diferente de las primitivas tecnologías del software y no solamente en la implantación de soluciones. Extendiéndose a todas las fases del ciclo de vida del software, es la tecnología de software más completa hoy en día. CASE aborda los problemas de la productividad del software durante todo el ciclo de vida, automatizando muchas de las tareas de análisis, diseño así como también las tareas de implantación y mantenimiento de los programas.

Como lo manuales de metodologías estructuradas son demasiado tediosas y de un trabajo muy intensivo, en la práctica raramente se siguen a un nivel más detallado. CASE hace prácticos los manuales de metodologías estructuradas al automatizar el dibujo de diagramas estructurados y la generación de la documentación del sistema.

La figura 4.22 muestra las ventajas que CASE ofrece a los profesionales del desarrollo de software. Esta exposición sugiere que no es enteramente una tecnología nueva, sino que está construida sobre técnicas y herramientas comprobadas en la práctica. En este sentido, puede definirse como un conjunto de conceptos estructurados y de metodologías con un nuevo envase. La novedad es la automatización [Piattini, 1995].

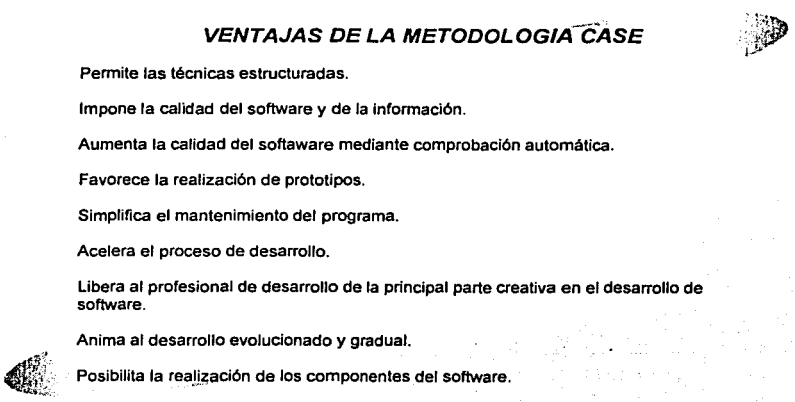


Figura 4.22 Las ventajas de la metodología CASE

4.7.1 ETAPAS EN LA METODOLOGÍA CASE

La metodología CASE se basa en un análisis y desarrollo del tipo descendente ("top-down") en que el ciclo de vida de un sistema se compone de las siguientes etapas. (Figura 4.23)

1. Estrategia
2. Análisis
3. Diseño

4.1 Construcción

4.2 Documentación

5. Transición

6. Producción

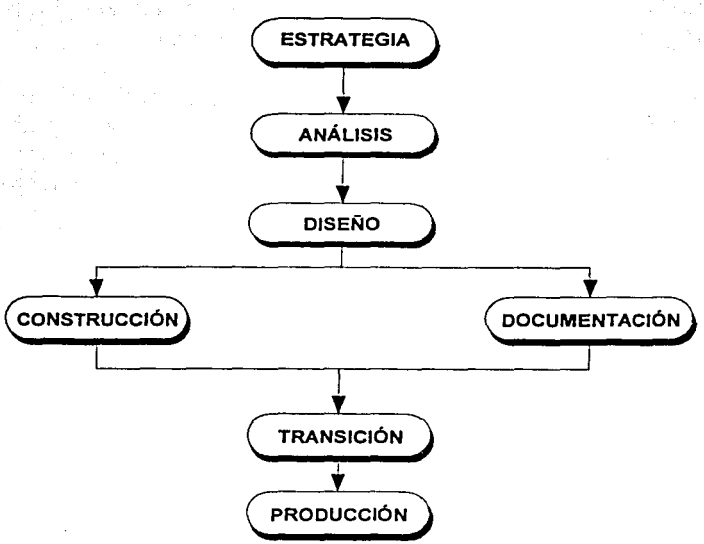


Figura 4.23 Etapas de la Metodología CASE

4.7.1.1 ESTRATEGIA

Esta es una de las etapas mas importantes, ya que tiene por objetivo lograr un entendimiento claro de las necesidades de la organización y del ambiente en que operará el sistema o sistemas a implantar.

Con el fin de tener una visión desde los puntos de vista de la dirección corporativa, se analizan las diferentes funciones que realiza la organización y sus necesidades de información a todos niveles, durante esta etapa se realizan una serie de entrevistas con la dirección y los responsables de los departamentos.

Así, a partir de esta información se realiza un primer modelado de los requerimientos del sistema de información adecuado a las necesidades de la organización. Posteriormente para la definición de una primera versión de la arquitectura del sistema, además de los requerimientos antes obtenidos, se toman en cuenta las tecnologías en ese momento disponibles y los sistemas de información ya existentes en operación. En la figura 4.24 se muestra este proceso.

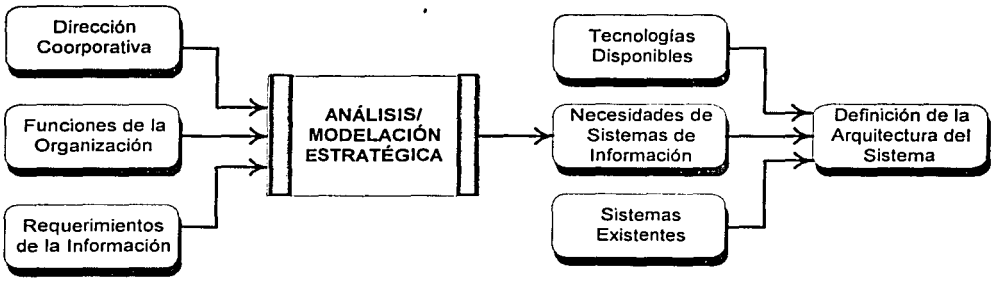


Figura 4.24 Etapa de Estrategia

Los resultados de esta etapa son, un conjunto de modelos de la empresa, un conjunto de recomendaciones, y un plan acordado de desarrollo de los sistemas de información, la elaboración de este último se hará de acuerdo las necesidades actuales y futuras de la organización, tomando en cuenta restricciones operativas, financieras y técnicas [Barker, 1992].

4.7.1.2 ANÁLISIS

La etapa de análisis toma y verifica los descubrimientos de la etapa de estrategia y expande estos en suficiente detalle, a través del análisis de funciones, de documentos y de datos, para asegurar la precisión de los modelos de la empresa, posibilitando un fundamento sólido para el diseño, dentro del alcance de la organización y tomando en cuenta sistemas existentes.

Con el fin de obtener un refinamiento de los modelos, se realizan otra serie de entrevistas ya no a un nivel directivo como en la anterior, sino a un nivel operativo y técnico. Con la participación los responsables de la operación de las funciones que serán automatizadas se realiza un análisis detallado de sus requerimientos específicos en cuanto a objetivos, subfunciones, información, datos, etc.

Así, en esta etapa, a partir de los modelos de la organización obtenidos en la descripción anterior y del producto del análisis de ésta, se genera el modelado del sistema. Los modelos básicos de esta etapa son:

El de entidad/relación, que modela mediante relaciones lógicas todos los datos involucrados en el sistema, de tal manera que cualquier tipo de explotación (consulta o modificación) sean posibles.

El funcional, que modela las diferentes servicios que ofrecerá el sistema mediante una organización y clasificación de las diversas funciones y subfunciones que fueron identificadas en el análisis. Como resultados de esta etapa, además del modelo de entidad/relación y el funcional, se definen las restricciones que tendrá el sistema y la estrategia que se seguirá en la etapa de transición. El proceso de esta etapa y sus resultados se muestran en la figura 4.25 [Barker, 1992].

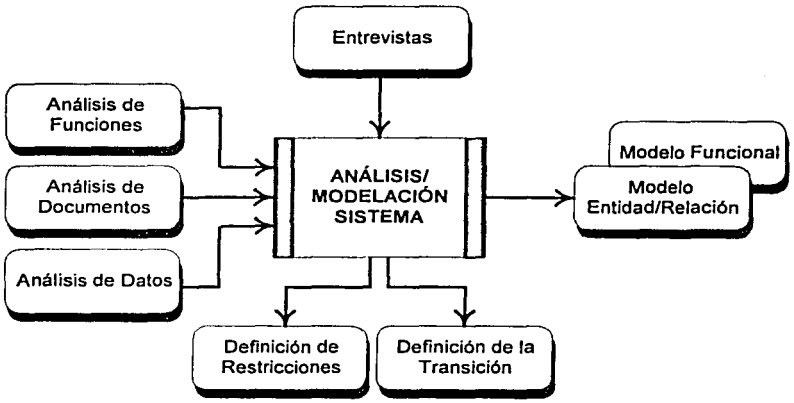


Figura 4.25 Etapa de Análisis

4.7.1.3 DISEÑO

La etapa de diseño toma los requerimientos y el modelado de la etapa de análisis y determina la mejor manera de satisfacerlos, logrando niveles de servicios acordados, dados el ambiente técnico y las decisiones previas en los niveles requeridos de automatización.

A partir del diseño conceptual se pasa al diseño final que será utilizado para la implantación, por ejemplo en esta etapa, el modelo entidad/relación será transformado en un diseño de base de datos, y en especificaciones de almacenamiento; el modelo funcional en módulos y manuales de procedimientos.

El diseño final del sistema integra tres diseños, el de la base de datos, el de la aplicación y el de la red, además se elaboran los planes de prueba, de transición y se realizan los diseños de los sistemas de auditoria, al igual que los de control, de respaldos y de recuperación.

Los resultados de esta etapa lo constituyen, la arquitectura del sistema, el diseño de la base de datos, la especificación de los programas, la especificación de los manuales de procedimientos. En la figura 4.26 se muestra esta etapa [Barker, 1992].

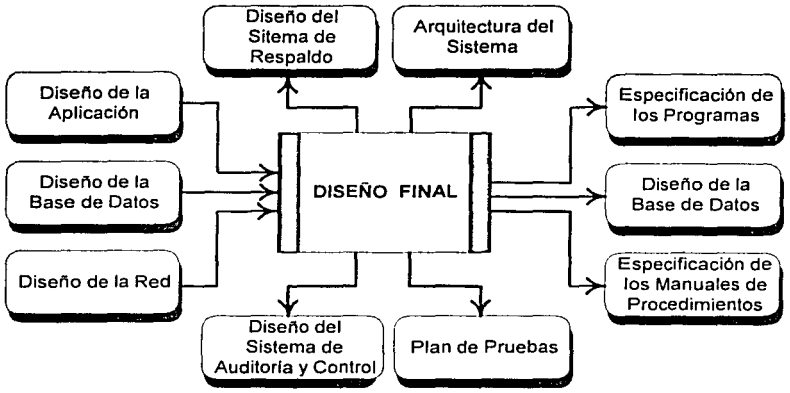


Figura 4.26 Etapa de Diseño

4.7.1.4 CONSTRUCCIÓN

A partir del diseño final generado en la etapa anterior, se construirán , codificarán y probarán los nuevos programas, usando herramientas apropiadas. Esta etapa involucra planeación, diseño de la estructura del sistema, codificación de abajo a arriba (prueba tanto de unidades, como de enlaces) y pruebas de arriba a abajo (prueba del sistema). Un enfoque disciplinado en la realización del trabajo y en el control de versiones del sistema, además de las pruebas, que están comprendidas en el diseño de la base de datos, la arquitectura del sistema y la especificación de los programas. Los resultados de esta etapa son los programas probados y la base de datos afinada [Barker, 1992].

Los resultados de esta etapa lo constituyen los programas probados y las bases de datos afinadas, en la siguiente figura se muestra el proceso de esta etapa.

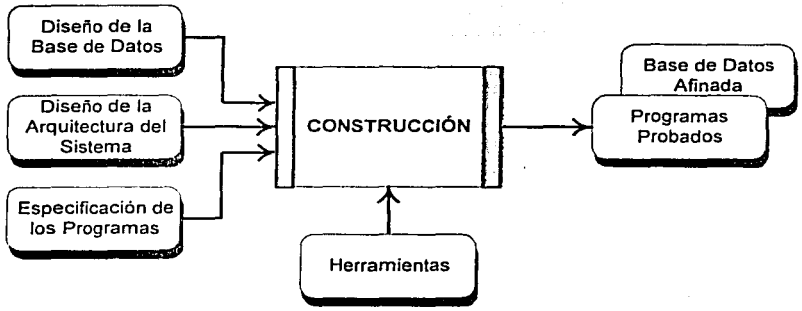


Figura 4.27 Etapa de Construcción

4.7.1.5 DOCUMENTACIÓN

Uno de los productos fundamentales para uso y el mantenimiento efectivos y eficientes de los sistemas programados, son los manuales. Esta metodología incluye una etapa dedicada a esta actividad tan importante haciendo hincapié para que en su elaboración se consideren el estilo de trabajo y las necesidades propias de los usuarios que utilizarán y mantendrán el sistema. Esta etapa se realiza al mismo tiempo que la de construcción.

Los manuales, resultados de esta etapa, se elaboran a partir de las especificaciones de diseño, de los programas realizados, así como del análisis del estilo de trabajo, además del nivel de competencia de los usuarios y operadores de los sistemas, dando como resultado tanto el manual de usuario, como el manual técnico [Barker, 1992]. En la figura 4.28 se muestra el proceso de esta etapa.

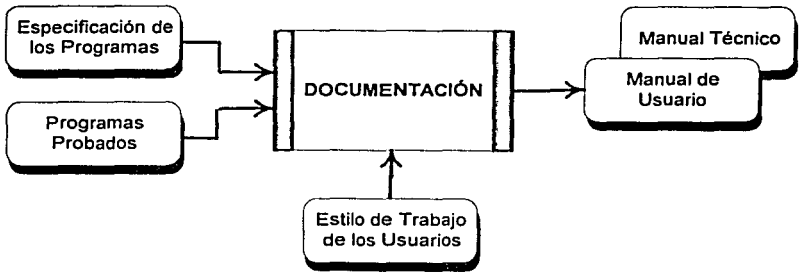


Figura 4.28 Etapa de Documentación

4.7.1.6 TRANSICIÓN

La implantación de sistemas no necesariamente implica la sustitución total de los antiguos subsistemas o de sus bases de datos correspondientes. En ciertos casos, por razones operativas y/o económicas, los nuevos sistemas integran algunos de los antiguos; pero como quiera que sea, la introducción ya sea de un sistema completamente nuevo o un sistema que integra ya existentes implica un nuevo tipo de uso y de operación que deberá ser tanto asimilado, como aprendido por los usuarios, así como por los operadores. Por esta razón, el desarrollo de un sistema no se termina con su programación; antes de su liberación para su uso, se debe prever un período de transición que deberá incluir la alimentación de la nuevas bases de datos, la capacitación de los usuarios y el desarrollo de pruebas.

En esta metodología la transición conforma una de sus etapas y en ella se realizan todas las tareas necesarias para la implementación, proporcionando un periodo inicial de soporte al sistema. La transición debe llevarse a cabo con una interrupción mínima de la organización, dejando a los usuarios confiados y listos para explotar el nuevo sistema.

El resultado final de esta etapa es un reporte que muestre que las pruebas fueron satisfactorias, en la figura 4.29 se muestra el proceso de esta etapa.

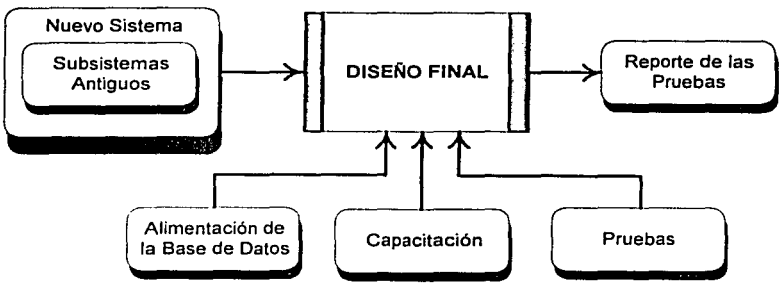


Figura 4.29 Etapa de Transición

4.7.1.7 PRODUCCIÓN

Finalmente, en la etapa de producción se asegura que el sistema funcione correctamente en la mayoría de los casos, con intervención mínima de los administradores del sistema. Para esto se realizan nuevas pruebas, se reevalúan los resultados, se hacen refinamientos del sistema. Los cambios necesarios deberán ser introducidos sin afectar a los usuarios, y deberá conseguirse la máxima confianza de los usuarios. El resultado de esta etapa es un sistema listo para su operación, como se muestra en la siguiente figura [Barker, 1992].

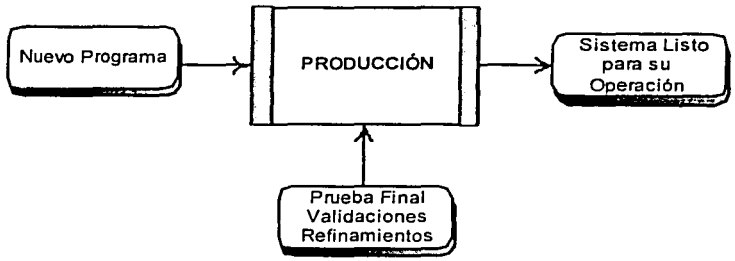


Figura 4.30 Etapa de Producción

4.7.2 BLOQUES BÁSICOS DE CASE

La ingeniería del software asistida por computadora puede ser tan sencilla como una única herramienta que preste su apoyo para una única actividad de ingeniería del software, o bien, puede ser tan compleja como todo un entorno que abarque herramientas, una base de datos, personas, hardware, una red, sistemas operativos, estándares, y otros mil componentes más. Los bloque de construcción de CASE se ilustran en la figura 4.31

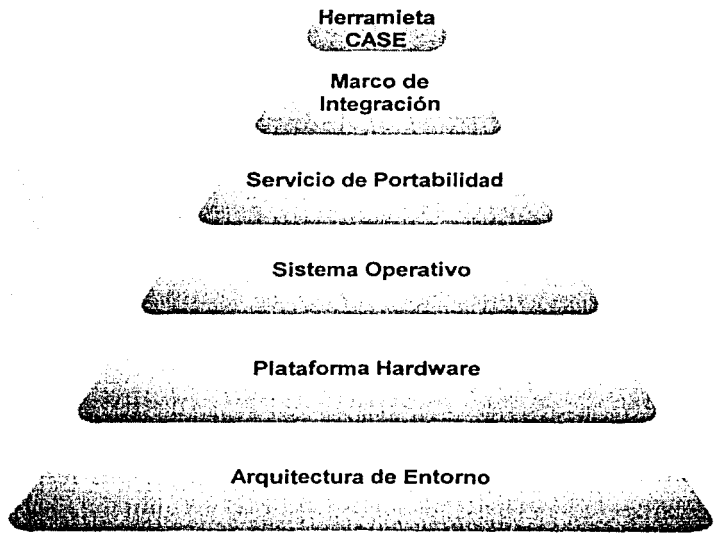


Figura 4.31 Bloque de construcción CASE

Es interesante tener en cuenta que el fundamentos de los entornos CASE efectivos tiene relativamente poco que ver con las herramientas de ingeniería del software en sí. Los sistema que tienen éxito para la ingeniería del software se construyen basándose en una arquitectura de entorno que abarca un hardware adecuado y un software de sistema adecuado. Además, la arquitectura del sistema debe considerar los patrones de trabajo humano que se aplicarán durante el proceso de ingeniería del software.

Las arquitecturas que constan de una plataforma hardware y de un apoyo de sistema operativo (incluyendo el software de red y de gestión de la base de datos), constituye los fundamentos de CASE. Existe un conjunto de servicios de portabilidad que proporciona un puente entre las herramientas CASE a su marco de referencia de integración y la arquitectura del entorno. El marco de referencia de integración es una colección de programas especializados que capacitan a las herramientas CASE individuales para comunicarse entre sí, para crear una base de datos del proyecto, y para mostrar el mismo aspecto al usuario final (el ingeniero del software). Los servicios de portabilidad permiten que las herramientas CASE y su marco de referencia de integración migren entre distintas plataformas del hardware y sistemas operativos sin un mantenimiento adaptativo que resulte significativo [Pressman, 1997].

Los niveles relativos de integración Se muestran en la figura 4.32. En el extremo inferior del espectro de integración se encuentra la herramienta individual (solución puntual). Cuando las herramientas individuales proporcionan capacidades adecuadas para el intercambio de datos (tal como hace la mayoría), el nivel de integración mejora ligeramente. En algunos casos, los constructores de herramientas CASE complementarias trabajan a la vez para formar un puente entre herramientas. Mediante el uso de este enfoque la asociación entre herramientas puede producir unos resultados finales que sería difícil crear empleando cada una de las herramientas por separado. La integración de fuente única se produce cuando un único vendedor de herramientas CASE integra un cierto número de herramientas distintas y las vende en forma de paquete.

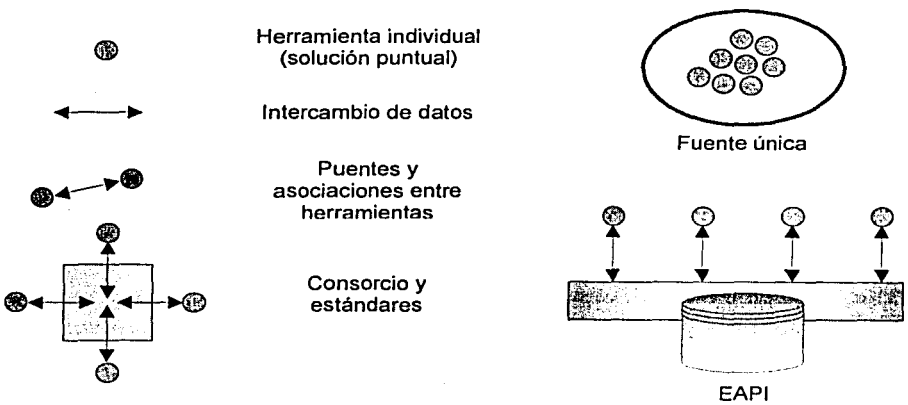


Figura 4.32 Opciones Integradas

En el extremo superior del espectro de integración se encuentra el entorno de apoyo de proyectos integrado (EAPI). Se crean estándares para cada uno de los bloques de construcción descritos anteriormente. Los fabricantes de herramientas CASE utilizan estos estándares EAPI para construir herramientas que sean compatibles con el EAPI, y que por tanto sean compatibles entre sí. [Pressman, 1997].

4.7.3 COMPONENTES HARDWARE DE UN SISTEMA CASE BÁSICO

Podemos dividir un sistema CASE en los siguientes componentes básicos:

- "Front-End"
- Depósito Central
- "Back-End"

El componente "front-end" corresponde a las fases primarias del ciclo de vida del software, el análisis y el diseño. El "front-end" puede corresponder a la parte de la computadora personal o estación de trabajo de la plataforma de hardware. Las herramientas CASE "front-end" proporcionan funciones para soportar las actividades de análisis, diseño, diagramación, prototipos, así como de la comprobación de especificaciones. Estas actividades se realizan mucho mejor en una máquina personal con "ratón", tiempo de respuesta alto y mapas de bits gráficos de alta resolución. [Pressman, 1997].

El componente "back-end" corresponde a las últimas fases del ciclo de vida del software; es decir, la implantación y el mantenimiento del programa. El "back-end" también corresponde a la porción de la computadora principal en la plataforma de hardware CASE. Las herramientas del "back-end" automatizan el código, las comprobaciones, la generación de las bases de datos, la normalización de los datos, además del análisis del impacto en el sistema existente. Estas tareas requieren la potencia y capacidad de almacenamiento de una máquina principal o de un servidor.

El depósito de información es el enlace entre los componentes "front-end" y "back-end" de un sistema CASE. Es el vehículo de comunicación, por el cual toda la información del sistema, reunida durante el ciclo de vida del software se gestiona y se comparte. Es también el vehículo en el cual el trabajo del sistema de los diferentes equipos pueden combinarse, para analizar y consolidar una representación del sistema consistente, en progreso y completa. El depósito es el integrador básico del ciclo de vida que permite a las herramientas utilizadas en una fase del ciclo de vida, pasar los datos a la fase siguiente.

Lógicamente, el depósito CASE está dividido en librerías de proyectos y en modelos del sistema. Físicamente, el depósito CASE está dividido en niveles que corresponden con las plataformas de hardware del sistema CASE. A nivel de la estación de trabajo, existe un depósito local para soportar el desarrollo individual. A nivel de computador anfitrión, existe un depósito principal para mantener toda la información corporativa. A nivel de departamento o de proyecto hay un depósito intermedio para mantener toda la información del proyecto. [Pressman, 1997].

Debe existir un enlace entre los depósitos físicos para que los datos puedan cargarse y descargarse entre los niveles. Por ejemplo, el contenido del depósito anfitrión puede descargarse total o parcialmente en un depósito de proyecto o de estación de trabajo; así mismo, el contenido del depósito a nivel de estación de trabajo puede cargarse y combinarse con los datos del depósito a nivel superior. Los enlaces los proporcionan las utilidades de comunicaciones como Ethernet, TCP/IP, NFS, Domain, DECnet, SNA y el protocolo 3270.

Sin embargo, el aspecto real del transporte de información del sistema entre los entornos de hardware CASE conlleva no solamente la posibilidad de transferir datos, sino también la facilidad de seleccionar y formatear los datos necesarios. Así pues, la conectividad entre los sistemas hardware debe incluir:

- Selección de los datos a descargar.
- Carga y consolidación de los datos.
- Necesidades de reconstrucción del formato de los datos.
- Control y seguridad de los datos.
- Velocidad.
- Eficiencia.
- Acceso multiusuario a los archivos.
- Copias de seguridad (backup).

4.7.4 HERRAMIENTAS CASE

Existe un cierto número de riesgos que son inherentes siempre que se intenta efectuar una categorización de las herramientas CASE. Existe una sutil implicación consistente en que para crear un entorno CASE efectivo uno debe de implementar todas las categorías de herramientas. Se puede crear una confusión o un antagonismo, al ubicar una herramienta especifica dentro de una categoría, cuando otras personas pueden creer que pertenece a otra categoría.

Las herramientas CASE se pueden clasificar por su función, por su papel como instrumentos para administradores o personal técnico, por su utilización en los distintos pasos del proceso de ingeniería del software, por la arquitectura de entorno (hardware y software) que les presta su apoyo, o incluso por su origen o su coste. La clasificación que se presenta en la tabla 4.4, utiliza como criterio principal la función [Pressman, 1997].

HERRAMIENTAS CASE

TIPO DE HERRAMIENTA	*DESCRIPCIÓN*
De Generación de Prototipos	Los generadores de pantallas permiten al ingeniero del software definir rápidamente la disposición de la pantalla para aplicaciones interactivas. Otras herramientas de prototipos CASE más sofisticadas permiten la creación de un diseño de datos, acoplado con las disposiciones de la pantalla y de los informes simultáneamente. Muchas herramientas de análisis y diseño proporcionan extensiones que ofrecen alguna opción de generación de prototipos. Las herramientas PRO/SIM generan un esqueleto de código fuente en ADA y C para las aplicaciones de ingeniería (en tiempo real).
De Programación	La categoría de herramientas de programación abarca los compiladores, editores y depuradores que están disponibles para prestar su apoyo en la mayoría de los lenguajes de programación convencionales. Además, los entornos de programación orientados a objetos (OO), los lenguajes de cuarta generación, los entornos de programación gráfica, los generadores de aplicaciones, y los lenguajes de consulta de bases de datos también están en esta.
De Análisis y Diseño	Estas herramientas capacitan al ingeniero del software para crear modelos del sistema que haya que construir. Los modelos contienen una representación de los datos, de la función y del comportamiento (en el nivel de análisis), así como caracterizaciones del diseño de datos, arquitectura, procedimientos e interfaz.
PRO/SIM	Las herramientas PRO/SIM (de prototipos y simulación) proporcionan al ingeniero del software la capacidad de predecir el comportamiento de un sistema en tiempo real antes de llegar a construirlo. Además, capacitan al ingeniero del software para desarrollar simulaciones del sistema de tiempo real que permitirán al cliente obtener ideas acerca de su funcionamiento, comportamiento. La respuesta antes de la verdadera implementación.
De la Ingeniería de la Información	Estas herramientas proporcionan un "metamodelo" del cual se derivan sistemas de información específicos, modelan la información de negocios cuando ésta se transfiere entre distintas entidades organizativas en el seno de una compañía.
De Seguimiento de Requisitos	El objetivo de estas herramientas es proporcionar un enfoque sistemático para el aislamiento de requisitos, comenzando por la solicitud del cliente de una propuesta (RFP) o especificación. Combinan una evaluación de textos por interacción humana, con un sistema de gestión de bases de datos que almacena y clasifica todos y cada uno de los requisitos del sistema que se analizan a partir de la especificación original .
De Documentación	Las herramientas de producción de documentos y de autoedición prestan su apoyo a casi todos los aspectos de la ingeniería del software, y representan una importante oportunidad de aprovechamiento para todos los desarrolladores de software.
De Manipulación de Bases de Datos	Sirve como fundamento para establecer una base de datos CASE (depósito), que también se denominará base de datos del proyecto. Dado el énfasis acerca de los objetos de configuración, las herramientas de gestión de bases de datos para CASE puede evolucionar a partir de los sistemas de gestión de bases relacionales (SGBDR). Para transformarse en sistemas de gestión de bases de datos orientadas a objetos (SGBDOO).
De Desarrollo y Diseño de Interfaz	Son en realidad un conjunto de primitivas de componente de programas tales como menús, botones, estructuras de ventanas, iconos, mecanismos de desplazamiento, controladores de dispositivos, etc. Sin embargo, estos conjuntos de herramientas se están viendo sustituidos por herramientas de generación de prototipos de interfaz, que permiten una rápida creación en pantalla de sofisticadas interfaces de usuario, las cuales se ajustan al estándar de interfaz adoptada para el software.

Tabla 4.4 Herramientas CASE

4.7.5 ENTORNOS CASE INTEGRADOS

Aún cuando se pueden obtener beneficios a partir de herramientas individuales CASE, que abarquen actividades de ingeniería del software por separado, la verdadera potencia de CASE solamente se puede lograr mediante la integración. Los beneficios del CASE integrado (I-CASE) incluyen [Pressman, 1997].

1. Una transferencia suave de información (modelos, programas, documentos, datos) entre una herramienta y otra, entre un paso de ingeniería y el siguiente.

2. Reducción del esfuerzo necesario para efectuar actividades globales tales como la administración de configuración de software, el control de calidad y la producción de documentos.

3. Aumento de control del proyecto, que se logra mediante una mejor planificación, monitorización y comunicación.

4. Mejor coordinación entre los miembros del personal que estén trabajando en grandes proyectos de software.

Ahora bien, I-CASE también presenta desafíos significativos. En cada acción exige unas representaciones consistentes de la información de la ingeniería del software, unas interfaces estandarizadas entre herramientas, un mecanismo homogéneo para la comunicación entre el ingeniero de software y todas sus herramientas, y un enfoque efectivo que capacite a I-CASE para desplazarse a lo largo de distintas plataformas de hardware y distintos sistemas operativos.

Mientras las soluciones de los problemas implícitos en estos desafíos se han propuesto ya, los entornos I-CASE generales que empiezan a emerger en la actualidad.

El término integración implica tanto la combinación como el cierre. La I-CASE combina toda una gama de herramientas diferentes y de informaciones distintas de tal modo que hace posible el cierre de la comunicación entre herramientas, entre personas, y entre procesos de software.

Se integran las herramientas de tal modo que la información de ingeniería del software esté disponible para todas las herramientas que se necesiten. La utilización se integra de tal modo que se proporcione un aspecto común para todas las herramientas; y se integra una filosofía de desarrollo, implicando un enfoque de ingeniería del software estandarizado que aplique prácticas modernas y métodos ya probados.

Para definir integración en el contexto del proceso del software, es necesario establecer un conjunto de requisitos para I-CASE. Un entorno CASE integrado debería de:

- Proporcionar un mecanismo para compartir la información de ingeniería del software entre todas las herramientas que estén contenidas en el entorno.

- Hacer posible que un cambio de un elemento de información se siga hasta los demás elementos de información relacionados.
- Proporcionar un control de versiones y una gestión de configuración general para toda la información de la ingeniería del software.
- Permitir un acceso directo y no secuencial a cualquiera de las herramientas contenidas en el entorno.
- Establecer un apoyo automatizado para un contexto de procedimientos para el trabajo de la ingeniería del software, que integra las herramientas y los datos en una estructura de desglose de trabajo estandarizada.
- Capacitar a los usuarios de cada una de las herramientas para experimentar una utilización consistente en la interfaz hombre-máquina.
- Permitir la comunicación entre ingenieros del software.

Para alcanzar estos objetivos, cada unos de los bloques de construcción de una arquitectura CASE (figura 4.32) debe encajar con los demás sin costuras. Según se muestra en la figura 4.33, los bloques de construcción fundamentales -arquitectura del entorno, plataforma hardware y sistema operativo- deben de unirse a través de un conjunto de servicios de portabilidad a un marco de referencia de integración que alcance los objetivos indicados anteriormente. [Pressman, 1997].

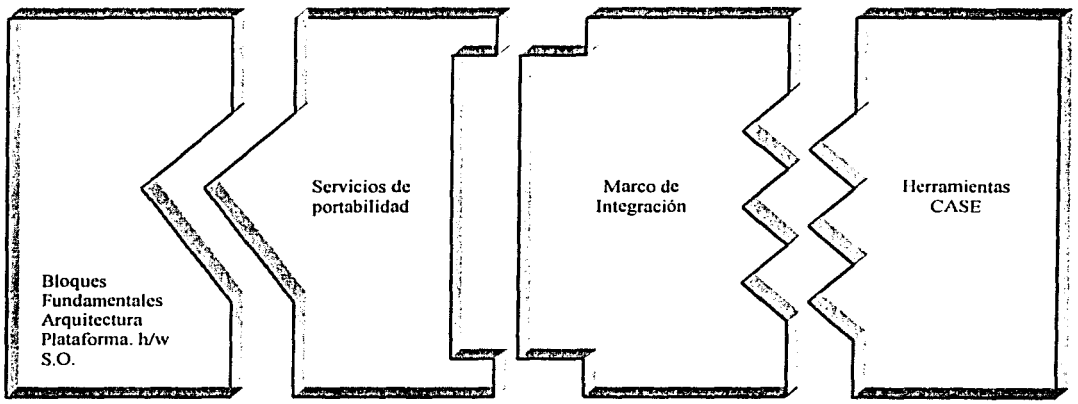


Figura 4.33 Elementos de I-CASE

4.7.6 LA ARQUITECTURA DE INTEGRACIÓN

Un equipo de ingeniería del software utiliza herramientas CASE, los métodos correspondientes, y un marco de referencia de proceso con objeto de crear un conjunto de informaciones de ingeniería del software. El marco de referencia de integración facilita la transferencia de información desde y hacia ese conjunto de informaciones.

Para lograr esto, deben existir los siguientes componentes de arquitectura: una base de datos para almacenar la información, un sistema de administración de objetos que gestione los cambios efectuados en la información, un mecanismo de control de herramientas que coordine la utilización de herramientas CASE, y una interfaz de usuario que proporcione una ruta consistente entre las acciones efectuadas por el usuario y las herramientas contenidas en el entorno. La mayoría de los modelos del marco de referencia de integración representan a estos componentes como si fueran capas.

La capa de interfaz de usuario incorpora un conjunto de herramientas de interfaz estandarizado, con un protocolo de presentación común. El grupo de herramientas de interfaz contiene software para la gestión de la interfaz hombre-máquina, y una biblioteca de objetos de visualización. Ambos proporcionan un mecanismo consistente para la comunicación entre la interfaz y las herramientas CASE individuales. El protocolo de presentación es el conjunto de líneas generales que proporciona a todas las herramientas CASE un mismo aspecto. Las convenciones de distribución de pantalla, los nombres de menú y la organización, los iconos, los nombres de los objetos, la utilización del teclado y el ratón, y el mecanismo para acceder a las herramientas se definen todos ellos como parte del protocolo de presentación. [Pressman, 1997].

La capa de herramientas contiene un conjunto de servicios de gestión de herramientas con las herramientas CASE en sí. Los servicios de gestión de herramientas (TMS) controlan el comportamiento de las herramientas dentro del entorno. Si se emplea multitarea durante la ejecución de una o más herramientas, TMS efectúa la sincronización y comunicación multitarea, coordina el flujo de información desde el depósito y sistema de gestión de objetos a las herramientas, realiza las funciones de seguridad y auditoría y recoge métricas acerca de la utilización de herramientas.

La capa de gestión de objetos (OML) lleva a cabo las funciones de gestión de configuración. En esencia, el software de esta capa de la arquitectura de marco de referencia proporciona el mecanismo para la integración de herramientas. Cada herramienta CASE se ensambla en la capa de gestión de objetos. Funcionando en conjunción con el depósito CASE, la OML proporciona los servicios de integración -un conjunto de módulos estándar que acoplan las herramientas con el depósito-.

Además, la OML proporciona los servicios de gestión de configuración haciendo posible la identificación de todos los objetos de configuración, llevando a cabo el control de versiones, y proporcionando apoyo para el control de cambios, las auditorías y la contabilidad de estados [Pressman, 1997].

4.7.7 DESARROLLO DE BASES DE DATOS CON HERRAMIENTAS CASE

El desarrollo de una base de datos se puede dividir, de forma general, en las siguientes fases:

Modelado conceptual. Cuyo objetivo es obtener una buena representación de los recursos de información de la empresa, con independencia de usuarios o aplicaciones en particular, y sin tener en cuenta consideraciones sobre eficiencia de la computadora.

Diseño lógico. Cuyo objetivo es transformar el esquema conceptual obtenido en la fase anterior, adaptándolo al modelo de datos que soporta el SGBD que se va a utilizar (Jerárquico, Codasyl, Relacional, Orientado al Objeto, etc.).

Diseño físico. Cuyo objetivo es instrumentar el esquema lógico de la forma más eficiente posible, teniendo en cuenta no solo las características propias del SGBD, sino también del sistema operativo (SO) y del soporte físico (hardware) empleados [Piattini, 1995].

Una herramienta CASE para el desarrollo de bases de datos, deberá soportar un conjunto de técnicas para cada una de estas fases que faciliten el diseño de los distintos esquemas (conceptual, lógico y físico), así como, algoritmos que permitan pasar de una fase a la siguiente, o refinar los esquemas de una fase determinada. Además, deberá generar toda la documentación asociada a estos esquemas, de acuerdo con las normas que imponga la metodología seguida.

En general, cabe destacar la necesidad de una integración cada vez mayor de sistemas expertos en las herramientas de desarrollo de bases de datos que almacenen el conocimiento sobre las técnicas y las metodologías a emplear, asistiendo así a los diseñadores.

4.7.7.1 DISEÑO CONCEPTUAL

En general, las herramientas CASE ofrecen un buen soporte gráfico para esquemas conceptuales de tamaño medio que facilita enormemente la labor del diseñador. Por lo que para bases de datos muy grandes, es más conveniente ir a soluciones más cooperativas, en las que se utilizan, por un lado, estaciones de trabajo o computadoras personales para soportar la visualización de los esquemas y realizar ciertos procesos, y por otro, una máquina de mas capacidad (incluso un "*mainframe*") para otros tipos de procesos.

Recientemente, algunos expertos sugieren la conveniencia de que las herramientas CASE ofrezcan mayor versatilidad en la representación de las estructuras de datos. Por lo que sería deseable que se soportaran [Pressman, 1997].

- Esquemas a los tres niveles de la arquitectura ANSI (externo, conceptual e interno).
- Varios diagramas sobre un mismo modelo.
- La definición de entidades e interrelaciones virtuales definidas a partir de otras entidades e interrelaciones, pero con un mayor nivel de abstracción.
- La visualización de subtipos sin tener que representar los supertipos.

Además de los gráficos, las herramientas CASE actuales ofrecen la posibilidad de verificar ciertas características de los esquemas conceptuales, por ejemplo que todas las entidades tengan un identificador, o que no haya objetos no asociados a ningún otro, e incluso, hay algunas que leen los esquemas en lenguaje natural, facilitando, así, su validación por parte de los usuarios finales. Un paso más en esta dirección consiste en que la herramienta pueda disponer de una serie de reglas en las que se exprese la metodología empleada, guiando al diseñador durante todo el ciclo de vida.

Un aspecto en el que las herramientas CASE dejan todavía que desear, aunque se esta mejorando considerablemente, es el escaso soporte que ofrecen a las reglas de identidad; es decir, poder comprobar o llevar a cabo un análisis de restricciones de integridad a nivel conceptual como lógico.

Para abordar diseños conceptuales de tamaño considerable, varias metodologías proponen seguir el enfoque conocido como integración de vistas, que consiste en elaborar esquemas conceptuales parciales (más manejables) y a partir de estos ir obteniendo esquemas conceptuales mayores que los engloben. Desafortunadamente, existen muy pocas herramientas CASE que soporten de manera satisfactoria esta técnica. La mayoría se limitan a realizar la detección de sinónimos y homónimos entre las diferentes vistas, pero no asisten al diseñador durante el proceso de integración.

Hay que destacar la posibilidad que ofrecen bastantes herramientas de construir esquemas conceptuales a partir de especificaciones en lenguajes como COBOL, o esquemas de bases de datos en red o relacionales, mediante un proceso de reingeniería, lo cual resulta bastante más fácil en el caso de los datos que para los procesos. [Pressman, 1997].

4.7.7.2 DISEÑO LÓGICO

En cuanto a diseño lógico las herramientas CASE suelen soportar, sobre todo el diseño relacional, debido al gran número de productos relacionales presentes en el mercado (ORACLE, INFORMIX, DB2, INGRES, INTERBASE, SYBASE, PROGRESS, etc.), aunque existen algunas que soportan bases de datos Codasyl o Jerárquicas, y ya empiezan a aparecer herramientas que permiten diseñar bases de datos Orientadas al Objeto.

Los productos actuales facilitan la obtención de esquemas lógicos, conjunto de tablas y restricciones, a partir del esquema conceptual, aplicando pocas reglas sencillas por ejemplo, se construye una tabla por cada entidad interrelación n a muchos (N:M), y se propagan las claves primarias para instrumentar las interrelaciones uno a n (1:N).

Existen algunas herramientas que permiten optimizar el paso a relacional teniendo en cuenta ciertas características de los procesos, como los atributos accedidos en cada proceso, prioridades, etc. Otra capacidad que suele atribuirse a las herramientas CASE es la de normalizar las tablas. Pocas herramientas incluyen algoritmos avanzados de normalización, limitándose a preguntar al usuario sobre los grupos repetitivos, las dependencias funcionales incompletas o las transitivas, y procediendo a descomponer las tablas en otras más normalizadas.

En general, suelen garantizar un esquema normalizado hasta tercera forma normal o forma normal de Boyce y Codd, aplicando las reglas de transformación a un esquema conceptual que esté correctamente diseñado.

En la fase de diseño lógico, es también muy importante la integración que exista entre la herramienta CASE y el SGBD, que suele ser mejor en el caso de que ambos productos sean del mismo fabricante, ya que se puede generar más fácilmente código en el lenguaje de cuarta generación (L4G) del SGBD a partir de los diagramas de flujo de datos, los diagramas de estructuras o la descomposición funcional.

4.7.7.3 DISEÑO FÍSICO

Desafortunadamente, muy pocas herramientas CASE permiten llevar a cabo un auténtico diseño físico, debido a las siguientes causas:

- La inexistencia de un modelo físico general (análogo al modelo relacional para el nivel lógico).
- Los escasos elementos de diseño físico presentes en la mayoría de los productos relaciónales.
- La poca integración entre los esquemas de datos y los procesos en algunas herramientas.

Existe un gran número de opciones de diseño físico que a menudo se encuentran interrelacionadas, complicando la labor del diseñador. Entre estas destacamos:

- Representación de los elementos, punteros, particionamiento y compresión.
- Formato de los ficheros y espacios libres en las páginas de disco.
- Selección de los índices.
- Replicación e información derivada.
- Agrupamiento.
- Ordenación.

Las herramientas CASE existentes en la actualidad se suelen limitar a generar algunos índices secundarios (por ejemplo, sobre las columnas que constituyen las claves ajenas), y calcular de manera aproximada el tamaño de la base de datos.

El diseño físico se podría soportar de una manera mucho más rigurosa, permitiendo al diseñador determinar una función de costo que considerará aspectos tales como el número de accesos a disco, utilización de CPU, costo de almacenamiento, etc. y que mediante la información sobre los procesos y conociendo los atributos accedidos por los mismos, se presentan diversas opciones de diseño físico para cada uno de los elementos anteriormente enumerados.

CAPÍTULO 5

APLICACIÓN DE LA METODOLOGÍA CASE

5.1 INTRODUCCIÓN

El presente capítulo describe la aplicación de CASE para la implantación de una base de datos de imágenes satelitales. Cada una de las etapas que conforman la metodología se describirá de manera específica, con el objetivo de plasmar cada uno de los pasos seguidos de acuerdo a la metodología y los resultados obtenidos de cada uno de ellos.

5.2 ETAPA DE ESTRATEGIA

Partiendo de la documentación existente, de entrevistas con los responsables del proyecto y personal que labora en el Laboratorio de Oceanografía Física (LOF); el cual forma parte del Instituto de Ciencias del Mar y Limnología (ICML) de la UNAM, se obtuvo la siguiente información:

5.2.1 ANTECEDENTES

La actividad necesaria e importante que se desarrolla en el contexto del proyecto, es el archivo de la información estadística de las imágenes de satélite de la TSM. Esta se genera con la acumulación sistemática de los datos que resultan de la observación y medición radiométrica de la superficie del océano, en particular de los mares mexicanos, desde dispositivos *ad hoc* montados en plataformas satelitales.

De manera sistemática se acumula, integra y maneja un acervo de datos de la TSM de una región oceánica que incluye a toda la zona económica exclusiva de nuestro país. Esta información es valiosa e indispensable para el conocimiento de la climatología de los mares mexicanos.

A partir de la segunda semana de enero de 1996 se inicio, en el ICML, el registro continuo de imágenes de satélite (NOAA–12 y NOAA-14) de la TSM en una región cuya extensión geográfica incluye los mares de México y a las aguas internacionales adyacentes.

El registro sistemático de estos datos han sido parte sustancial del proyecto de investigación oceanográfica. Una actividad importante que se desarrolla en el contexto del proyecto OPOS es el almacenamiento de la información estadística de las imágenes de satélite de la TSM.

5.2.2 OBJETIVO

El propósito del proyecto "Acervo de Datos Radiométricos Satelitales para el estudio de los procesos oceánicos y de la climatología de la temperatura de la superficie de los mares de México y aguas internacionales adyacentes" es construir un banco de información de imágenes satelitales de la TSM de los mares de México principalmente, bien organizado y de fácil acceso.
Las características de este banco de información, una vez instalado, se ajusta a las necesidades de diversos usuarios: meteorólogos, oceanógrafos, climatólogos y otros especialistas investigadores en ciencias geofísicas y ambientales. Se requiere que lo puedan consultar personas involucradas en relación con este ámbito en general por medio de servicios en línea.

Con este acervo acumulativo de información de la TSM, en los próximos años se podrían hacer análisis locales y regionales para identificar y estudiar procesos climáticos tan relevantes como, por ejemplo, los de el El Niño y La Niña, así como la influencia respectiva en el régimen de lluvias sobre México, en las pesquerías comerciales residentes en el Océano Pacifico Oriental (incluyendo el Golfo de California) en la frecuencia e intensidad de los huracanes que ocasionalmente invaden el territorio mexicano.

Para llevar a cabo el registro de imágenes se ha laborado con dos estaciones de trabajo (Silicon Graphics Indy y Sun Sparc 4), conectadas por la red de internet a la estación de trabajo esclava del sistema instalado en el Instituto de Geografía de la UNAM. Se cuenta también en el LOF con dispositivos de almacenamiento magnético (Unidad de Grabación de Cintas de 8mm) y con discos duros.

5.2.3 HERRAMIENTAS DE TRABAJO

Para llegar a la decisión de adquirir el software y hardware de desarrollo requerido, se realizaron diferentes cuestionamientos en el grupo de trabajo, con el fin de tomar la decisión adecuada. Entre estas consideraciones, están las siguientes:

- Presupuesto asignado para el desarrollo del proyecto.
- Soporte proporcionado por los diferentes proveedores de software y hardware.
- Experiencia en las diferentes plataformas de desarrollo.
- Calidad reconocida en el posible software y hardware a adquirir.

De las consideraciones anteriores, se llego a la conclusión de adquirir las siguientes herramientas de trabajo:

- Servidor: Sun Ultra 10 con Sistema Operativo Solaris 4.0
- Sistema Manejador de Bases de Datos (DBMS) Oracle 8i.
- Dos PC's con procesador pentiumIII a 550 Mhz y 128 Mbytes en RAM.

5.2.4 GRUPO DE TRABAJO

Además del equipo de trabajo para el correcto desarrollo del sistema se obtuvieron los servicios de dos becarios de licenciatura, necesarios para cumplir con los siguientes propósitos académicos y de servicio de este proyecto:

- Desarrollar y establecer un banco de información de la temperatura de la superficie del mar (TSM) de los mares mexicanos y aguas internacionales adyacentes.
- Disponer de la información contenida en las imágenes satelitales para que sirva de sustento a estudios de procesos oceánicos y de la climatología de la TSM relevantes al clima regional, de Mesoamérica.
- Alimentar el acervo de la TSM, con la información que se obtiene con los datos radiométricos de los AVHRR a bordo de los satélites NOAA-12, NOAA-14, NOAA-15 y NOAA-16.
- Proporcionar el acceso al banco de información a través de la red Internet, de manera eficaz y amigable.

5.2.5 CONDICIONES ACTUALES

Para la captura de imágenes satelitales se realiza el siguiente proceso: se estable una sesión remota en ambiente UNIX, a la estación de trabajo en donde se encuentra instalado el sistema de adquisición y procesamiento de imágenes satelitales llamado TERASCAN, esta estación se encuentra localizada en el Instituto de Geografía de la UNAM. Una vez establecido el canal de comunicación de ejecutan una serie de comandos para la captura de imágenes sin procesar, conocidas también como "datos crudos".

Cada imagen sin procesar es almacenada en una estación de trabajo del LOF, de manera temporal en el directorio que corresponde al mes en curso. Esto tiene como propósito, el que al final del mes se realice un respaldo en cinta de todas las imágenes sin procesar capturadas durante el mes. A las cintas que contienen las imágenes sin procesar se les asigna una etiqueta en donde se indica el mes y año de las imágenes almacenadas.

Para tener un control de esta información se tienen dos cuadernos de registro, en uno de los cuales se anota cada dato crudo almacenado, agregando la información de la elevación del sol y la elevación del satélite. Además, si no se pudo almacenar la imagen sin procesar, se comenta el motivo, por ejemplo: falta de datos, ruido, error en los parámetros, etc. En el otro cuaderno se registra el mes que se ha respaldado.

De cada dato crudo se puede generar otra imagen que también es respaldada en cinta, para obtener una fotografía, la cual debe formar parte de un acervo fotográfico, dichas fotografías se toman de manera no periódica. Se requiere tener reportes del estado del acervo fotográfico, ya que es importante saber qué imágenes tienen su respectiva foto.

En el caso de las imágenes procesadas, se realiza una serie de tareas apara su obtención. Primero para calcular una imagen de TSM, se requiere un dato crudo. Para cada mes que se procesa es necesario disponer de los datos crudos de ese mes en el sistema TERASCAN, para ejecutar los comandos que permitan calcular la TSM. Por lo que se tienen que recuperar los datos de las cintas de respaldo, en el caso de trabajar con meses que no estén procesados. Una vez procesadas las imágenes del mes en cuestión se almacenan en cinta. Se cuenta con un cuaderno en donde se registran cada una de las imágenes procesadas, así como los parámetros empleados en el cálculo de la temperatura.

En cuanto a las imágenes promedio semanal y mensual, éstas se calculan al mismo tiempo en que se procesan los datos crudos. Se tiene un cuaderno en donde se registra la imagen promedio semanal, indicando cuáles imágenes procesadas son incluidas en el cálculo del promedio. En el mismo cuaderno se registran las imágenes promedio mensual y también se indica cuáles imágenes promedio semanal se analizaron para obtener el promedio mensual. Para obtener las imágenes recorte de región se tiene que realizar una serie de etapas, primero se crea mediante comandos del sistema TERASCAN un master (archivo que contiene los datos geográficos de la localización de la región), este master es empleado en la obtención de recortes.

Una vez concluida esta etapa, es necesario disponer de las imágenes procesadas de las cuales se obtendrán los recortes. Estas imágenes, pueden ser imágenes crudas, procesadas diarias, promedios semanal o mensual. Una vez obtenidas las imágenes recorte de región, son almacenadas en cinta o de manera temporal en el disco de alguna de las estaciones de trabajo del LOF. Actualmente no se cuenta con registro de estas imágenes, por lo que se hace necesario tener un control de las mismas, además de agregar la información del master empleado en su obtención.

5.3 ETAPA DE ANÁLISIS

En esta etapa se tomaron los elementos definidos durante la etapa de estrategia, los cuales ayudaron en la elaboración de diferentes modelos y diagramas que representan, tanto la forma de funcionamiento actual del sistema, así como las soluciones planteadas para el cumplimiento adecuado de los requerimientos detectados en la etapa anterior y el satisfacer las necesidades de información por parte del sistema.

El objetivo del sistema es brindar la información de la base de datos a través de la red. Toda persona que visite la página podrá consultar solamente las imágenes promedio mensual que correspondan a los dos meses más recientes. Para brindar más servicios se hace necesario definir usuarios, por lo que se tendrán dos tipos de usuarios en la web, quienes tendrán el acceso a todo el acervo de las imágenes satelitales (usuario frecuente, usuario especial). En cuanto a las necesidades a cubrir dentro del laboratorio y miembros del instituto, se definen dos tipos de usuarios. Los usuarios icml y lof, tendrán diferentes privilegios de acceso al acervo.

5.3.1 MODELO FUNCIONAL

A través del análisis de los requisitos mencionados se crearon modelos del sistema a construir, estos modelos se concentran en lo que debe hacer el sistema, no en como lo hace. Los modelos creados emplean una notación gráfica que muestra información, procesamiento, comportamiento del sistema y otras características con diferentes símbolos. Otras partes del modelo pueden ser texto puro.

El modelo funcional transforma la información, y para hacerlo, debe realizar al menos tres funciones genéricas: entrada, procesamiento y salida. Lo anterior se obtiene con el Modelo Funcional (figura 5.1), con el cual se representa el funcionamiento general del sistema a implantar como una única transformación de información. En el se tienen entidades externas, representadas por cuadros que originan una o mas entradas, que aparecen como flechas etiquetadas. La entrada conduce a la transformación que produce información de salida (también representada con flechas etiquetadas) dirigida hacia otra transformación, entidad externa o almacén de datos.

5.3.2 DIAGRAMA DE PROCESOS

El entendimiento pleno de las actividades y funciones que se realizan dentro del sistema a modelar, es un punto de partida fundamental para alcanzar un diseño claro y preciso del sistema en desarrollo.

Una forma de reflejar este entendimiento fue mediante la elaboración de diversos diagramas, entre los cuales se encuentra el diagrama de procesos que tiene como objetivo el modelar los procesos realizados en el sistema (ver anexo A).

Este diagrama representa una actividad necesaria para esquematizar una tarea, con un grado de detalle especifico. En donde las actividades se subdividen en diferentes niveles. El diagrama de procesos no proporciona información del desarrollo temporal o la sucesión de procesos; pero describe los datos necesarios para la creación de información en los diferentes procesos.

Inicialmente, se debe considerar a un proceso, como una sucesión de pasos y decisiones que se siguen para realizar una determinada actividad o tarea; también se puede definir a un proceso como un conjunto de actividades secuenciales, que realizan una transacción. con una serie de entradas (material, mano de obra, capital, información, etc.) a las salidas deseadas (bienes y/o servicios).

Los procesos identificados pueden ser de diferentes tipos, es decir, puede ser un proceso manual, el cual requiere la participación de al menos una persona, lo cual significa que aun cuando el proceso sea realizado en gran parte por el sistema, requerirá invariablemente de la interacción humana. También se tienen procesos automáticos, los cuales son realizados enteramente por un sistema de cómputo.

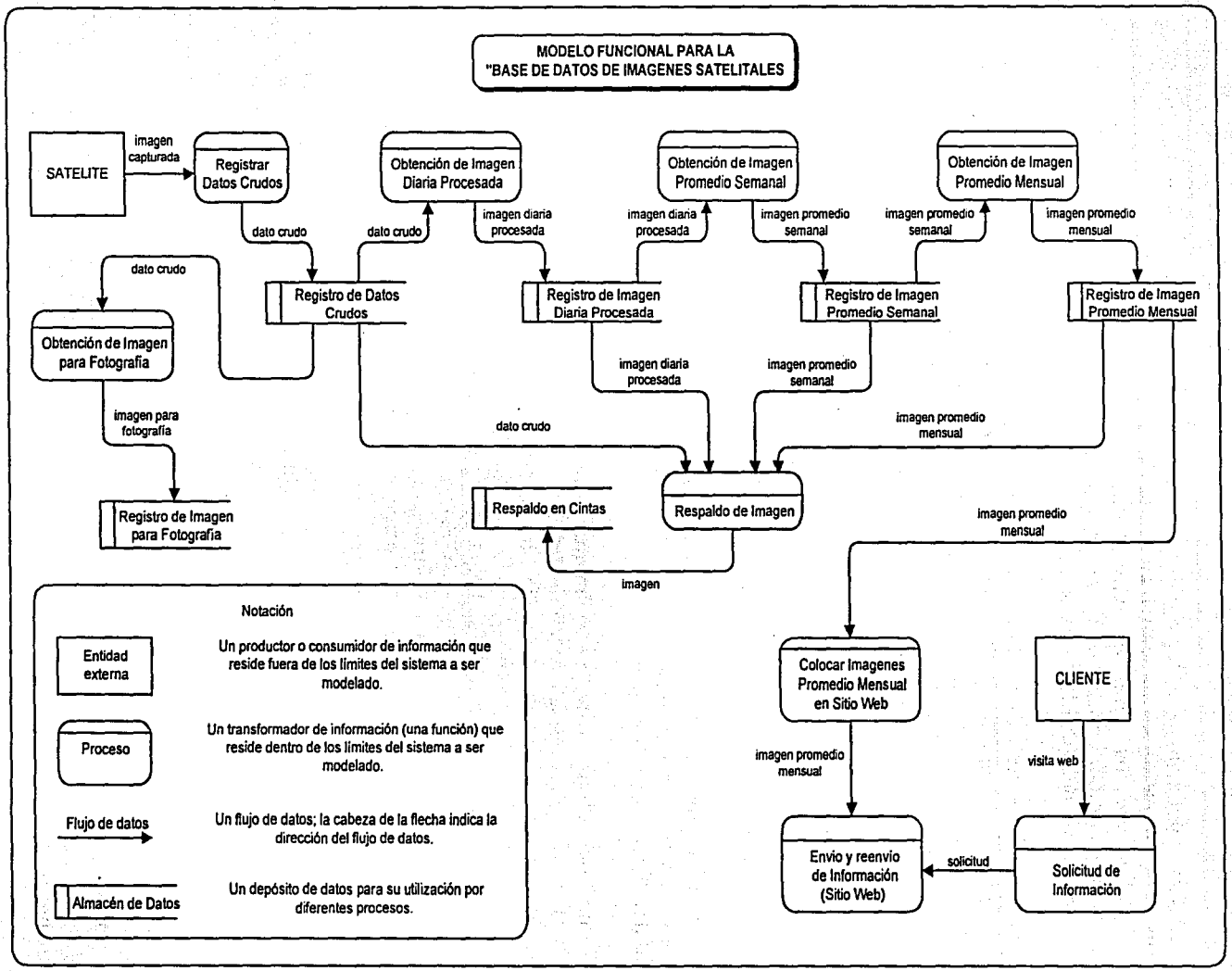


Figura 5.1 Modelo Funcional

Los procesos externos, son efectuados por una entidad externa y no se tiene control sobre ellos; y finalmente los procesos terminales, que indican la terminación de una serie de procesos intermedios y que su resultado no requiere ningún contacto con una entidad. Los diferentes procesos mencionados se apoyan para su funcionamiento de:

- Datos de entrada: Datos que necesita el proceso y se transforman en datos de salida.
- Datos de salida: Datos o informaciones creados por el proceso.

Dentro del diagrama, cada proceso fue representado por una pequeña imagen (icono), y las flechas representan la relación entre procesos. El diagrama de procesos para la Base de Datos de Imágenes Satelitales obtenido se compone de 16 unidades de procesos, de las cuales cada una esta formada por diversas tareas, las unidades de proceso son:

Imágenes Satelitales (Unidad de Proceso 1)

El objetivo de esta unidad, es el de obtener los datos crudos enviados por los satélites NOAA, al servidor ubicado en el Instituto de Geografía. Con una sesión remota hacia esa máquina, y ejecutando comandos, se descargan los datos crudos en máquinas del propio laboratorio.

Registro de imágenes: datos crudos (Unidad de Proceso 2)

Esta unidad representa la forma en la que se realiza el almacenamiento de datos crudos, en donde, se lleva a cabo el registro de las imágenes en cuadernos (bitácoras), posteriormente se almacenan en disco duro donde se comprimen y finalmente se respaldan en cinta.

Creación de imágenes para foto (Unidad de Proceso 3)

La unidad describe gráficamente las tareas en que se realizan en la obtención de las imágenes para fotografía, se eligen los mejores datos crudos que se tengan durante el día seleccionado para obtener su imagen para fotografía, posteriormente se almacenan con el mismo nombre del dato crudo pero con terminación ter, que la identificara. Enseguida se comprime el archivo y se respalda en cinta.

Toma de Fotografía (Unidad de proceso 4)

En esta unidad se representa como se obtiene una fotografía. Primero, se recupera la imagen que se selecciono como imagen para fotografía, a esta imagen se le toma una fotografía. Esta fotografía es registrada en una bitácora, posteriormente se clasifica la foto para que pueda ser agregada al acervo de fotografías (con el que cuenta el laboratorio).

Imágenes Diarias SST (Unidad de Proceso 5)

Aquí se muestra como se generan las imágenes diarias procesadas (SST). Inicialmente se recupera la imagen diaria sin procesar, a la cual se le desea procesar.

La recuperación se realiza de las maquinas del laboratorio (si es una fecha reciente) o de lo contrario se recupera de la cinta que la contenga. Teniendo la imagen diaria sin procesar, se le aplican una serie de procesos por medio del sistema TERASCAN, para generar la imagen diaria procesada, A esta imagen procesada se le asigna un nombre que lo identificará y relacionará con el día que corresponda. La imagen diaria procesada se registra en una bitácora; además de ser almacenada en disco, comprimida y finalmente respaldada en cinta.

Promedios Semanales (Unidad de Proceso 6)

Recuperando las imágenes diarias procesadas que conforman una semana completa, todas estas se analizan a través del sistema TERASCAN, para poder generar la imagen promedio semanal, que es registrada en la bitácora correspondiente, almacenada en disco, es comprimida y respaldada en cinta.

Obtención de Promedios Mensuales (Unidad de Proceso 7)

Primeramente se recuperan las imágenes promedio semanal que serán representativas del mes a promediar. Estas imágenes son procesadas en TERASCAN, que calcula la imagen promedio mensual. La imagen obtenida se registra en la bitácora correspondiente, después se almacena en disco duro, se comprime y se respalda en cinta.

Casos de Estudio (Unidad de Proceso 8)

En esta unidad se representa la forma en la que se realiza el estudio de una imagen determinada. Primero se obtiene un recorte del área de estudio que se basa en la petición de un usuario, sobre una determinada región. En su caso la imagen obtenida del recorte de región se envía al usuario que la solicito, o bien se tiene lista para analizarla.

Análisis de Imágenes de Estudio (Unidad de Proceso 9)

Se representa el análisis de imágenes de estudio. Esta actividad comienza a partir de una imagen recorte de región, a la cual se le aplicarán cálculos. El análisis de los resultados es en base a la información calculada de la imágenes de recorte de región.

Organización de Peticiones (Unidad de Proceso 10)

La unidad representa la recepción de las diferentes solicitudes de información, que realizarán los diversos usuarios. Cada solicitud se clasificará de acuerdo al tipo de información solicitada y del usuario en cuestión, para posteriormente definir la información a buscar.

Revisión de Consultas (Unidad de Proceso 11)

Se muestra como se realiza la búsqueda de la información requerida por las diversas solicitudes. El resultado de la búsqueda, previa revisión será enviada a los usuarios

Cumplimiento de Petición (Unidad de Proceso 12)

Esta unidad representa, como se proporcionará la información solicitada, se estimará el tiempo necesario para el cumplimiento de tal petición y una calendarización o programación de donde se dispondrá de la información en el sitio web, para que el usuario pueda acceder a ella.

Alta de Usuario LOF (Unidad de Proceso 13)

En esta unidad se muestra la forma en la que el usuario LOF es registrado. Estando registrado en el sistema, el usuario LOF manipulará la información que se le haya permitido.

Alta de Usuario ICML (Unidad de Proceso 14)

Esta unidad representa la manera en que el usuario ICML será dado de alta en el sistema, lo cual se realizara mediante el correcto llenado de una solicitud por parte del aspirante. A este aspirante se le enviará la contestación, en cuanto si es registrado o no.

Alta de Usuario Especial y Frecuente (Unidad de Proceso 15)

El registro de usuario especial y el frecuente se auxiliará del sitio web, en donde existirán las formas de solicitud para cada tipo de usuario. Estos usuarios deberán llenar la solicitud correspondiente, que será enviada al laboratorio para su recepción y después de ser evaluadas, se mandará la respuesta a la solicitud, al usuario correspondiente.

Convenio (Unidad de Proceso 16)

Aquella persona que desee ser usuario especial, adicionalmente a lo establecido en la unidad de proceso 15, deberá establecer formalmente un convenio con el laboratorio, el cual lo registrará.

La figura 5.3 presenta un fragmento del diagrama de procesos obtenido para la Base de Datos de Imágenes Satelitales. En la imagen se tiene la unidad de proceso 1 y la unidad de proceso 2, las cuales en su conjunto representan las actividades de que se realizan para llevar acabo la descarga, el registro, almacenamiento temporal, la compresión y finalmente el respaldo de datos crudos.

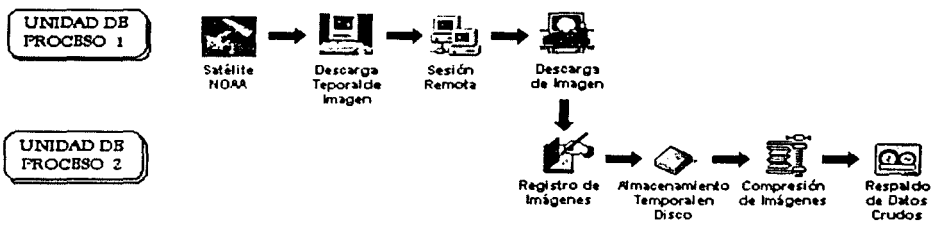


Figura 5.2 Diagrama de Procesos

5.3.3 DIAGRAMA DE FLUJO DE DATOS

El diagrama de flujo de datos (DFD) se utiliza para el modelado de procesos. Esta técnica de modelado muestra el flujo y la transformación de los datos sin detenerse en detalles de estructura o tipo de datos. Representa con claridad donde ocurren las transacciones y transformaciones del sistema. Los diagramas de flujo de datos no están designados para mostrar flujo de materiales, sólo datos. El DFD sustenta los conceptos de proceso, flujo de datos, almacén y entidad externa. Los términos de la notación se definen como sigue:

- *Proceso.* Representa una actividad dentro del sistema de información. En particular, un proceso puede generar, usar, manipular o destruir información.

- *Flujo de datos.* Es un intercambio de información entre procesos. Se recalca que los flujos de datos no representan flujos de control, como la activación de procesos. Indican, en cambio, paquetes discretos de datos que fluyen hacia adentro o hacia afuera del proceso.

- *Almacén de datos.* Es un depósito de información. Los archivos temporales, tablas de consulta, formularios de papel, formularios electrónicos y registros permanentes, todos éstos, pueden representarse como almacenes de datos.

- *Entidad externa.* Es un usuario externo del sistema de información, que puede ser el originador o receptor de los flujos de datos o de los almacenes de datos.

Para describir el diagrama de flujo de datos de la Base de Datos de Imágenes Satelitales (ver anexo B), se tomó la perspectiva del LOF que utilizará el sistema de información, por lo que todas las actividades que contribuyen a la manipulación de los datos dentro del mismo LOF, se modelan a través de procesos. Las entidades externas representan las fronteras del sistema de información, esto es, los generadores o receptores de la información que se manipulan dentro del sistema.

En el diagrama se observan las diferentes entidades externas que solicitan y proporcionan información importante del sistema, la primera entidad externa es "satélite", la cual proporciona las imágenes a partir de las cuales se realizan los diferentes procesos para la obtención de la imágenes satelitales.

La siguiente entidad, representa las peticiones que provendrán del sitio web, las cuales serán clasificadas, atendidas y se colocara la información solicitada en el sitio web para su utilización.

Las otras dos entidades representan a los usuarios que solicitaran información al sistema. Estas entidades son: "Usuario LOF e ICML" y "Usuarios Especial y Frecuente". Estos usuarios serán registrados en el sistema, para que posteriormente hagan las peticiones deseadas.

La figura 5.3 muestra únicamente los procesos que se realizan para llevar a cabo la obtención de la información cruda, la que se obtiene de la entidad externa "Satélite", posteriormente ya que se cuenta con la información cruda, esta es registrada en la bitácora correspondiente, almacenada y comprimida para su posterior utilización en la obtención de otras imágenes.

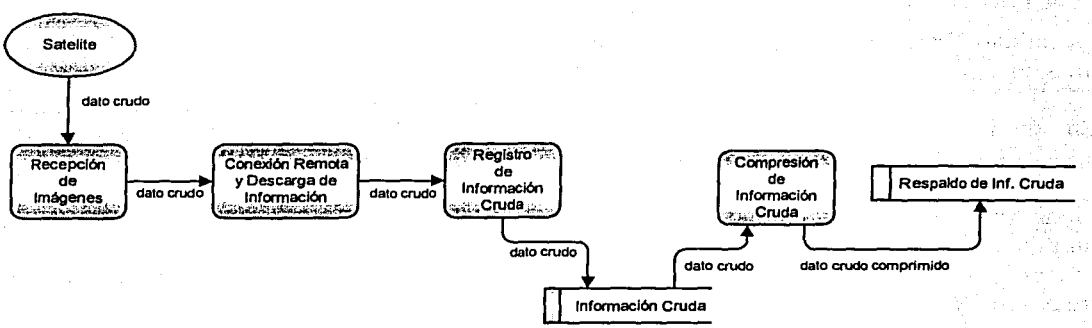


Figura 5.3 Diagrama de Flujo de Datos

5.3.4 DIAGRAMA JERÁRQUICO

Este diagrama muestra de manera gráfica la relación entre las diferentes unidades de procesamiento, describiendo la jerarquía de las unidades y los datos que serán transmitidos entre ellos. La descomposición de las unidades complejas en unidades inferiores, guía al diagrama jerárquico. Así se pueden representar estas unidades, además de reducir su complejidad. La descomposición esta terminada cuando quedan las unidades elementales.

El diagrama jerárquico obtenido para la base de datos de imágenes satelitales, esta formado por 4 niveles de jerarquía (ver anexo C). El primer nivel lo forma la unidad principal denominada "Imágenes Satelitales", de esta unidad se desprenden 6 sub-unidades que representan a cada una de las unidades de proceso mostradas en el diagrama de procesos.

El segundo nivel en la jerarquía lo componen las siguientes unidades de procesos:

- Recibir peticiones
- Procesar imágenes
- Descargar y almacenar imágenes
- Incrementar acervo fotográfico
- Estudiar imágenes
- Definir usuarios del sistema

La unidad "Recibir peticiones" se compone a su vez de dos sub-unidades: "Organizar peticiones de información" y "Consultar el sistema", además, de esta ultima se desprenden las siguientes unidades: "Revisar consultas" y "Cumplir petición".

La siguiente unidad en el segundo nivel es "Procesar imágenes" la cual esta formada por las unidades de nivel inferior: "Calcular imágenes SST", "Calcular imágenes semanales" y "Calcular promedios mensuales".

La unidad "Descargar y almacenar imágenes" la forman las unidades "Seleccionar datos" y "Registrar imagen".

La unidad "incrementar acervo fotográfico" es llevada a cabo por las sub-unidades "Crear imágenes para foto" y "Tomar fotografía".

En seguida se tiene la unidad "Estudiar imágenes" que se compone de las unidades del tercer nivel, "Definir casos de estudio" y "Analizar imágenes de estudio".

Finalmente se tiene la unidad "Definir usuarios del sistema" que se lleva a cabo por la unidades: "Inscribir usuario LOF", "Inscribir usuario ICML" y la unidad "Inscribir usuarios especial". Donde esta última se forma de las unidades del cuarto nivel, "Inscribir usuario frecuente" y "Formalizar convenio".

Cabe señalar que cada una de las unidades que forman el tercer y cuarto nivel, se encuentran descritas en el apartado anterior, Diagrama de Procesos; indicando con la notación *UP # unidad* (*UP* = Unidad de Proceso y *# unidad* es el número de la unidad a la que se hace referencia en el diagrama de procesos).

En la siguiente figura se muestran tres niveles de la jerarquía contenidos en el diagrama jerárquico, obtenido para la Base de Datos de Imágenes Satelitales.

En esta se tiene como unidad padre a "Imágenes Satelitales" de la cual se origina la unidad hijo "Descargar y Almacenar Imágenes" que a su vez se forma de las unidades 1 y 2 , que son: "Seleccionar Datos Crudos" y "Registrar Imagen (dato crudo)", respectivamente.

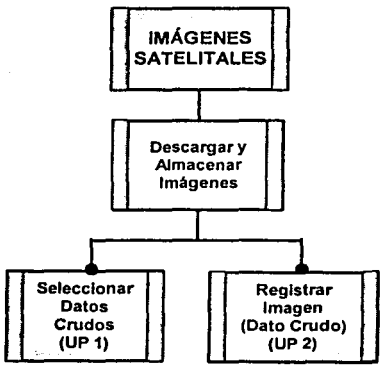


Figura 5.4 Diagrama Jerárquico

5.3.5 DIAGRAMA ENTIDAD/RELACIÓN

La pareja Entidad-Relación es la parte mas importante en el modelado de datos. Estas parejas se pueden representar gráficamente mediante el Diagrama Entidad/Relación (DER), que es utilizado ampliamente para el diseño de sistemas de bases de datos relaciónales. Se identifica un conjunto de elementos importantes para el DER: entidades, atributos, relaciones y varios indicadores de tipo de dato. El objetivo primordial del DER es representar entidades de datos y sus relaciones.

La notación DER ya se ha introducido en secciones anteriores. Las entidades de datos son representadas por un rectángulo con esquinas redondeadas y etiquetado con el nombre de la entidad en singular. Las relaciones se indican mediante una línea etiquetada conectando entidades. Las conexiones entre entidades de datos y relaciones se establecen mediante una variedad de símbolos especiales que indican la cardinalidad y el grado de la relación.

Enseguida se muestra un fragmento del DER obtenido para la Base de Datos de Imágenes Satelitales (ver anexo D). La figura 5.5 presenta las relaciones existentes entre los subtipos "Imagen Promedio Mensual" e "Imagen Promedio Semanal" contenidos dentro de la Superentidad "Tipo Imagen", con la entidad "Imagen Bitácora"., que se leen como sigue:
Cada IMAGEN_BITACORA debe ser para una y solo una IMAGEN_PROMEDIO_MENSUAL.

Cada IMAGEN_PROMEDIO_MENSUAL puede estar proviniendo de una o más IMAGENES_BITACORAS.

Cada IMAGEN_BITACORA debe ser para una y solo una IMAGEN_PROMEDIO_SEMANAL.

Cada IMAGEN_PROMEDIO_SEMANAL puede estar proviniendo de una o más IMAGENES_BITACORAS.

Donde además de involucrar subtipos también se observa que las dos relaciones que se originan de la entidad "IMAGEN_BITACORA", están contenidas en un arco, lo que indica que solo se puede presentar una relación a la vez y no el caso de que se tengan ambas relaciones en un mismo instante de tiempo.

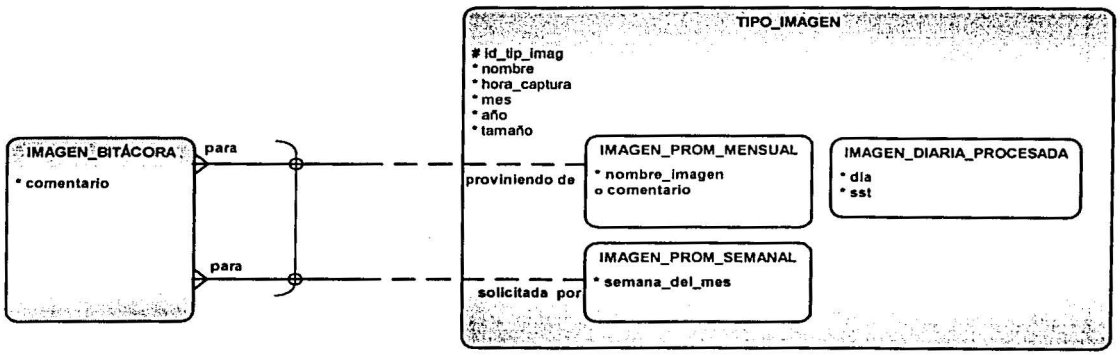


Figura 5.5 Diagrama Entidad - Relación

5.4 ETAPA DE DISEÑO

Retomando los requerimientos, así como los modelos obtenidos en la etapa de análisis, esta etapa tiene como objetivo plasmar ambos utilizando las herramientas con que se dispone.

5.4.1 DISEÑO DE RED

A partir del equipo de trabajo con que se contaba en el LOF, además del nuevo equipo adquirido con los recursos destinados para ello, se estableció un diseño de red que permitiera la correcta interacción entre el servidor de la Base de Datos y las distintas máquinas cliente, incluyendo la fase de pruebas que se realizaría posteriormente, como parte de la creación de la Base de Datos.

En el Servidor Sun Ultra 10 con Sistema Operativo Solaris 4.0 fue instalado el Sistema Manejador de Bases de Datos Oracle 8i. Este servidor será el encargado de proporcionar el acceso a la Base de Datos, es decir, se encargará de proporcionarle seguridad y protección de accesos no autorizados y de usuarios malintencionados dentro de la red de internet.
Las dos computadoras con procesador pentiumIII (con S.O. windows 2000), y una existente con windows 98 fueron colocadas en red a través del cableado con el que se cuenta en el ICML, lo que permitió al grupo de trabajo contar con ingresos al servidor de la base de datos, además de que proporciono la posibilidad de compartir los diversos documentos generados en el análisis y desarrollo del sistema, incluyendo la opción de imprimir la documentación obtenida. El diseño de red establecido para el desarrollo del sistema, se muestra en la figura 5.6.

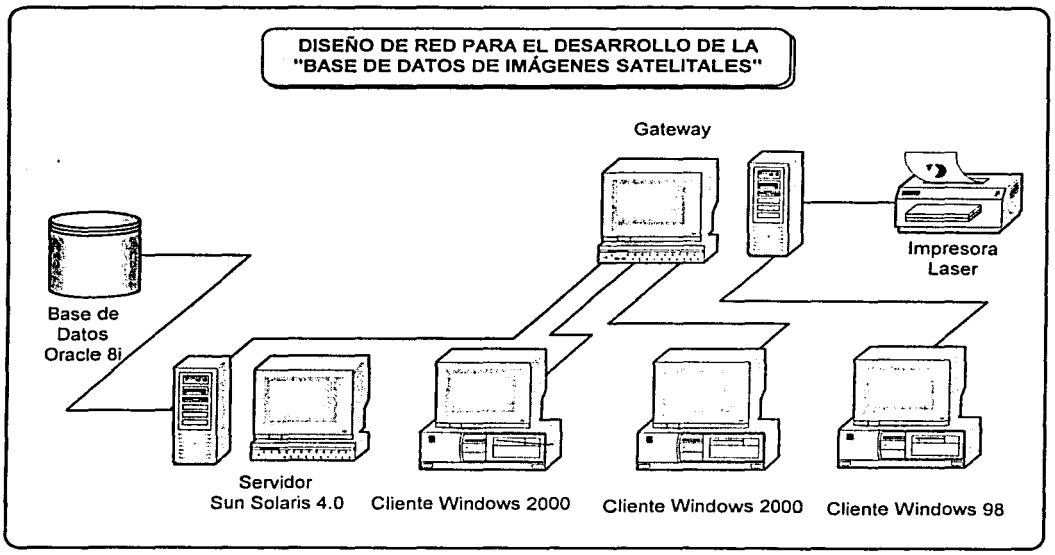


Figura 5.6 Diseño de Red

5.4.2 DISEÑO DE LA BASE DE DATOS

Cuando se diseña, se toman los requerimientos detallados del negocio y se busca la mejor manera de utilizar el ambiente tecnológico para cumplirlos. Diseñar una Base de Datos relacional involucra tomar un modelo de información, el cual proporciona una vista de los requerimientos de información del negocio, y transformarlo en una representación de software práctica:

1. Mapear entidades simples a tablas
2. Mapear atributos a columnas
3. Mapear identificadores a llaves primarias
4. Mapear relaciones a llaves foráneas

La palabra SISTEMA implica una colección de aplicaciones bien integrada. Si estas son construidas casualmente, las oportunidades de integración son pocas. Lo mínimo que puede pasar son variaciones confusas, pero en caso extremo se pueden presentar redundancias incontrolables y una terrible presentación.

Antes de empezar a convertir el Diagrama Entidad/Relación a un diseño inicial, se debe asegurar su calidad completamente.

- Todas las entidades deben tener descripción.
- Todas las entidades deben tener un nombre.
- Todas las relaciones muchos a muchos deben estar resueltas.
- Todas las entidades deben tener un identificador.
- La información volumétrica debe ser presentada.
- Las reglas de integridad referencial deben ser colocadas.
- Todos los atributos opcionales deben ser especificados.
- Todos los atributos deben tener un formato.
- El modelo debe estar completamente normalizado.

Se puede convertir una entidad a una tabla de manera casi directa o se pueden acondicionar las entidades a tablas para que aparezcan adecuadamente en el ambiente de la implantación. Todas las relaciones del Diagrama Entidad/Relación deben ser convertidas a llaves foráneas, lo cual significa que se deben crear columnas extra en la tabla.

Se debe crear una columna dentro de la tabla, para cada atributo en la entidad. En un modelo conceptual, el nombre de un atributo puede ser tan largo como se quiera y puede contener espacios. Los nombres de columnas deben ser concisos y utilizar guiones bajos para reemplazar los espacios.

Convertir cada identificador único de una entidad en columna de llave primaria o llave secundaria, según sea el caso. Todas las columnas que serán llaves primarias deben ser obligatorias y únicas, se identifican con PK en el renglón TIPO LLAVE.

Dentro del diseño, se pueden crear índices para acelerar el acceso y hacerlo más eficiente. Se pueden crear vistas para restringir el ingreso a partes de la Base de Datos, también hacer rutas de acceso complejas y consultas transparentes. Se pueden crear secuencias para permitirle a la máquina localizar números únicos de columnas especificas.

5.4.2.1 MAPEANDO ATRIBUTOS A COLUMNAS

Se inicio por hacer una lista de los atributos de la entidad, que serán los encabezados de cada columna, utilizando el nombre del atributo donde fue posible, para facilitar el traspaso del Diagrama Entidad-Relación:

- Se evito el uso de palabras reservadas de SQL, por ejemplo, NUMBER, CHAR, etc. como nombres de columnas.
- Se utilizaron abreviaciones consistentes para evitar confusiones.
- Se ocuparon nombres cortos significativos para las columnas, porque ayudan a reducir el tiempo requerido para la ejecución de una sentencia SQL.
- No se utilizo el nombre de la tabla como parte del nombre de la columna.

Respecto a los arcos obtenidos en el DER, estos tienen diversas alternativas de llaves foráneas. Las siguientes son dos formas de diseño al mapear arcos a llaves foráneas:

- Explícito
- Genérico

En el diseño de la Base de Datos de Imágenes Satelitales, se eligió el mapeo de los arcos, utilizando la forma genérica, es decir, este diseño crea una sola columna de llave foránea y una columna adicional para usarla como bandera en el tipo.

En las relaciones que son exclusivas, como lo son las contenidas en un arco, solo un valor en las FK existe para cada renglón en la tabla. Si todas las relaciones en el arco son obligatorias, entonces se debe hacer la columna de llave foránea NO NULA. Para poder implantar el arco genérico se aseguro que todas las llaves foráneas compartían el mismo formato.

5.4.2.2 SUPERTIPOS

Un supertipo es una entidad que puede ser dividida en pequeños grupos mutuamente exclusivos. Un supertipo puede tener atributos o solo ser utilizado como el nombre de un grupo.

5.4.2.3 SUBTIPOS

Un subtipo es una entidad que representa la composición de un grupo en un supertipo. Cada subtipo puede tener sus propios atributos, pero inherentemente tendrá todos los atributos asignados al supertipo.

En el DER obtenido para la Base de Datos de Imágenes Satelitales, se obtuvieron los supertipos: "TIPO IMAGEN" e "INFORMACION CRUDA", los cuales a su vez tienen los subtipos "IMAGEN PROMEDIO SEMANAL", "IMAGEN PROMEDIO MENSUAL", "IMAGEN DIARIA PROCESADA" Y "IMAGEN PARA FOTO" e "IMAGEN SIN PROCESAR", respectivamente.

Al crear las tablas correspondientes a estos supertipos se mapearon todos los atributos de cada subtipo a una sola tabla para el supertipo. También se adiciono la columna de tipo T_IMAGEN para identificar a que subtipo corresponde cada renglón.

La propiedad obligatoria de columnas que se originan de atributos obligatorios en un subtipo, fue forzada por restricciones o triggers de la Base de Datos. Y se considero utilizar el diseño en una sola tabla, ya que es recomendable su utilización cuando los subtipos tienen pocos atributos y relaciones, como es el caso.

Es importante mencionar que el formato que se sigue en el diseño (tablas) es el siguiente:

- El nombre de la tabla irá en mayúsculas y en plural.
- La primera columna que conformara cada una de las tablas, lleva los siguientes datos:

  - *Tipo Llave:* Esta fila contendrá únicamente la información que indicará que campo es la llave primaria de la tabla, así como las llaves foráneas.

  - *No Nulo/Único:* En esta fila se indica que campos serán no nulos y únicos. En los renglones NULOS se ingresa la opcionalidad de cada atributo y se mapea cada atributo obligatorio como NO NULO (NN), además de dejar los atributos opcionales vacíos.

  - *Ejemplos:* En este renglón se incluyen datos de ejemplo para documentos demostrativos del contenido de la tabla. Los datos presentan el contenido de la tabla en una forma visual y ayudan a validar las columnas.

Enseguida se muestra el diseño de las tablas (ver anexo E):

Nombre de la tabla: **TIPOS_IMAGEN**

Columna	id_tipo_imag	nombre	hora_captura	mes	año
Tipo Llave					
Nulo / Único	NN	NN	NN	NN	NN
Ejemplos	4022	n14.010309.1137.sst	1137	Marzo	2001
	3040	1201.005.01	0005	Diciembre	2000
	3010	0100.2030.compuesta	2030	Enero	2000

Columna	tamaño	t_imagen	id_inf_cruda	hora	dia
Tipo Llave			FK1	FK2	
Nulo / Único	NN	NN			
Ejemplos	98,466256	01	2036	01	1
	200,190	02			
	198,302	03			

Columna	sst	semana_del_mes	nombre_imagen	comentario
Tipo Llave				
Nulo / Único				
Ejemplos	sst			
		1		
			Mensual_Enero_2000	Buena Resolución

Figura 5.7 Tipos_Imágen

Nombre de la tabla: **IMAGENES_BITACORAS**

Columna	id_tipo_imag	id_bitacora	comentario
Tipo Llave	PK, FK	PK, FK	
Nulo / Único	NN, U	NN, U	NN
Ejemplos	3010	20	Se completo la semana
	3040	20	Mala semana
	4022	25	Excelente Mensual

Figura 5.8 Imágenes_Bitácoras

5.5 ETAPA DE CONSTRUCCIÓN

Oracle es una base de datos relacional. El lenguaje utilizado para acceder a las bases de datos relacionales es el denominado *Lenguaje de Consulta Estructurado* (SQL). SQL es un lenguaje flexible y eficiente, con características que han sido diseñadas para la manipulación y examen de datos relaciónales.

SQL es un *lenguaje de cuarta generación*, lo que quiere decir que el lenguaje describe lo que debe hacerse, pero no la manera de llevarlo a cabo. Cada lenguaje tiene sus ventajas e inconvenientes.

Los 4GL como SQL son, por regla general, bastante simples y tienen menos órdenes. Asimismo aíslan al usuario de los algoritmos y estructuras de datos subyacentes. En algunos casos, sin embargo, las estructuras procedimentales disponibles en los 3GL resultan útiles para expresar un determinado programa.

Aquí es donde entra el PL/SQL, que combina la potencia y flexibilidad de un lenguaje SQL con las estructuras procedimentales de un 3GL. PL/SQL significa Procedural Language/SQL. Como su propio nombre lo indica, PL/SQL amplía la funcionalidad de SQL añadiendo estructuras de las que pueden encontrarse en otros lenguajes procedimentales, como:

- Variables y tipos (tanto predefinidos como definidos por el usuario).
- Estructuras de control, como bucles y órdenes IF-THEN-ELSE.
- Procedimientos y funciones.
- Tipos de objetos y métodos (en PL/SQL versión 8 o superior).

Las construcciones procedimentales están perfectamente integradas con Oracle SQL, lo que da como resultado un lenguaje potente y estructurado. PL/SQL es un sofisticado lenguaje de programación que se utiliza para acceder a bases de datos Oracle desde distintos entornos.

Está integrado con el servidor de base de datos, de modo que el código PL/SQL puede ser procesado de forma rápida y eficiente. Integra tanto las estructuras procedimentales necesarias como el acceso a bases de datos. El resultado es un lenguaje robusto y potente, bien adaptado al diseño de aplicaciones complejas

5.5.1 CREACIÓN DE TABLAS

Con la combinación de SQL y PL/SQL se realizo la construcción de la Base de Datos de Imágenes Satelitales, es decir las entidades, atributos y relaciones obtenidas en el DER y en el diseño de la base de datos, fueron mapeadas a tablas, columnas, llaves primarias y llaves foráneas, mediante sentencias de SQL y PL/SQL (ver anexo F).

Tal como se observa en el siguiente fragmento de código SQL, que muestra como se creo la superentidad "TIPO_IMAGEN", incluyendo sus correspondientes subtipos, además de la entidad "IMAGEN_BITACORA" y las relaciones existentes entre estas entidades, como se observo en la figura 5.5.

Tabla "TIPOS_IMAGEN":

```
CREATE TABLE TIPOS_IMAGEN (
ID_TIP_IMAG VARCHAR2(10) NOT NULL,
NOMBRE VARCHAR2(7) NOT NULL,
HORA_CAPTURA VARCHAR2(5) NOT NULL,
MES NUMBER(2) NOT NULL,
ANIO NUMBER(4) NOT NULL,
TAMANIO NUMBER(9) NOT NULL,
T_IMAGEN NUMBER(2) NOT NULL,
INF_CRUD_ID_INF_CRUDA VARCHAR2(10),
TIP_HR_HORA VARCHAR2(3),
DIA NUMBER(2),
SST VARCHAR2(3),
SEM_DEL_MES VARCHAR2(2),
NOMBRE_IMAGEN BLOB,
COMENTARIO VARCHAR2(50)
)/
```

Tabla "IMAGENES_BITACORAS":

```
CREATE TABLE IMAGENES_BITACORAS (
BIT_MESTUD_ID_BITACORA NUMBER(4) NOT NULL,
COMENTARIO VARCHAR2(50) NOT NULL,
T_IMAGEN_ID_TIP_IMAG VARCHAR2(10) NOT NULL
)/
```

5.5.2 CREACIÓN DE LLAVES PRIMARIAS Y FORÁNEAS

```
ALTER TABLE TIPOS_IMAGEN ADD (
CONSTRAINT T_IMAGEN_PK PRIMARY KEY
(ID_TIP_IMAG)
)
/

ALTER TABLE IMAGENES_BITACORAS ADD (
CONSTRAINT IMAG_BITAC_PK PRIMARY KEY
(BIT_MESTUD_ID_BITACORA)
)
/

ALTER TABLE TIPOS_IMAGEN ADD (
CONSTRAINT T_IMAGEN_TIP_HR_FK FOREIGN KEY
(TIP_HR_HORA) REFERENCES TIPOS_HORAS (HORA)
)
/

ALTER TABLE TIPOS_IMAGEN ADD (
CONSTRAINT T_IMAGEN_INF_CRUD_FK FOREIGN KEY
(INF_CRUD_ID_INF_CRUDA) REFERENCES INFORMACIONES_CRUDAS (ID_INF_CRUDA)
)
/
```

```
ALTER TABLE IMAGENES_BITACORAS ADD (
CONSTRAINT IMAG_BITAC_BIT_MESTUD_FK FOREIGN KEY
(BIT_MESTUD_ID_BITACORA) REFERENCES BITACORAS_DE_MES_EN_ESTUDIOS
(ID_BITACORA)
)
/
```

De esta forma, se crearon todas las tablas necesarias en la construcción de la base de datos, creando primero las tablas, estableciendo la llave primaria de cada tabla y finalmente las relaciones entre ellas (llaves foráneas).

5.5.3 CREACIÓN DE TRIGGERS

Un trigger es un procedimiento almacenado o un bloque en PL/SQL asociado a una tabla, que se ejecuta automáticamente cuando se produce uno o más sucesos en SQL sobre dicha tabla (ver anexo F). Los eventos frente a los que reacciona incluyen operaciones de inserción, actualización y borrado ya que los triggers se pueden ejecutar para la tabla como un todo o para cada fila afectada de la tabla . A continuación se muestra la estructura del trigger DISPARA1_TIMAG para la tabla TIPOS_IMAGEN :

```
CREATE OR REPLACE TRIGGER DISPARA1_TIMAG
BEFORE INSERT ON TIPOS_IMAGEN
FOR EACH ROW
DECLARE
       T_IMG NUMBER(2);
       TEMP1 INTEGER;
       TEMP2 INTEGER;
       TEMP3 INTEGER;
       TEMP4 INTEGER;
       TEMP5 INTEGER;
       TEMP6 INTEGER;
BEGIN
       T_IMG:=:new.T_IMAGEN;
       TEMP1:=INSTR(:new.ID_TIP_IMAG,'-',1,1);
       TEMP2:=INSTR(:new.ID_TIP_IMAG,'-',1,2);
       TEMP3:=LENGTH(:new.ID_TIP_IMAG);
       TEMP4:=INSTR(UPPER(:new.ID_TIP_IMAG),'ID',1,1);
       TEMP5:=INSTR(UPPER(:NEW.ID_TIP_IMAG),'IS',1,1);
       TEMP6:=INSTR(UPPER(:NEW.ID_TIP_IMAG),'IM',1,1);
       IF T_IMG=1 AND TEMP1=3 AND TEMP3=10 AND TEMP2=6 AND TEMP4=4 THEN
              :new.SEM_DEL_MES:=NULL;
              :new.COMENTARIO:=NULL;
              :new.NOMBRE:=INITCAP(LOWER(:new.NOMBRE));
       ELSIF T_IMG=2 AND TEMP1=3 AND TEMP3=10 AND TEMP2=6 AND TEMP5=4 THEN
              :new.DIA:=NULL;
              :new.SST:=NULL;
              :new.COMENTARIO:=NULL;
              :new.NOMBRE:=INITCAP(LOWER(:new.NOMBRE));
       ELSIF T_IMG=3 AND TEMP1=3 AND TEMP3=10 AND TEMP2=6 AND TEMP6=4 THEN
              :new.DIA:=NULL;
              :new.SST:=NULL;
```

```
        :new.SEM_DEL_MES:=NULL;
        :new.NOMBRE:=INITCAP(LOWER(:new.NOMBRE));
    ELSE
        :new.ID_TIP_IMAG:=NULL;
    END IF;
END DISPARA1_TIMAG;
/
```

El uso de OR REPLACE permite sobreescribir un trigger existente. Si se omite, y el trigger existe, se producirá, un error.

El modificador FOR EACH ROW indica que el trigger se disparará cada vez que se desee hacer operaciones sobre una fila de la tabla. Si se acompaña del modificador WHEN, se establece una restricción; el trigger solo actuará, sobre las filas que satisfagan la restricción.

5.5.4 CREACIÓN DE CONSTRAINTS

La integridad de dominios define el rango o intervalo de valores aceptables para una columna. Además de utilizar los tipos de datos de columna, Oracle soporta dos tipos de restricciones de integridad que permiten que se limiten más el dominio de una columna.

Una columna no puede tener una restricción no nula para eliminar la posibilidad de que haya nulos (valores ausentes) en la columna.

Se puede utilizar una restricción de comprobación para declarar una regla de integridad de dominio compleja como parte de una tabla. Una restricción de comprobación contiene comúnmente una lista explícita de valores para una columna (ver anexo F).

Ahora se muestra la estructura del constraint CHECA_TIMAG para la tabla TIPOS_IMAGEN:

```
ALTER TABLE TIPOS_IMAGEN
ADD CONSTRAINT checa_timag
CHECK(T_IMAGEN IN (01,02,03)) DEFERRABLE
/
```

El uso de la sentencia CHECK en conjunto con DEFERRABLE, es por que Oracle puede retrasar la imposición de una restricción de integridad diferible justo antes de la validación de una transacción.

Cuando se valida una transacción y la transacción tiene datos de tabla modificados de tal manera que no cumple las restricciones de integridad, se deshace automáticamente toda la transacción (es decir, los efectos de todas las sentencias de la transacción).

5.5.5 CREACIÓN DE ARCOS

Es un caso particular de opcionalidad múltiple entre tablas (o una u otra) esto es, aunque una tabla tenga múltiples relaciones con otras tablas englobadas en un arco, solo puede aceptar una y solo una relación a la vez mediante sentencias de SQL y PL/SQL (anexo F).

En seguida se muestra la estructura del trigger ARCO_IMAG_BITACORAS para la tabla IMÁGENES_BITACORAS:

```
CREATE OR REPLACE TRIGGER ARCO_IMG_BITACORAS
BEFORE INSERT ON IMAGENES_BITACORAS
FOR EACH ROW
DECLARE
ID_TIPO VARCHAR2(10);
TEMP INTEGER;
TEMP2 INTEGER;
TEMP4 INTEGER;
TEMP5 INTEGER;
TEMP6 INTEGER;
BEGIN
      TEMP:=LENGTH(:new.T_IMAGEN_ID_TIP_IMAG);
      TEMP2:=INSTR(:new.T_IMAGEN_ID_TIP_IMAG,'-',1,1);
      TEMP4:=INSTR(:new.T_IMAGEN_ID_TIP_IMAG,'-',1,2);
      TEMP5:=INSTR(UPPER(:new.T_IMAGEN_ID_TIP_IMAG),'IS',1,1);
      TEMP6:=INSTR(UPPER(:new.T_IMAGEN_ID_TIP_IMAG),'IM',1,1);
      IF TEMP=10 AND TEMP2=3 AND TEMP4=6 AND  TEMP5=4 THEN
            SELECT ID_TIP_IMAG INTO ID_TIPO FROM TIPOS_IMAGEN
            WHERE ID_TIP_IMAG=:new.T_IMAGEN_ID_TIP_IMAG;
      ELSIF TEMP=10 AND TEMP2=3 AND TEMP4=6 AND  TEMP6=4 THEN
            SELECT ID_TIP_IMAG INTO ID_TIPO FROM TIPOS_IMAGEN
            WHERE ID_TIP_IMAG=:new.T_IMAGEN_ID_TIP_IMAG;
      ELSE
            :new.BIT_MESTUD_ID_BITACORA:=NULL;
      END IF;
      EXCEPTION
      WHEN NO_DATA_FOUND THEN
            :new.BIT_MESTUD_ID_BITACORA:=NULL;
END ARCO_IMG_BITACORAS;
/
```

5.5.6 CREACIÓN DE PROCEDIMIENTOS ALMACENADOS

Un procedimiento almacenado es un conjunto de instrucciones en PL/SQL, que pueden ser llamado usando el nombre que se le haya asignado. El uso de OR REPLACE permite sobreescribir un procedimiento existente. Si se omite, y el procedimiento ya existe, se producirá un error.

Inicialmente se creo el directorio IMAGENES que hace referencia a una carpeta dentro del sistema operativo, donde se colocaran imágenes que se deseen insertar en la base de datos, además de otorgar los permisos de lectura para este directorio. Lo anterior se realizo con las siguientes instrucciones:

```
CREATE OR REPLACE DIRECTORY IMAGENES AS '/export/home/oracle/imgs';
grant read on directory imagenes to public with grant option;
```

El siguiente procedimiento, CARGA_IMAGEN, recibe como parámetros cada uno de los campos que forman parte del subtipo IMAGEN_MENSUAL, incluyendo el nombre de la imagen que será insertada en la tabla TIPOS_IMAGEN. Los modificadores `IN`, `OUT`, `IN OUT` indican si el parámetro es de entrada, salida o ambos.

Dentro del cuerpo del procedimiento se hace uso de dos tipos de datos LOB (BFILE y BLOB). Con el tipo de dato BFILE se hace referencia a la imagen que esta en el sistema operativo y posteriormente la imagen a la que apunta se pasa al tipo de dato BLOB, para finalmente realizar la inserción en la tabla correspondiente.

Enseguida se muestra la estructura del procedimiento almacenado CARGA_IMAGEN para la tabla TIPOS_IMAGEN:

```
CREATE OR REPLACE PROCEDURE CARGA_IMAGEN (
id                  tipos_imagen.id_tip_imag%TYPE,
nombrei             tipos_imagen.nombre%TYPE,
horac               tipos_imagen.hora_captura%TYPE,
mesi                tipos_imagen.mes%TYPE,
anioi               tipos_imagen.anio%TYPE,
tamanioi            tipos_imagen.tamanio%TYPE,
tipoi               tipos_imagen.t_imagen%TYPE,
id_cruda            tipos_imagen.inf_crud_id_inf_cruda%TYPE,
tipo_hora           tipos_imagen.tip_hr_hora%TYPE,
nombreimagen              IN VARCHAR2,
comenti             tipos_imagen.comentario%TYPE) AS
apuntador_bfile     BFILE;
apuntador_blob      BLOB;
tamanio_imagen      INTEGER;
BEGIN
      DBMS_OUTPUT.ENABLE(1000000);
      apuntador_bfile := BFILENAME('IMAGENES',nombreimagen);
      DBMS_LOB.FILEOPEN(apuntador_bfile);
      tamanio_imagen := DBMS_LOB.GETLENGTH(apuntador_bfile);
      DBMS_OUTPUT.PUT_LINE('El  tamanio  de  la  imagen  leida  es:  '||
tamanio_imagen);
      INSERT INTO TIPOS_IMAGEN VALUES (
                        id,nombrei,horac,mesi,anioi,tamanioi,tipoi,
                        id_cruda,tipo_hora,'','','',EMPTY_BLOB(),comenti)
                        RETURNING nombre_imagen INTO apuntador_blob;
      DBMS_LOB.LOADFROMFILE(apuntador_blob,
                              apuntador_bfile,tamanio_imagen);
      DBMS_LOB.FILECLOSE(apuntador_bfile);
      COMMIT;
```

```
      EXCEPTION
            WHEN OTHERS THEN
                  DBMS_OUTPUT.PUT_LINE('Tipo de Excepcion OTHERS: ' ||
sqlerrm);
END Carga_Imagen;
/
```

La forma de llamar al procedimiento anterior es la siguiente:

```
BEGIN
CARGA_IMAGEN(
'96-IM-
0002','MENSUAL','09:11',10,1997,123456789,03,'V','imagen1.gif','Excelen
te');
END;
```

5.6 ETAPA DE PRUEBAS

En esta etapa se realizaron una serie de inserciones en la base, de datos contenidos en documentos y bitácoras del LOF, con el objetivo de realizar consultas y observar si la información obtenida era la esperada. Enseguida se presentan ejemplos de sentencias utilizadas para ingresar los datos mencionados.

5.6.1 INSERCIÓN DE DATOS

TABLA "SATELITES":

```
insert into satelites values(
'n12','noaa-12','Primer Satelite'
)
/

insert into satelites values(
'n14','noaa-14','Segundo Satelite'
)
/
```

TABLA "INFORMACIONES_CRUDAS":
imagen sin procesar

```
insert into informaciones_crudas values(
'96-SP-0001','n12','01-sep-1999',123456789,'I_SPROC',10,20,'V',0
)
/
```

imagen para foto

```
insert into informaciones_crudas values(
'96-IF-0002','n12','01-sep-1999',123456789,'I_PFOTO','','','','ter'
)
/
```

TABLA "TIPOS_HORAS":

```
insert into tipos_horas values(
'h01','MC',0.25,0.2,0.5,0.25,-0.5,01.2,1.2,1.5,1.25,
'00:30'
)
/
```

TABLA "TIPOS_IMAGEN":
imagen diaria procesada

```
insert into tipos_imagen values(
'96-ID-0001','DIARIA','00:37',01,1996,123456789,01,'96-SP-0001',
'h01',01,'sst','',EMPTY_BLOB(),''
)
/
```

imagen promedio semanal

```
insert into tipos_imagen values(
'96-IS-0001','SEMANAL','11:37',12,1999,123456789,02,'96-SP-0001',
'','','','1a',EMPTY_BLOB(),''
)
/
```

imagen promedio mensual

```
insert into tipos_imagen values(
'96-IM-0001','MENSUAL','09:11',10,1997,123456789,03,'96-SP-0001',
'','','','',EMPTY_BLOB(),'Excelente Imagen'
);
```

5.6.2 CONSULTA DE INFORMACIÓN

Los siguientes son ejemplos de consultas que se realizaron a la base de datos, con el fin de comprobar que se obtenían los resultados esperados.

1.¿Cúal es el nombre de la persona que tiene el convenio 0002?.

```
SQL>select USUARIOS.nombre from USUARIOS,CONVENIOS where
conv_num_conv=num_conv;

NOMBRE
--------
Irlanda
```

2. ¿ A qué semana corresponde la imagen con identificador 96-ID-0001?.

```
SQL>select SEMANAS_EN_ESTUDIO.id_sem_est from SEMANAS_EN_ESTUDIO,
IMAGENES_DIA_SEMANAS_ESTUDIO,TIPOS_IMAGEN where
id_sem_est=sem_estudi_id_sem_est and t_imagen_id_tip_imag=id_tip_imag;
```

```
ID_SEM
------
004-96
```

3. ¿ Cuál es el país de residencia, así como el cargo del usuario con mail irlanda@unam.mx?

```
SQL>select PAISES.nombre,CARGOS.nombre from PAISES, CARGOS, USUARIOS,
DETALLES_LUGAR where PAISES.id_pais=DETALLES_LUGAR.pais_id_pais and
USUARIOS.DET_LUGAR_PAIS_ID_PAIS=DETALLES_LUGAR.pais_id_pais and
cargos.id_cargo=usuarios.cargo_id_cargo and email='irlanda@unam.mx';

NOMBRE               NOMBRE
-------------------- ---------------
Mexico               Tesista
```

4.¿Qué imagen tienen el tipo de hora h01 y en que formato se encuentra su respaldo?.

```
SQL>select TIPOS_IMAGEN.id_tip_imag from TIPOS_IMAGEN, TIPOS_HORAS,
TIPOS_IMAGENES_RESPALDOS,RESPALDOs where TIPOS_HORAS.hora='h01' and
TIPOS_IMAGEN.tip_hr_hora=TIPOS_HORAS.hora and
TIPOS_IMAGENES_RESPALDOS.t_imagen_id_tip_imag=TIPOS_IMAGEN.id_tip_imag
and TIPOS_IMAGENES_RESPALDOS.resp_id_respaldo=RESPALDOS.id_respaldo;

ID_TIP_IMA
----------
96-ID-0001
```

5. ¿ De qué satélite proviene el recorte de región 0001-1996?.

```
SQL>select SATELITES.id_satelite from SATELITES, INFORMACIONES_CRUDAS,
RECORTES_DE_REGIONES where id_recorte='001-1996' and
RECORTES_DE_REGIONES.infcruda_timagen=
INFORMACIONES_CRUDAS.id_inf_cruda and
INFORMACIONES_CRUDAS.sat_id_satelite=SATELITES.id_satelite;

no rows selected
```

6.¿En que acervo se encuentra la fotografía con ID 2000?

```
SQL>select ACERVOS_FOTOGRAFICOS.id_acervo from ACERVOS_FOTOGRAFICOS,
FOTOGRAFIAS,REGISTROS where id_foto='2000' and
REGISTROS.acerv_foto_id_acervo=ACERVOS_FOTOGRAFICOS.id_acervo and
REGISTROS.fotog_id_foto=FOTOGRAFIAS.id_foto;

no rows selected
```

7. ¿Cuál es el nombre del tipo de imagen que se utiliza para la imagen con ID 96-IM-0001, así como el estado en que se encuentra la bitácora que alberga dicha imagen?

```
SQL>select TIPOS_IMAGEN.nombre, BITACORAS_DE_MES_EN_ESTUDIOS.comentario
from TIPOS_IMAGEN,BITACORAS_DE_MES_EN_ESTUDIOS,IMAGENES_BITACORAS where
TIPOS_IMAGEN.id_tip_imag='96-IM-0001' and
TIPOS_IMAGEN.id_tip_imag=IMAGENES_bitacoras.t_imagen_id_tip_imag and
imagenes_bitacoras.bit_MESTUD_ID_BITACORA=BITACORAS_DE_MES_EN_ESTUDIOS.id
_bitacora;

NOMBRE   COMENTARIO
-------      -------------------------
Mensual      Incompleta
```

Capítulo 6

CONCLUSIONES

Como primer paso en el desarrollo del presente trabajo se realizó el estudio de los modelos de datos, donde se eligió el modelo Entidad/Relación debido a que es un modelo simple, potente y formal para representar la realidad. También ofrecía una base robusta para enfocar y analizar los problemas relacionados con la manipulación de bases de datos, como el diseño de la base de datos, la redundancia, y la distribución de la información.

En lo que respecta a la metodología elegida, la cual fue CASE porque proporcionó un entorno de desarrollo interactivo con un tiempo de respuesta rápido, recursos dedicados y una comprobación de errores desde el principio.

Llevando a la práctica la metodología CASE, en la etapa de estrategia se estableció como sistema manejador de base de datos a ORACLE en su versión 8i. Es un software que basa su desarrollo en CASE. Como resultado de la etapa de análisis se obtuvieron una serie de diagramas que permitieron una completa comprensión para el desarrollo del sistema. Como lo son: el diagrama de procesos, el diagrama jerárquico, el diagrama Entidad/Relación, etc.

El diagrama Entidad/Relación fue una pieza muy importante para el desarrollo de la base de datos ya que en este diagrama quedan plasmados todos los requerimientos de la base de imágenes satelitales, así como las restricciones que llevaría. La forma para verificar el diagrama fue aplicando las formas normales de Codd. A partir de este diagrama se generaron las entidades y sus relaciones, así como las integridades referenciales de la mismas. Con esto se tiene la estructura del diccionario de datos y los objetos que se crearon. Fue importante analizar todos estos aspectos para poder separar la información que tendrá mucha demanda, para lograr un mejor rendimiento (tiempo de respuesta) de la base de datos.

A partir del diagrama Entidad/Relación se obtuvo el diseño de la base de datos. El cual hace una simulación de cómo queda insertada la información en la base de datos y de cómo será mostrada la información en las consultas que realicen los usuarios finales.

Posteriormente en la etapa de construcción se mapearon las entidades y sus atributos del diagrama Entidad/Relación a tablas y los tipos de datos para las columnas, mediante el lenguaje de consulta estructurado SQL. Enseguida se crearon las reglas de integridad referencial con las llaves primarias y foráneas en las respectivas relaciones entre tablas. Cabe mencionar que a partir de aquí, los demás elementos que se construyeron fueron con programación y con la ayuda de la herramienta SQL*PLUS.

Finalmente, se hicieron las pruebas de inserción y consultas de datos, principalmente de las imágenes satelitales. En donde se comprobó que la información quedó clasificada según los requerimientos así como automatizada y de fácil consulta para los usuarios finales. Por lo tanto el sistema cumple con todos los requisitos preestablecidos.

En cuanto a trabajos a futuro en el sistema desarrollado se pueden contemplar los siguientes:

- Migraciones a futuras versiones de Oracle para tener actualizado el sistema.
- Posibles modificaciones al sistema, por las demandas de los clientes.
- Creación de tablas e índices particionados.
- Integración de nuevos módulos al sistema.
- Creación de un "*Data Warehouse*" por el aumento masivo de información a través del tiempo.

BIBLIOGRAFÍA

[Alonso, 1998] Esaú Alonso Elizo. **"Manual Avanzado de Oracle8".** ANAYA Multimedia, Madrid, 1998.

[Barker, 1992] R. Barker, C. Logman. **"Elementos y Herramientas en Desarrollo de Sistemas de Información, (Una Visión Actual de la Tecnología CASE) ".** Adison Wesley Iberoamericana, México, 1995.

[Begoña, 1989] Miren Begoña, Albizuri Romero. **"Estructuras de Datos e Introducción a Bases de Datos".** Limusa, México, 1989.

[Bobrowski,2001] Steve Bobrowski. **"Oracle8i para Linux Edición de Aprendizaje".** Osborne/McGraw-Hill, España, 2001.

[Brodie, 1984] Brodie,M.L., J.Mylopoulos, J.W. Schmidt. **"On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases, and Programming Languages".** New York: Springen-Verlag, 1984.

[De Miguel, 1999] Adoración de Miguel Castaño, Mario G. Piattini. **"Fundamentos y Modelos de Bases de Datos".** Alfaomega, México, 1999.

[De Miguel, 1993] Adoración de Miguel Castaño., Mario Gerardo Piattini Velthuis. **"Concepción y Diseño de Bases de Datos, del Modelo E/R al Modelo Relacional".** Adison Wesley Iberoamericana, E.U.A., 1993.

[Gardarin, 1994] Georges Gardarin. **"Dominar las Bases de Datos, Modelos y Lenguajes".** Ediciones Gestión 2000, Barcelona, 1994.

[Guerrero, 2001] Luis Guerrero. **"Modelos y Herramientas en Tecnología de la Información".** Departamento de Ciencias de la Computación. Universidad de Chile, 2001. http://www.dcc.uchile.cl/~luguerre/cc20a/introdb.html

[Hansen, 1997] Gary W. Hansen, James V. Hansen. **"Diseño y Administración de Bases de Datos".** Prentice-Hall, España, 1997.

[Hawryszkiewycz, 1994] I.T. Hawryszkiewycz. **"Análisis y Diseño de Bases de Datos".** Noriega Editores, México, 1994.

[Hernández, 2001] Tony Hernández. **"Informática Documental. Curso 2000/2001".** Universidad Carlos III de Madrid. http://www.bib.uc3m.es/~tony/informática/infcatema6.htm

[Koch, 1994] George Koch. **"Oracle7 Manual de Referencia".** Osborne/McGraw-Hill, España, 1994.

[Lascano, 2000] Edison Lascano. **"Systems Engineer at Escuela Politécnica Nacional".** Ecuador, 2000. http://iwia.sis.epn.edu.ec/~elascano/desarrollo/oo/modelofuncional.html

[López, 1991] Antonio López Fuensalida. **"Metodologías de Desarrollo, Producción Automática de Software con Herramientas CASE"**. Macrobit: Ra-Ma, México, 1991.

[Moreno, 2000] Antonio Moreno Ortiz. **"Diseño e Implementación de un Lexicón Computacional, para Lexicografía y Traducción Automática"**. Volumen 9 (2000) . Facultad de Filosofía y Letras Universidad de Málaga, España http://elies.rediris.es/elies9/4-2.htm

[Parrilla, 1997] Juan Carlos Parrilla, Juan José Rubio. **"Sistemas Gestores de Bases de Datos"**. Síntesis, España, 1997.

[Pereda, 2001] Manuel Pereda Domínguez. **"Bases de Datos (2º Diplomatura de Estadística) Curso 2001-2002 "**. Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad de Sevilla, España, 2002. http://www.cs.us.es/cursos/bd-2001/temas/diseno.html

[Piattini, 1995] Mario G. Piattini, Sunil N. Daryanani. **"Elementos y Herramientas en el Desarrollo de Sistemas de Información"**. Ra-Ma, Madrid, 1995.

[Piattini, 1992] Mario G. Piattini, Sunil N. Daryanani. **"Elementos y Herramientas en el Desarrollo de Sistemas de Información, (Una Visión Actual de la Tecnología CASE) "**. Adison Wesley Iberoamericana, .E.U.A., 1992.

[Pressman, 1998] Roger S. Pressman. **"Ingeniería de Software, Un Enfoque Práctico"**. Mc Graw-Hill, España, 1998.

[Rodríguez, 2000] Olga Rodríguez. **"Ambientación y Definición de Términos"**. Colombia, 2000 http://atenea.udistrital.edu.co/estudiantes/olmerol/algunos.html

[Rosenthal, 1989] A. Rosenthal, J.A.Pino, **"A Generalized Algorithm for Centrality Problems on Trees"**. Journal of the ACM, 36, No.2, Abril 1989.

[Tsai, 1990] Alice Y.H. Tsai. **"Sistemas de Bases de Datos, Administración y Uso"**. Prentice-Hall, México, 1990.

[Tsichritzis, 1982] Dionysios Tsichritzis. **"Data Models"**. Prentice-Hall, U.S.A., 1982.

ANEXOS

ANEXO

Diagrama de Procesos (1a Parte)

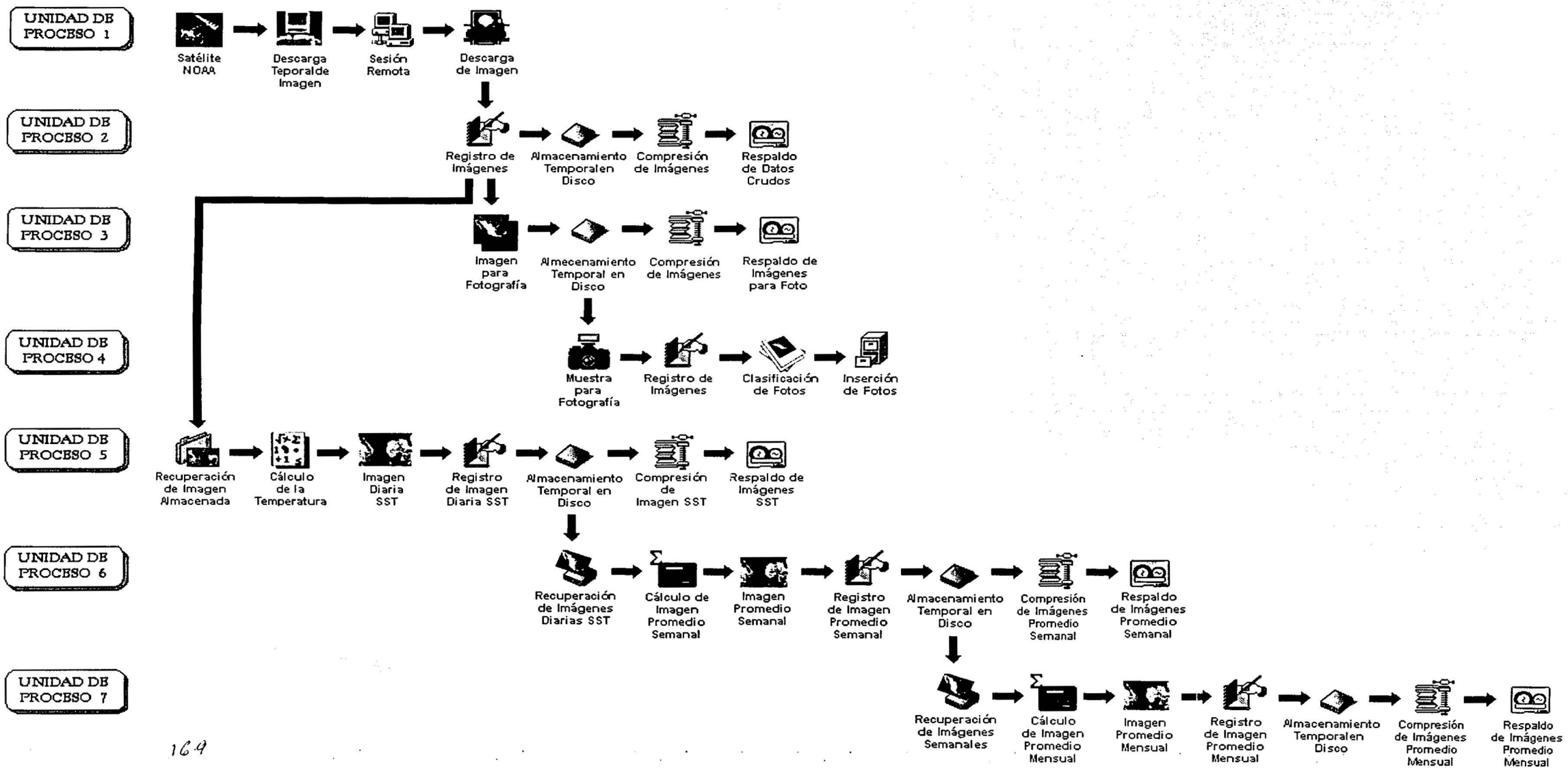
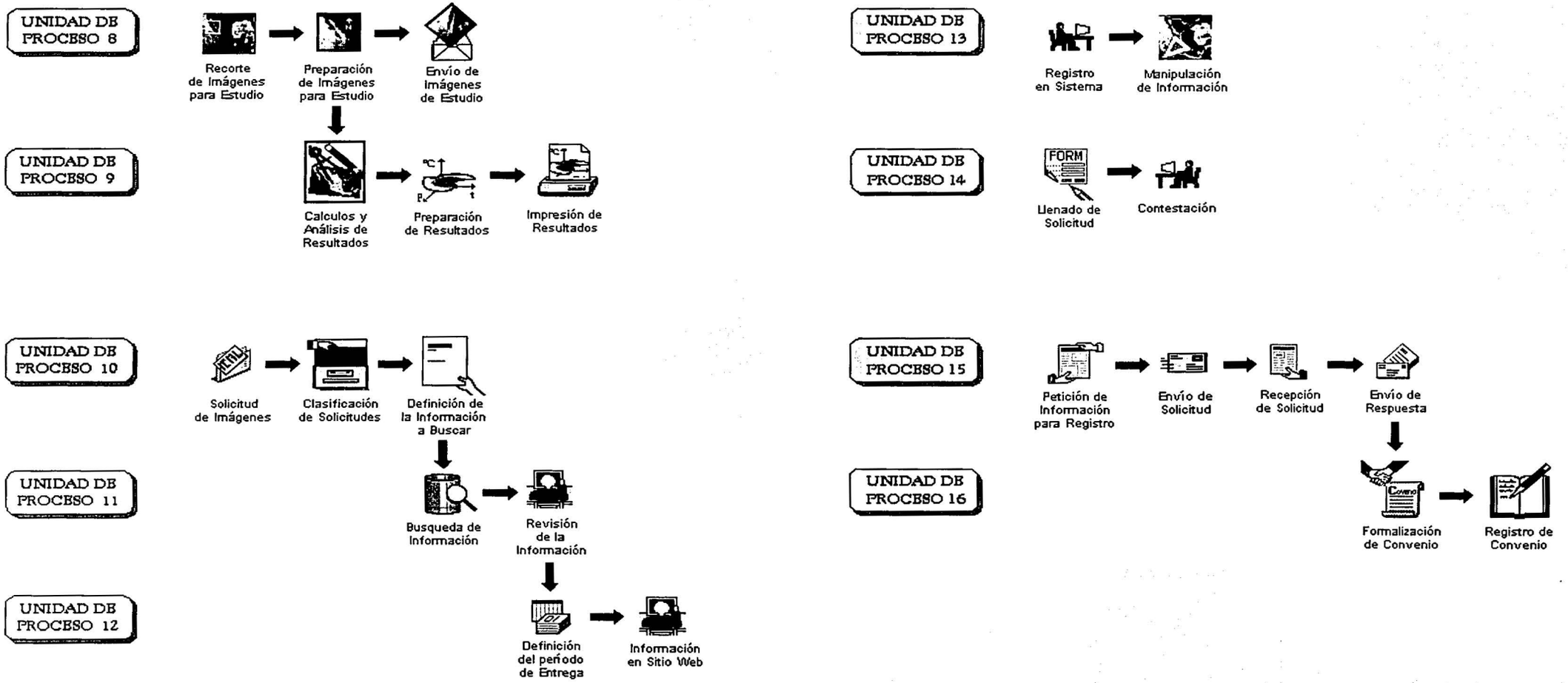


Diagrama de Procesos (2a Parte)



ANEXO

ANEXO

B

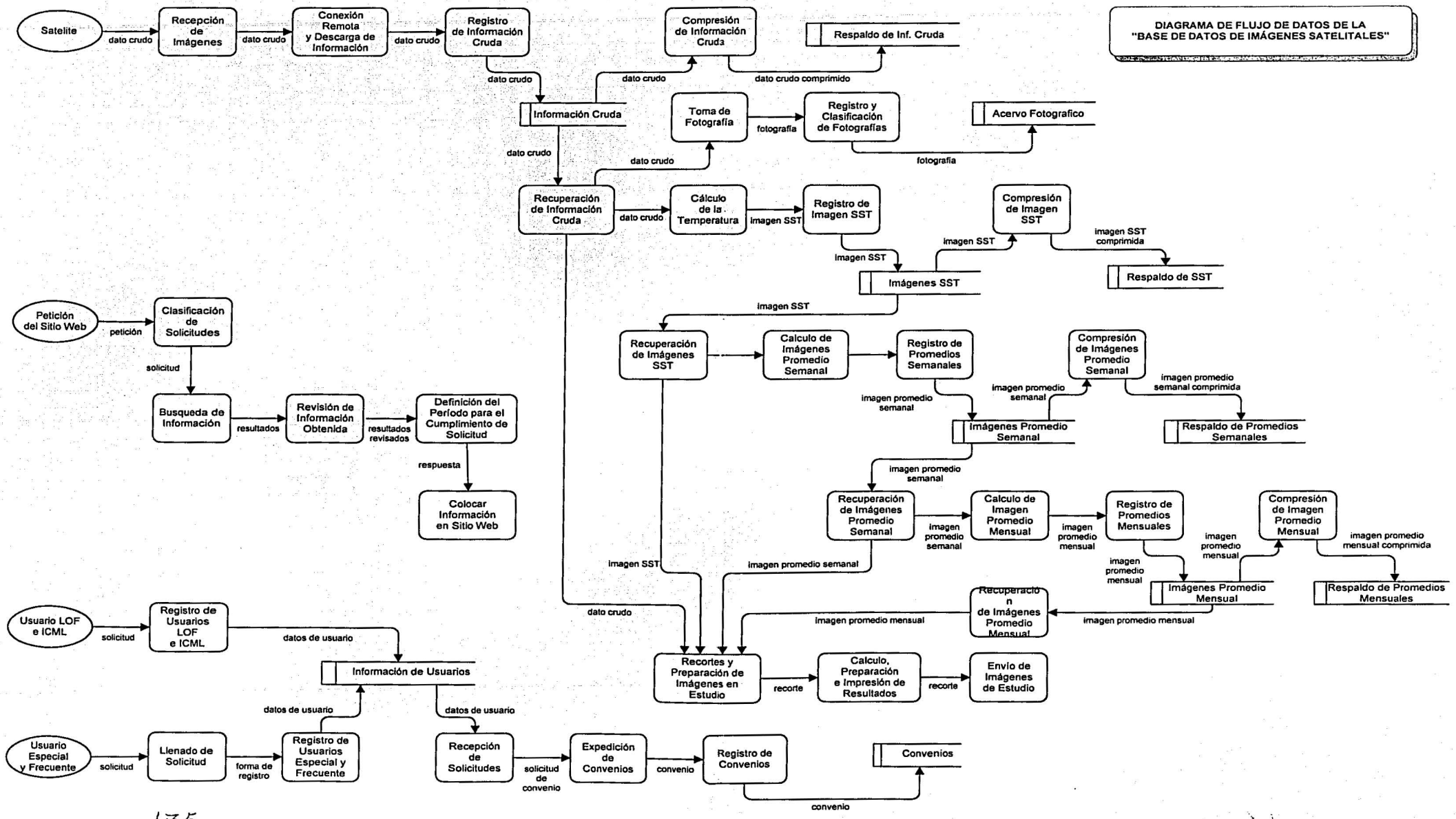
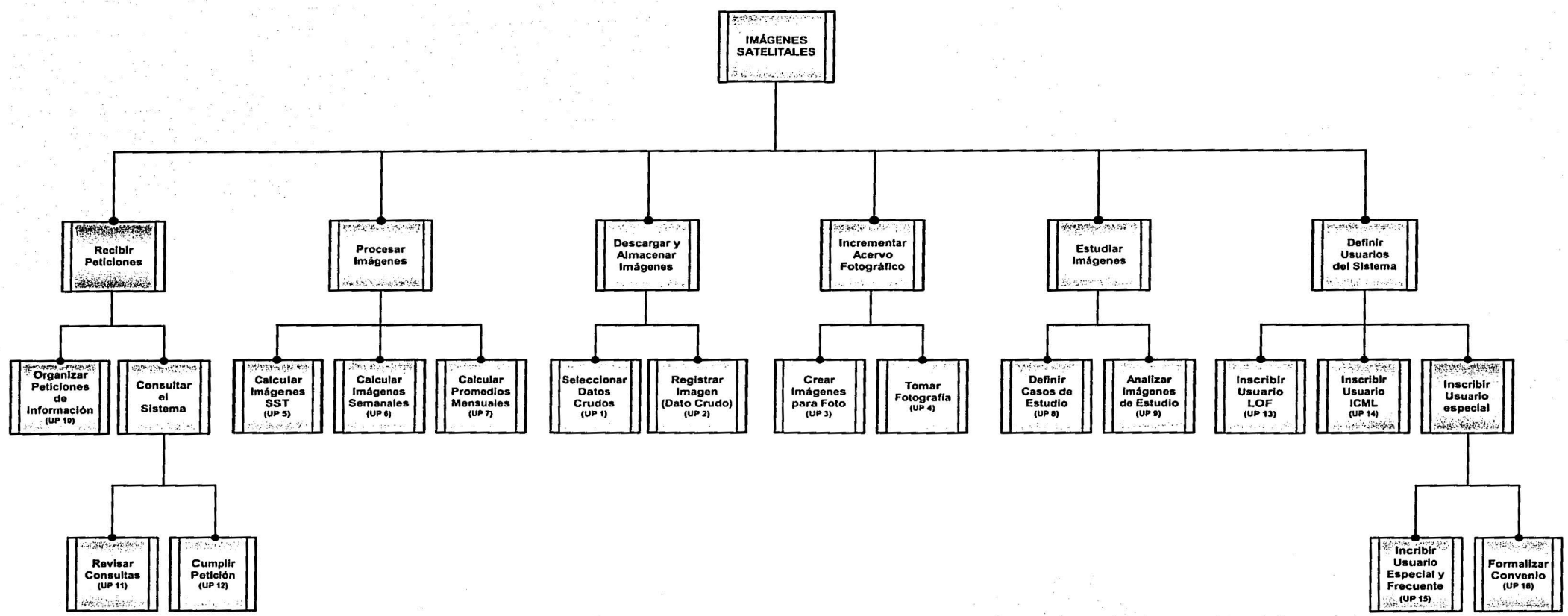


DIAGRAMA DE FLUJO DE DATOS DE LA "BASE DE DATOS DE IMÁGENES SATELITALES"
Satelite
dato crudo
Recepción de Imágenes
dato crudo
Conexión Remota y Descarga de Información
dato crudo
Registro de Información Cruda
dato crudo
Información Cruda
dato crudo
Compresión de Información Cruda
dato crudo comprimido
Respaldo de Inf. Cruda
dato crudo
Toma de Fotografía
fotografía
Registro y Clasificación de Fotografías
fotografía
Acervo Fotografico
dato crudo
Recuperación de Información Cruda
dato crudo
Cálculo de la Temperatura
imagen SST
Registro de Imagen SST
Imagen SST
Imágenes SST
imagen SST
Compresión de Imagen SST
imagen SST comprimida
Respaldo de SST
Imagen SST
Recuperación de Imágenes SST
Calculo de Imágenes Promedio Semanal
Registro de Promedios Semanales
imagen promedio semanal
Imágenes Promedio Semanal
imagen promedio semanal
Compresión de Imágenes Promedio Semanal
imagen promedio semanal comprimida
Respaldo de Promedios Semanales
imagen promedio semanal
Recuperación de Imágenes Promedio Semanal
imagen promedio semanal
Calculo de Imagen Promedio Mensual
imagen promedio mensual
Registro de Promedios Mensuales
imagen promedio mensual
Imágenes Promedio Mensual
imagen promedio mensual
Compresión de Imagen Promedio Mensual
imagen promedio mensual comprimida
Respaldo de Promedios Mensuales
imagen promedio mensual
Recuperación de Imágenes Promedio Mensual
Imagen SST
imagen promedio semanal
dato crudo
Recortes y Preparación de Imágenes en Estudio
recorte
Calculo, Preparación e Impresión de Resultados
recorte
Envio de Imágenes de Estudio
Petición del Sitio Web
petición
Clasificación de Solicitudes
solicitud
Busqueda de Información
resultados
Revisión de Información Obtenida
resultados revisados
Definición del Período para el Cumplimiento de Solicitud
respuesta
Colocar Información en Sitio Web
Usuario LOF e ICML
solicitud
Registro de Usuarios LOF e ICML
datos de usuario
Información de Usuarios
datos de usuario
Usuario Especial y Frecuente
solicitud
Llenado de Solicitud
forma de registro
Registro de Usuarios Especial y Frecuente
datos de usuario
Recepción de Solicitudes
solicitud de convenio
Expedición de Convenios
convenio
Registro de Convenios
convenio
Convenios

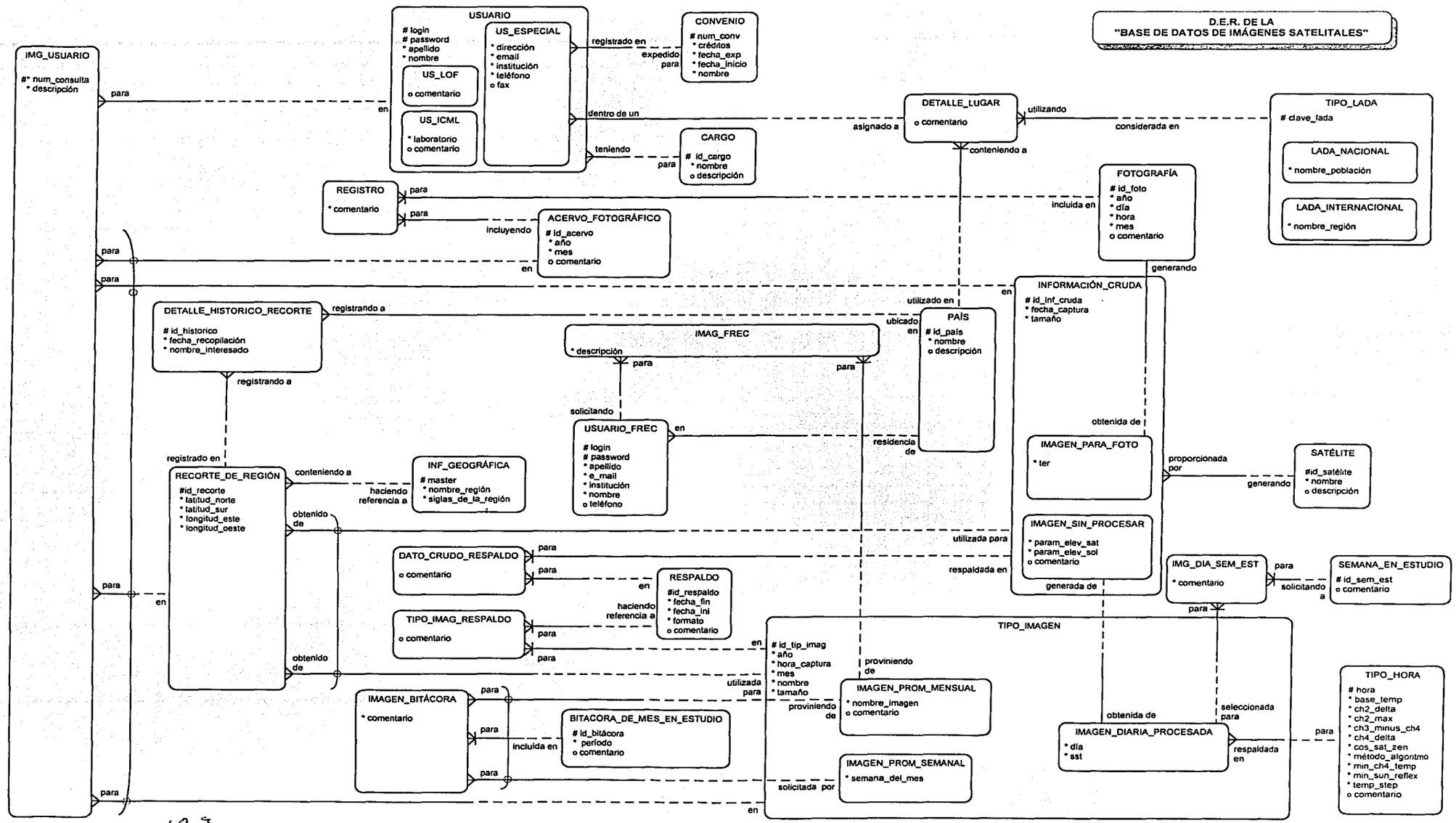
ANEXO

DIAGRAMA JERÁRQUICO DE LA
"BASE DE DATOS DE IMÁGENES SATELITALES"



* UP = Unidad de Proceso

ANEXO


D.E.R. DE LA
"BASE DE DATOS DE IMÁGENES SATELITALES"
IMG_USUARIO
#* num_consulta
* descripción
USUARIO
login
password
* apellido
* nombre
US_ESPECIAL
* dirección
* email
* institución
* teléfono
o fax
US_LOF
o comentario
US_ICML
* laboratorio
o comentario
CONVENIO
num_conv
* créditos
* fecha_exp
* fecha_inicio
* nombre
CARGO
id_cargo
* nombre
o descripción
DETALLE_LUGAR
o comentario
TIPO_LADA
clave_lada
LADA_NACIONAL
* nombre_población
LADA_INTERNACIONAL
* nombre_región
REGISTRO
* comentario
ACERVO_FOTOGRÁFICO
id_acervo
* año
* mes
o comentario
FOTOGRAFÍA
id_foto
* año
* día
* hora
* mes
o comentario
INFORMACIÓN_CRUDA
id_inf_cruda
* fecha_captura
* tamaño
DETALLE_HISTORICO_RECORTE
id_historico
* fecha_recopilación
* nombre_interesado
IMAG_FREC
* descripción
PAÍS
id_país
* nombre
o descripción
USUARIO_FREC
login
password
* apellido
* e_mail
* institución
* nombre
o teléfono
IMAGEN_PARA_FOTO
* ter
SATÉLITE
#id_satélite
* nombre
o descripción
RECORTE_DE_REGIÓN
#id_recorte
* latitud_norte
* latitud_sur
* longitud_este
* longitud_oeste
INF_GEOGRÁFICA
master
* nombre_región
* siglas_de_la_región
IMAGEN_SIN_PROCESAR
* param_elev_sat
* param_elev_sol
o comentario
DATO_CRUDO_RESPALDO
o comentario
RESPALDO
#id_respaldo
* fecha_fin
* fecha_ini
* formato
o comentario
TIPO_IMAG_RESPALDO
o comentario
IMG_DIA_SEM_EST
* comentario
SEMANA_EN_ESTUDIO
id_sem_est
o comentario
TIPO_IMAGEN
id_tip_imag
* año
* hora_captura
* mes
* nombre
* tamaño
IMAGEN_PROM_MENSUAL
* nombre_imagen
o comentario
IMAGEN_BITÁCORA
* comentario
BITACORA_DE_MES_EN_ESTUDIO
id_bitácora
* período
o comentario
IMAGEN_PROM_SEMANAL
* semana_del_mes
IMAGEN_DIARIA_PROCESADA
* día
* sst
TIPO_HORA
hora
* base_temp
* ch2_delta
* ch2_max
* ch3_minus_ch4
* ch4_delta
* cos_sat_zen
* método_algoritmo
* min_ch4_temp
* min_sun_reflex
* temp_step
o comentario
registrado en
expedido para
dentro de un
asignado a
utilizando
considerada en
teniendo
para
conteniendo a
para
incluida en
generando
para
incluyendo
en
para
en
registrando a
utilizado en
ubicado en
registrando a
para
para
solicitando
en
residencia de
obtenida de
registrado en
conteniendo a
haciendo referencia a
proporcionada por
generando
obtenido de
utilizada para
para
para
en
respaldada en
generada de
para
solicitando a
para
haciendo referencia a
para
para
en
obtenido de
utilizada para
proviniendo de
obtenida de
seleccionada para
respaldada en
para
proviniendo de
para
para
incluida en
para
solicitada por
en
para
183

ANEXO

T

Nombre de la tabla: **INFORMACIONES_CRUDAS**

Columna	id_inf_cruda	id_satélite	fecha_captura	tamaño	tipo_inf	param_elev_sat	param_elev_sol	ter
Tipo Llave	PK	FK						
Nulo/Único	NN, U		NN	NN	NN			
Ejemplos	96-SP-0001	N12	01-sep-1999	90789789	I_SPROC	10	20	
	96-IF-0002	N14	15-apr-2000	89567894	I_PFOTO			ter
	96-IF-0003	N12	20-may-2000	91234567	I_PFOTO			Ter

Nombre de la tabla: **TIPOS_IMAGEN**

Columna	id_tipo_imag	nombre	hora_captura	mes	año	tamaño	tipo_imagen	id_inf_cruda	id_tipo_hora	dia	sst	semana_del_mes	nombre_imagen	comentario
Tipo Llave								FK1	FK2					
Nulo/Único	NN	NN	NN	NN	NN	NN	NN							
Ejemplos	96-ID-0001	DIARIA	00:37	1	1996	80987567	1	96-SP-0001	H01	1	sst			
	96-IS-0001	SEMANAL	11:37	12	1999	81897568	2	96-SP-0001				1a		
	96-IM-0001	MENSUAL	09:11	10	1997	90897654	3						Mensual_Octubre_97	Buena Resolución

Nombre de la tabla: **USUARIOS (SUPERTABLA)**

Columna	login	password	id_cargo	nombre	apellido	tipo_usr	coment_lof	laboratorio	coment_icml
Tipo Llave	PK	PK	FK1			FK2			
Nulo/Único	NN, U	NN, U		NN	NN	NN			
Ejemplos	rmarqlof	0LF0mrq3	1	Erick	Marquez	U_LOF	Persona que trabaja imágenes		
	hreyesfi	0IC0rys1	2	Héctor Enrique	Reyes Delgado	U_ICM		LOF	Realiza tesis
	rjimenfi	0SP0jmn2	2	Irlanda	Jiménez Hernández	U_ESP			

Columna	id_país	clv_lada	num_conv	institucion	dirección	teléfono	email	fax
Tipo Llave	FK3	FK4	FK5					
Nulo/Único								
Ejemplos								
	001	55N	1	UNAM	Duraznos 576	56688722	irlanda@servidor.unam.mx	

Nombre de la tabla: **USUARIOS_FRECUENTES**

Columna	login_frec	password_frec	id_pais	nombre	apellido	institución	teléfono	email
Tipo Llave	PK	PK	FK					
Nulo/Unico	NN, U	NN, U		NN	NN	NN		NN
Ejemplos	tonyhtfr	0UF0prz1	1	Antonio	Pérez Sandoval	UNAM	56984231	tony@yahoo.com
	dianarrr	0UF0mrt2	1	Diana	Martínez Ruiz	UAM	57980934	dianita@altavista.com
	joselgon	0UF0gnz3	1	José Luis	González López	UNAM	58354322	josell@tecnoweb.com

Nombre de la tabla: **FOTOGRAFIAS**

Columna	id_foto	id_inf_cruda	día	hora	mes	año	comentario
Tipo Llave	PK	FK					
Nulo/Unico	NN, U		NN	NN	NN	NN	
Ejemplos	1	96-IF-0002	12	00:05	1	2000	Muy clara
	10	96-IF-0003	14	20:30	1	2000	Con nubosidad excesiva
	20	97-IF-0050	5	11:32	3	2001	Opaca

Nombre de la tabla: **RESPALDOS**

Columna	id_respaldo	formato	fecha_ini	fecha_fin	comentario
Tipo Llave	PK				
Nulo/Unico	NN, U	NN	NN	NN	
Ejemplos	024/1999	8mm	01/01/2000	20/01/2000	Un solo cassette
	032/2000	8mm	21/11/2000	10/11/2000	Un solo cassette
	048/2000	8 mm	05/02/2001	20/02/2001	Dos cassettes

Nombre de la tabla: **CONVENIOS**

Columna	num_conv	nombre	fecha_ini	fecha_exp	créditos
Tipo Llave	PK				
Nulo/Unico	NN, U	NN	NN	NN	NN
Ejemplos	1	UAM Xochimilco	1-feb-2000	12-mar-2000	Publicación de Fotos
	2	UVM Coapa	2-nov-2001	12-dec-2001	Donación de equipo
	3	Revista El Foco	15-apr-2001	30-may-2001	Mención en la revista

Nombre de la tabla: **DETALLES_HISTORICOS_RECORTES**

Columna	id_histórico	id_país	id_recorte	fecha_recop	nombre_interesado	apellido_interesado
Tipo Llave	PK	FK1	FK2			
Nulo/Unico	NN,U			NN	NN	NN
Ejemplos	1	1	002/1996	14/-sep-1998	Raúl	González
	48	1	020/1997	3-oct-1999	Marcos	Magaña
	59	1	010/1996	5-nov-1998	Agustín	Espinosa

Nombre de la tabla: **TIPOS_HORAS**

Columna	hora	método_algoritmo	cos_sat_zen	ch4_delta	ch2_max	ch2_delta
Tipo Llave	PK					
Nulo/Unico	NN, U	NN	NN	NN	NN	NN
Ejemplos	H01	MC	0.25	0.2	0.5	0.25
	H02	MC	0.4	1	2	0.25
	H03	MC	0.2	1	2	0.25

Columna	ch3_minus_ch4	base_temp	temp_step	min_ch4_temp	min_sun_reflex	comentario
Tipo Llave						
Nulo/Unico	NN	NN	NN	NN	NN	
Ejemplos	-1.5	10	0.1	0	0	A
	-1.5	10	0.1	0	0	B
	-1.5	10	0.1	0	0	C

Nombre de la tabla: **IMAGENES_USUARIOS (ARCO_GENERICO)**

Columna	num_consulta	usr_login	usr_passwd	id_tipo_inf	descripción
Tipo Llave	PK,FK1	FK2	FK2		
Nulo/Unico	NN,U1	NN	NN		
Ejemplos	1	rjimenh	0sp0jmn2	96-SP-0001	Trabajo de imágenes
	2	rmarqlof	0LF0mrq3	1	Trabajo de Imágenes
	3	hreyesfi	0IC0rys1	96-ID-0001	Trabajo de Imágenes

Nombre de la tabla: **IMAGENES_FRECUENTES**

Columna	login_frec	passwd_frec	id_tipo_imag	descripción
Tipo Llave	PK, FK1	PK, FK1	PK,FK2	
Nulo/Unico	NN, U1	NN, U1	NN, U2	NN
Ejemplos	tonyhtfr	0UF0prz1	96-IM-0001	Observo la mensual de enero
	dianarrr	0UF0mrt2	97-IM-0012	Observo solo las recientes
	joselgon	0UF0gnz3	99-IM-0003	Observo todas las mensuales

Nombre de la tabla: **TIPOS_LADAS**

Columna	clave_lada	lada_tipo	nombre_población	nombre_región
Tipo Llave	PK			
Nulo/Unico	NN,U	NN		
Ejemplos	726N	LN	Villa Victoria	
	833N	LN	Tampico, Tamps.	
	81+31	LI		Tokio

Nombre de la tabla: **ACERVOS_FOTOGRAFICOS**

Columna	id_acervo	mes	año	comentario
Tipo Llave	PK			
Nulo/Unico	NN, U	NN	NN	
Ejemplos	1	5	1996	Excelentes imágenes
	110	4	1998	Bueno
	204	12	2000	Completo

Nombre de la tabla: **IMAGENES_BITACORAS (ARCO GENÉRICO)**

Columna	id_bitácora	id_tipo_imagen	comentario
Tipo Llave	PK, FK1	FK2	
Nulo/Unico	NN, U1	NN	
Ejemplos	1	96-IM-0001	Mensual
	4	96-IM-0001	Mensual
	5	96-IS-0001	Semanal

Nombre de la tabla: **SATELITES**

Columna	id_satélite	nombre	descripción
Tipo Llave	PK		
Nulo/Único	NN,U	NN	
Ejemplos	n12	NOAA-12	Primer Satélite
	n02	NOAA-14	Segundo Satélite
	n03	NOAA-16	Tercer Satélite

Nombre de la tabla: **PAISES**

Columna	id_país	nombre	descripción
Tipo Llave	PK		
Nulo/Único	NN, U	NN	
Ejemplos	1	México	América Central
	2	Egipto	
	3	Alemania	

Nombre de la tabla: **SEMANAS_EN_ESTUDIO**

Columna	id_sem_est	comentario
Tipo Llave	PK	
Nulo/Único	NN,U	
Ejemplos	001-96	Primera de enero
	002-96	Segunda de enero
	003-96	Tercera de enero

Nombre de la tabla: **INFORMACIONES_GEOGRAFICAS**

Columna	master	nombre_región	siglas_región
Tipo Llave	PK		
Nulo/Único	NN, U	NN	NN
Ejemplos	mast001	Tehuantepec	Thtp
	mast002	Manzanillo	Mnzn
	mast003	Tepeji	Tpji

Nombre de la tabla: **BITACORAS_DE_MES_EN_ESTUDIOS**

Columna	id-bitácora	periodo	comentario
Tipo Llave	PK		
Nulo/Único	NN, U	NN	
Ejemplos	1	Enero– Febrero	completa
	2	Febrero-Febrero	completa
	3	Febrero- Marzo	incompleta

Nombre de la tabla: **CARGOS**

Columna	id_cargo	nombre	descripción
Tipo Llave	PK		
Nulo/Único	NN, U	NN	
Ejemplos	1	Académico	Actividades en el LOF
	2	Tesista	Servicio social
	3	Laboratorista	Trabajador de la UNAM

Nombre de la tabla: **REGISTROS**

Columna	id_acervo	id_foto	comentario
Tipo Llave	PK, FK	PK, FK	
Nulo/Único	NN, U	NN, U	NN
Ejemplos	1	96-IF-0001	Completo
	1	96-IF-0002	Completo
	1	96-IF-0003	Incompleto

Nombre de la tabla: **DATOS_CRUDOS_RESPALDOS**

Columna	id_respaldo	id_inf_cruda	comentario
Tipo Llave	PK, FK	PK, FK	
Nulo/Único	NN, U	NN, U	
Ejemplos	002/2000	96-SP-0001	
	001/1996	96-IF-0001	
	003/2001	97-IF-0100	

Nombre de la tabla: **TIPOS_IMAGENES_RESPALDOS**

Columna	id_respaldo	id_tipo_imag	comentario
Tipo Llave	PK, FK	PK, FK	
Nulo/Único	NN, U	NN, U	
Ejemplos	003/2001	96-IM-0145	
	031/2000	97-IS-4589	
	010/1998	00-ID-8765	

Nombre de la tabla: **DETALLES_LUGARES**

Columna	id_pais	clave_lada	comentario
Tipo Llave	PK, FK2	PK, FK1	
Nulo/Único	NN, U2	NN, U1	
Ejemplos	001	726N	Poblado de México
	001	833N	Poblado de Tamaulipas
	004	81+3I	Japón

Nombre de la tabla: **IMAGENES_DIA_SEMANAS_ESTUDIO**

Columna	id_semana_estudio	id_tipo_imagen	comentario
Tipo Llave	PK, FK1	PK, FK1	
Nulo/Único	NN, U	NN, U	NN
Ejemplos	004-96	96-ID-0001	Semana con 14 imágenes
	260-97	96-ID-0002	Semana con 10 imágenes
	040-96	96-ID-0003	Semana con 10 imágenes

ANEXO

CREACIÓN DE TABLAS

```
CREATE TABLE USUARIOS_FRECUENTES (
LOGIN_FREC VARCHAR2(8) NOT NULL,
PASSWORD_FREC VARCHAR2(8) NOT NULL,
PAIS_ID_PAIS NUMBER(3) NOT NULL,
NOMBRE VARCHAR2(15) NOT NULL,
APELLIDO VARCHAR2(20) NOT NULL,
INSTITUCION VARCHAR2(20) NOT NULL,
TELEFONO VARCHAR2(10),
EMAIL VARCHAR2(40) NOT NULL
)
/
CREATE TABLE DETALLES_HISTORICOS_RECORTES (
ID_HISTORICO NUMBER(3) NOT NULL,
PAIS_ID_PAIS NUMBER(3) NOT NULL,
REC_REGION_ID_RECORTE VARCHAR(8) NOT NULL,
FECHA_RECOPILACION DATE NOT NULL,
NOMBRE_INTERESADO VARCHAR2(15) NOT NULL,
APELLIDO_INTERESADO VARCHAR2(20) NOT NULL
)
/
CREATE TABLE BITACORAS_DE_MES_EN_ESTUDIOS (
ID_BITACORA NUMBER(4) NOT NULL,
PERIODO VARCHAR2(16) NOT NULL,
COMENTARIO VARCHAR2(50)
)
/
CREATE TABLE IMAGENES_BITACORAS (
BIT_MESTUD_ID_BITACORA NUMBER(4) NOT NULL,
COMENTARIO VARCHAR2(50) NOT NULL,
T_IMAGEN_ID_TIP_IMAG VARCHAR2(10) NOT NULL
)
/
CREATE TABLE TIPOS_IMAGEN (
ID_TIP_IMAG VARCHAR2(10) NOT NULL,
NOMBRE VARCHAR2(7) NOT NULL,
HORA_CAPTURA VARCHAR2(5) NOT NULL,
MES NUMBER(2) NOT NULL,
ANIO NUMBER(4) NOT NULL,
TAMANIO NUMBER(9) NOT NULL,
T_IMAGEN NUMBER(2) NOT NULL,
INF_CRUD_ID_INF_CRUDA VARCHAR2(10),
TIP_HR_HORA VARCHAR2(3),
DIA NUMBER(2),
SST VARCHAR2(3),
SEM_DEL_MES VARCHAR2(2),
NOMBRE_IMAGEN BLOB,
COMENTARIO VARCHAR2(50)
)
/
```

ANEXO F

```
CREATE TABLE IMAGENES_USUARIOS (
NUM_CONSULTA NUMBER(4) NOT NULL,
USR_LOGIN VARCHAR2(8),
USR_PASSWORD VARCHAR2(8),
DESCRIPCION VARCHAR2(30) NOT NULL,
ID_TIPO_INF VARCHAR2(10) NOT NULL
)
/
CREATE TABLE USUARIOS (
LOGIN VARCHAR2(8) NOT NULL,
PASSWORD VARCHAR2(8) NOT NULL,
NOMBRE VARCHAR2(15) NOT NULL,
APELLIDO VARCHAR2(20) NOT NULL,
CARGO_ID_CARGO NUMBER(2) NOT NULL,
TIPO_USR VARCHAR(5) NOT NULL,
COMENTARIO VARCHAR2(50),
LABORATORIO VARCHAR2(30),
US_ICML_COMENTARIO VARCHAR2(50),
DET_LUGAR_PAIS_ID_PAIS NUMBER(3),
DET_LUGAR_TIP_LADA_CLAVE_LADA VARCHAR2(10),
CONV_NUM_CONV NUMBER(4),
INSTITUCION VARCHAR2(20),
DIRECCION VARCHAR2(30),
TELEFONO VARCHAR2(10),
EMAIL VARCHAR2(40),
FAX VARCHAR2(10)
)
/
CREATE TABLE RESPALDOS (
ID_RESPALDO VARCHAR2(8) NOT NULL,
FORMATO VARCHAR2(3) NOT NULL,
FECHA_INI DATE NOT NULL,
FECHA_FIN DATE NOT NULL,
COMENTARIO VARCHAR2(50)
)
/
CREATE TABLE TIPOS_HORAS (
HORA VARCHAR2(3) NOT NULL,
METODO_ALGORITMO VARCHAR2(4) NOT NULL,
COS_SAT_ZEN NUMBER(4,2) NOT NULL,
CH4_DELTA NUMBER(4,2) NOT NULL,
CH2_MAX NUMBER(4,2) NOT NULL,
CH2_DELTA NUMBER(4,2) NOT NULL,
CH3_MINUS_CH4 NUMBER(4,2) NOT NULL,
BASE_TEMP NUMBER(4,2) NOT NULL,
TEMP_STEP NUMBER(4,2) NOT NULL,
MIN_CH4_TEMP NUMBER(4,2) NOT NULL,
MIN_SUN_REFLEX NUMBER(4,2) NOT NULL,
COMENTARIO VARCHAR2(50)
)
/
CREATE TABLE CARGOS (
ID_CARGO NUMBER(2) NOT NULL,
NOMBRE VARCHAR2(15) NOT NULL,
DESCRIPCION VARCHAR2(30)
)
/
```

```
CREATE TABLE INFORMACIONES_CRUDAS (
ID_INF_CRUDA VARCHAR2(10) NOT NULL,
SAT_ID_SATELITE VARCHAR2(3) NOT NULL,
FECHA_CAPTURA DATE NOT NULL,
TAMANIO NUMBER(9) NOT NULL,
TIPO_INFORMACION VARCHAR2(7) NOT NULL,
PARAM_ELEV_SAT NUMBER(2),
PARAM_ELEV_SOL NUMBER(2),
COMENTARIO VARCHAR2(50),
TER VARCHAR2(3)
)
/
CREATE TABLE INFORMACIONES_GEOGRAFICAS (
MASTER VARCHAR2(7) NOT NULL,
NOMBRE_REGION VARCHAR2(30) NOT NULL,
SIGLAS_DE_REGION VARCHAR2(4) NOT NULL
)
/
CREATE TABLE RECORTES_DE_REGIONES (
ID_RECORTE VARCHAR2(8) NOT NULL,
INF_GEGRAF_MASTER VARCHAR2(7) NOT NULL,
LATITUD_NORTE NUMBER(6,3) NOT NULL,
LATITUD_SUR NUMBER(6,3) NOT NULL,
LONGITUD_ESTE NUMBER(6,3) NOT NULL,
LONGITUD_OESTE NUMBER(6,3) NOT NULL,
INFCRUDA_TIMAGEN VARCHAR2(10) NOT NULL
)
/
CREATE TABLE PAISES (
ID_PAIS NUMBER(3) NOT NULL,
NOMBRE VARCHAR2(20) NOT NULL,
DESCRIPCION VARCHAR2(30)
)
/
CREATE TABLE REGISTROS (
ACERV_FOTO_ID_ACERVO NUMBER(6) NOT NULL,
FOTOG_ID_FOTO NUMBER(6) NOT NULL,
COMENTARIO VARCHAR2(50) NOT NULL
)
/
CREATE TABLE TIPOS_LADAS (
CLAVE_LADA VARCHAR2(10) NOT NULL,
LADA_TIPO VARCHAR2(2) NOT NULL,
NOMBRE_POBLACION VARCHAR2(30),
NOMBRE_REGION VARCHAR2(30)
)
/
CREATE TABLE TIPOS_IMAGENES_RESPALDOS (
RESP_ID_RESPALDO VARCHAR2(8) NOT NULL,
T_IMAGEN_ID_TIP_IMAG VARCHAR2(10) NOT NULL,
COMENTARIO VARCHAR2(50)
)
/
CREATE TABLE SEMANAS_EN_ESTUDIO (
ID_SEM_EST VARCHAR2(6) NOT NULL,
COMENTARIO VARCHAR2(50)
)/
```

ANEXO F

```
CREATE TABLE ACERVOS_FOTOGRAFICOS (
ID_ACERVO NUMBER(6) NOT NULL,
MES NUMBER(2) NOT NULL,
ANIO NUMBER(4) NOT NULL,
COMENTARIO VARCHAR2(50)
)
/
CREATE TABLE DATOS_CRUDOS_RESPALDOS (
RESP_ID_RESPALDO VARCHAR2(8) NOT NULL,
INF_CRUD_ID_INF_CRUDA VARCHAR2(10) NOT NULL,
COMENTARIO VARCHAR2(50)
)
/
CREATE TABLE SATELITES (
ID_SATELITE VARCHAR2(3) NOT NULL,
NOMBRE VARCHAR2(7) NOT NULL,
DESCRIPCION VARCHAR2(30)
)
/
CREATE TABLE DETALLES_LUGAR (
PAIS_ID_PAIS NUMBER(3) NOT NULL,
TIP_LADA_CLAVE_LADA VARCHAR2(10) NOT NULL,
COMENTARIO VARCHAR2(50)
)
/
CREATE TABLE FOTOGRAFIAS (
ID_FOTO NUMBER(6) NOT NULL,
INF_CRUD_ID_INF_CRUDA VARCHAR2(10) NOT NULL,
DIA NUMBER(2) NOT NULL,
HORA VARCHAR2(5) NOT NULL,
MES NUMBER(2) NOT NULL,
ANIO NUMBER(4) NOT NULL,
COMENTARIO VARCHAR2(50)
)
/
CREATE TABLE CONVENIOS (
NUM_CONV NUMBER(4) NOT NULL,
NOMBRE VARCHAR2(15) NOT NULL,
FECHA_INI DATE NOT NULL,
FECHA_EXP DATE NOT NULL,
CREDITOS VARCHAR2(50) NOT NULL
)
/
CREATE TABLE IMAGENES_DIA_SEMANAS_ESTUDIO (
SEM_ESTUDI_ID_SEM_EST VARCHAR2(6) NOT NULL,
T_IMAGEN_ID_TIP_IMAG VARCHAR2(10) NOT NULL,
COMENTARIO VARCHAR2(50) NOT NULL
)
/
CREATE TABLE IMAGENES_FRECUENTES (
US_FREC_LOGIN_FREC VARCHAR2(8) NOT NULL,
US_FREC_PASSWORD_FREC VARCHAR2(8) NOT NULL,
T_IMAGEN_ID_TIP_IMAG VARCHAR2(10) NOT NULL,
DESCRIPCION VARCHAR2(30) NOT NULL
)
/
```

CREACIÓN DE LLAVES PRIMARIAS

```
ALTER TABLE USUARIOS_FRECUENTES
 ADD (CONSTRAINT US_FREC_PK PRIMARY KEY
  (LOGIN_FREC
  ,PASSWORD_FREC))
/
ALTER TABLE DETALLES_HISTORICOS_RECORTES
 ADD (CONSTRAINT DET_HRECOR_PK PRIMARY KEY
  (ID_HISTORICO))
/
ALTER TABLE IMAGENES_BITACORAS
 ADD (CONSTRAINT IMAG_BITAC_PK PRIMARY KEY
  (BIT_MESTUD_ID_BITACORA))
/
ALTER TABLE BITACORAS_DE_MES_EN_ESTUDIOS
 ADD (CONSTRAINT BIT_MESTUD_PK PRIMARY KEY
  (ID_BITACORA))
/
ALTER TABLE TIPOS_IMAGEN
 ADD (CONSTRAINT T_IMAGEN_PK PRIMARY KEY
  (ID_TIP_IMAG))
/
ALTER TABLE IMAGENES_USUARIOS
 ADD (CONSTRAINT IMAG_USR_PK PRIMARY KEY
 (NUM_CONSULTA))
/
ALTER TABLE USUARIOS
 ADD (CONSTRAINT USR_PK PRIMARY KEY
  (LOGIN
  ,PASSWORD))
/
ALTER TABLE RESPALDOS
 ADD (CONSTRAINT RESP_PK PRIMARY KEY
  (ID_RESPALDO))
/
ALTER TABLE TIPOS_HORAS
 ADD (CONSTRAINT TIP_HR_PK PRIMARY KEY
  (HORA))
/
ALTER TABLE INFORMACIONES_CRUDAS
 ADD (CONSTRAINT INF_CRUD_PK PRIMARY KEY
  (ID_INF_CRUDA))
/
ALTER TABLE INFORMACIONES_GEOGRAFICAS
 ADD (CONSTRAINT INF_GEGRAF_PK PRIMARY KEY
  (MASTER))
/
ALTER TABLE RECORTES_DE_REGIONES
 ADD (CONSTRAINT REC_REGION_PK PRIMARY KEY
  (ID_RECORTE))
/
```

```
ALTER TABLE CARGOS
 ADD (CONSTRAINT CARGO_PK PRIMARY KEY
  (ID_CARGO))
/
ALTER TABLE PAISES
 ADD (CONSTRAINT PAIS_PK PRIMARY KEY
  (ID_PAIS))
/
ALTER TABLE REGISTROS
 ADD (CONSTRAINT REG_PK PRIMARY KEY
  (ACERV_FOTO_ID_ACERVO
  ,FOTOG_ID_FOTO))
/
ALTER TABLE TIPOS_LADAS
 ADD (CONSTRAINT TIP_LADA_PK PRIMARY KEY
  (CLAVE_LADA))
/
ALTER TABLE TIPOS_IMAGENES_RESPALDOS
 ADD (CONSTRAINT TIP_IMRESP_PK PRIMARY KEY
  (RESP_ID_RESPALDO
  ,T_IMAGEN_ID_TIP_IMAG))
/
ALTER TABLE SATELITES
 ADD (CONSTRAINT SAT_PK PRIMARY KEY
  (ID_SATELITE))
/
ALTER TABLE SEMANAS_EN_ESTUDIO
 ADD (CONSTRAINT SEM_ESTUDI_PK PRIMARY KEY
  (ID_SEM_EST))
/
ALTER TABLE DATOS_CRUDOS_RESPALDOS
 ADD (CONSTRAINT DAT_CRESP_PK PRIMARY KEY
  (INF_CRUD_ID_INF_CRUDA
  ,RESP_ID_RESPALDO))
/
ALTER TABLE ACERVOS_FOTOGRAFICOS
 ADD (CONSTRAINT ACERV_FOTO_PK PRIMARY KEY
  (ID_ACERVO))
/
ALTER TABLE DETALLES_LUGAR
 ADD (CONSTRAINT DET_LUGAR_PK PRIMARY KEY
  (PAIS_ID_PAIS
  ,TIP_LADA_CLAVE_LADA))
/
ALTER TABLE CONVENIOS
 ADD (CONSTRAINT CONV_PK PRIMARY KEY
  (NUM_CONV))
/
ALTER TABLE FOTOGRAFIAS
 ADD (CONSTRAINT FOTOG_PK PRIMARY KEY
  (ID_FOTO))
/
ALTER TABLE IMAGENES_DIA_SEMANAS_ESTUDIO
 ADD (CONSTRAINT I_DIA_SEM_PK PRIMARY KEY
  (T_IMAGEN_ID_TIP_IMAG
  ,SEM_ESTUDI_ID_SEM_EST))
/
```

```
ALTER TABLE IMAGENES_FRECUENTES
 ADD (CONSTRAINT IM_FREC_PK PRIMARY KEY
   (US_FREC_LOGIN_FREC
   ,US_FREC_PASSWORD_FREC
   ,T_IMAGEN_ID_TIP_IMAG))
/
```

CREACIÓN DE LLAVES FORANEAS

```
ALTER TABLE USUARIOS_FRECUENTES ADD (CONSTRAINT
 US_FREC_PAIS_FK FOREIGN KEY
   (PAIS_ID_PAIS) REFERENCES PAISES
   (ID_PAIS))
/
ALTER TABLE DETALLES_HISTORICOS_RECORTES ADD (CONSTRAINT
 DET_HRECOR_REC_REGION_FK FOREIGN KEY
   (REC_REGION_ID_RECORTE) REFERENCES RECORTES_DE_REGIONES
   (ID_RECORTE))
/
ALTER TABLE DETALLES_HISTORICOS_RECORTES ADD (CONSTRAINT
 DET_HRECOR_PAIS_FK FOREIGN KEY
   (PAIS_ID_PAIS) REFERENCES PAISES
   (ID_PAIS))
/
ALTER TABLE IMAGENES_BITACORAS ADD (CONSTRAINT
 IMAG_BITAC_BIT_MESTUD_FK FOREIGN KEY
   (BIT_MESTUD_ID_BITACORA) REFERENCES BITACORAS_DE_MES_EN_ESTUDIOS
   (ID_BITACORA))
/
ALTER TABLE TIPOS_IMAGEN ADD (CONSTRAINT
 T_IMAGEN_TIP_HR_FK FOREIGN KEY
   (TIP_HR_HORA) REFERENCES TIPOS_HORAS
   (HORA))
/
ALTER TABLE TIPOS_IMAGEN ADD (CONSTRAINT
 T_IMAGEN_INF_CRUD_FK FOREIGN KEY
   (INF_CRUD_ID_INF_CRUDA) REFERENCES INFORMACIONES_CRUDAS
   (ID_INF_CRUDA))
/
ALTER TABLE IMAGENES_USUARIOS ADD (CONSTRAINT
 IMAG_USR_USR_FK FOREIGN KEY
   (USR_LOGIN
   ,USR_PASSWORD) REFERENCES USUARIOS
   (LOGIN
   ,PASSWORD))
/
ALTER TABLE USUARIOS ADD (CONSTRAINT
 USR_CONV_FK FOREIGN KEY
   (CONV_NUM_CONV) REFERENCES CONVENIOS
   (NUM_CONV))
/
```

```
ALTER TABLE USUARIOS ADD (CONSTRAINT
 USR_DET_LUGAR_FK FOREIGN KEY
  (DET_LUGAR_PAIS_ID_PAIS
  ,DET_LUGAR_TIP_LADA_CLAVE_LADA) REFERENCES DETALLES_LUGAR
  (PAIS_ID_PAIS
  ,TIP_LADA_CLAVE_LADA))
/
ALTER TABLE USUARIOS ADD (CONSTRAINT
 USR_CARGO_FK FOREIGN KEY
  (CARGO_ID_CARGO) REFERENCES CARGOS
  (ID_CARGO))
/
ALTER TABLE INFORMACIONES_CRUDAS ADD (CONSTRAINT
 INF_CRUD_SAT_FK FOREIGN KEY
  (SAT_ID_SATELITE) REFERENCES SATELITES
  (ID_SATELITE))
/
ALTER TABLE RECORTES_DE_REGIONES ADD (CONSTRAINT
 REC_REGION_INF_GEGRAF_FK FOREIGN KEY
  (INF_GEGRAF_MASTER) REFERENCES INFORMACIONES_GEOGRAFICAS
  (MASTER))
/
ALTER TABLE REGISTROS ADD (CONSTRAINT
 REG_FOTOG_FK FOREIGN KEY
  (FOTOG_ID_FOTO) REFERENCES FOTOGRAFIAS
  (ID_FOTO))
/
ALTER TABLE REGISTROS ADD (CONSTRAINT
 REG_ACERV_FOTO_FK FOREIGN KEY
  (ACERV_FOTO_ID_ACERVO) REFERENCES ACERVOS_FOTOGRAFICOS
  (ID_ACERVO))
/
ALTER TABLE TIPOS_IMAGENES_RESPALDOS ADD (CONSTRAINT
 TIP_IMRESP_RESP_FK FOREIGN KEY
  (RESP_ID_RESPALDO) REFERENCES RESPALDOS
  (ID_RESPALDO))
/
ALTER TABLE TIPOS_IMAGENES_RESPALDOS ADD (CONSTRAINT
 TIP_IMRESP_T_IMAGEN_FK FOREIGN KEY
  (T_IMAGEN_ID_TIP_IMAG) REFERENCES TIPOS_IMAGEN (ID_TIP_IMAG))
/
ALTER TABLE DATOS_CRUDOS_RESPALDOS ADD (CONSTRAINT
 DAT_CRESP_RESP_FK FOREIGN KEY
  (RESP_ID_RESPALDO) REFERENCES RESPALDOS
  (ID_RESPALDO))
/
ALTER TABLE DATOS_CRUDOS_RESPALDOS ADD (CONSTRAINT
 DAT_CRESP_INF_CRUD_FK FOREIGN KEY
  (INF_CRUD_ID_INF_CRUDA) REFERENCES INFORMACIONES_CRUDAS
  (ID_INF_CRUDA))
/
ALTER TABLE DETALLES_LUGAR ADD (CONSTRAINT
 DET_LUGAR_TIP_LADA_FK FOREIGN KEY
  (TIP_LADA_CLAVE_LADA) REFERENCES TIPOS_LADAS
  (CLAVE_LADA))
/
```

```
ALTER TABLE DETALLES_LUGAR ADD (CONSTRAINT
 DET_LUGAR_PAIS_FK FOREIGN KEY
  (PAIS_ID_PAIS) REFERENCES PAISES
  (ID_PAIS))
/
ALTER TABLE FOTOGRAFIAS ADD (CONSTRAINT
 FOTOG_INF_CRUD_FK FOREIGN KEY
  (INF_CRUD_ID_INF_CRUDA) REFERENCES INFORMACIONES_CRUDAS
  (ID_INF_CRUDA))
/
ALTER TABLE IMAGENES_DIA_SEMANAS_ESTUDIO ADD (CONSTRAINT
 I_DIA_SEM_SEM_ESTUDI_FK FOREIGN KEY
  (SEM_ESTUDI_ID_SEM_EST) REFERENCES SEMANAS_EN_ESTUDIO
  (ID_SEM_EST))
/
ALTER TABLE IMAGENES_DIA_SEMANAS_ESTUDIO ADD (CONSTRAINT
 I_DIA_SEM_T_IMAGEN_FK FOREIGN KEY
  (T_IMAGEN_ID_TIP_IMAG) REFERENCES TIPOS_IMAGEN
  (ID_TIP_IMAG))
/
ALTER TABLE IMAGENES_FRECUENTES ADD (CONSTRAINT
 IM_FREC_US_FREC_FK FOREIGN KEY
  (US_FREC_LOGIN_FREC
  ,US_FREC_PASSWORD_FREC) REFERENCES USUARIOS_FRECUENTES
  (LOGIN_FREC
  ,PASSWORD_FREC))
/
ALTER TABLE IMAGENES_FRECUENTES ADD (CONSTRAINT
 IM_FREC_T_IMAGEN_FK FOREIGN KEY
  (T_IMAGEN_ID_TIP_IMAG) REFERENCES TIPOS_IMAGEN
  (ID_TIP_IMAG))
/
```

CREACIÓN DE TRIGGERS

```
                --ID DE BITACORAS DE MES EN ESTUDIO

CREATE OR REPLACE TRIGGER BITACORA_D_MES_EN_ESTUDIO
BEFORE INSERT ON BITACORAS_DE_MES_EN_ESTUDIOS
FOR EACH ROW
DECLARE
      TEMPORAL INTEGER;
BEGIN
      TEMPORAL:=LENGTH(:new.ID_BITACORA); --FORMATO: ####
      IF TEMPORAL>4 THEN
            :new.ID_BITACORA:=NULL;
      END IF;
END BITACORA_D_MES_EN_ESTUDIO;
/
```

```
                        --ID DE RESPALDOS

CREATE OR REPLACE TRIGGER RESPALDOS
BEFORE INSERT ON RESPALDOS
FOR EACH ROW
DECLARE
      TEMP1 INTEGER;
      TEMP2 INTEGER;
BEGIN
      TEMP1:=INSTR(:new.ID_RESPALDO,'/',1,1);   --FORMATO: ###/####
      TEMP2:=LENGTH(:new.ID_RESPALDO);

      IF TEMP1!=4 AND TEMP2>8 THEN
            :new.ID_RESPALDO:=NULL;
      END IF;
END RESPALDOS;
/

                  --ID DE DETALLES HISTORICOS RECORTES

CREATE OR REPLACE TRIGGER DETALLE_H_RECORT
BEFORE INSERT ON DETALLES_HISTORICOS_RECORTES
FOR EACH ROW
DECLARE
      TEMPORAL INTEGER;
BEGIN
      TEMPORAL:=LENGTH(:new.ID_HISTORICO);      --FORMATO: ###
      IF TEMPORAL>3 THEN
            :new.ID_HISTORICO:=NULL;
      ELSE
      :new.NOMBRE_INTERESADO:=INITCAP(LOWER(:new.NOMBRE_INTERESADO));
      :new.APELLIDO_INTERESADO:=INITCAP(LOWER(:new.APELLIDO_INTERESADO));
      END IF;
END DETALLES_H_RECORT;
/

                  --ID DE INFORMACIONES GEOGRAFICAS

CREATE OR REPLACE TRIGGER INF_GEOGRAF
BEFORE INSERT ON INFORMACIONES_GEOGRAFICAS
FOR EACH ROW
DECLARE
      TEMPORAL INTEGER;
      TEMPORAL1 INTEGER;
BEGIN
      TEMPORAL:=INSTR(UPPER(:new.MASTER),'MAST',1,1);--FORMATO: MAST###
      TEMPORAL1:=LENGTH(:new.MASTER);
      IF TEMPORAL!=1 AND TEMPORAL1>7 THEN
            :new.MASTER:=NULL;
      ELSE
            :new.MASTER:=UPPER(:new.MASTER);
            :new.NOMBRE_REGION:=INITCAP(LOWER(:new.NOMBRE_REGION));

      END IF;
END INF_GEOGRAF;
/
```

```
--ID DE PAISES

CREATE OR REPLACE TRIGGER PAIS
BEFORE INSERT ON PAISES
FOR EACH ROW
DECLARE
      TEMPORAL INTEGER;
BEGIN
      TEMPORAL:=LENGTH(:new.ID_PAIS);     --FORMATO: ###
      IF TEMPORAL>3 THEN
            :new.ID_PAIS:=NULL;
      ELSE
            :new.NOMBRE:=INITCAP(LOWER(:new.NOMBRE));
      END IF;
END PAIS;
/

--ID DE FOTOGRAFIAS

CREATE OR REPLACE TRIGGER FOTOGRAFIAS
BEFORE INSERT ON FOTOGRAFIAS
FOR EACH ROW
DECLARE
      TEMPORAL INTEGER;
BEGIN
      TEMPORAL:=LENGTH(:new.ID_FOTO);     --FORMATO: ######
      IF TEMPORAL>6 THEN
            :new.ID_FOTO:=NULL;
      END IF;
END FOTOGRAFIAS;
/

--ID DE USUARIOS FRECUENTES

CREATE OR REPLACE TRIGGER USUARIO_FREC
BEFORE INSERT ON USUARIOS_FRECUENTES
FOR EACH ROW
DECLARE
      TEMP1 INTEGER;
      TEMP2 INTEGER;
BEGIN
      TEMP1:=INSTR(UPPER(:new.PASSWORD_FREC),'UF',1,1);
      TEMP2:=LENGTH(:new.LOGIN_FREC);
      IF TEMP1!=2 AND TEMP2!=9 THEN
            :new.PASSWORD_FREC:=NULL;
            :new.LOGIN_FREC:=NULL;
      ELSE
            :new.NOMBRE:=INITCAP(LOWER(:new.NOMBRE));
            :new.APELLIDO:=INITCAP(LOWER(:new.APELLIDO));
      END IF;
END USUARIO_FREC;
/
```

```
                        --ID DE SEMANAS EN ESTUDIO

CREATE OR REPLACE TRIGGER SEM_ESTUDIO
BEFORE INSERT ON SEMANAS_EN_ESTUDIO
FOR EACH ROW
DECLARE
      TEMP1 INTEGER;
      TEMP2 INTEGER;
BEGIN
      TEMP1:=INSTR(:new.ID_SEM_EST,'-',1,1);    --FORMATO: ###-##
      TEMP2:=LENGTH(:new.ID_SEM_EST);
      IF TEMP1!=4 AND TEMP2>6 THEN
            :new.ID_SEM_EST:=NULL;

      END IF;
END SEM_ESTUDIO;
/

                        --ID DE TIPOS HORAS

CREATE OR REPLACE TRIGGER TIPO_HORA
BEFORE INSERT ON TIPOS_HORAS
FOR EACH ROW
DECLARE
      TEMP1 INTEGER;
      TEMP2 INTEGER;
BEGIN
      TEMP1:=INSTR(:new.HORA,'h',1,1);    --FORMATO: h##
      TEMP2:=LENGTH(:new.HORA);
      IF TEMP1!=1 AND TEMP2>3 THEN
            :new.HORA:=NULL;
      END IF;
END TIPO_HORA;
/

                        --ID DE SATELITES

CREATE OR REPLACE TRIGGER SATELITE
BEFORE INSERT ON SATELITES
FOR EACH ROW
DECLARE
      TEMP1 INTEGER;
      TEMP2 INTEGER;
BEGIN
      TEMP1:=INSTR(UPPER(:new.ID_SATELITE),'N',1,1); --FORMATO: N##
      TEMP2:=LENGTH(:new.ID_SATELITE);
      IF TEMP1!=1 OR TEMP2>3 THEN
            :new.ID_SATELITE:=NULL;
      ELSE
            :new.ID_SATELITE:=UPPER(:new.ID_SATELITE);
            :new.NOMBRE:=UPPER(:new.NOMBRE);
      END IF;
END SATELITE;
/
```

```
                        --ID DE ACERVOS FOTOGRAFICOS

CREATE OR REPLACE TRIGGER ACERVOS_FOTOGRAFICOS
BEFORE INSERT ON ACERVOS_FOTOGRAFICOS
FOR EACH ROW
DECLARE
      TEMP1 INTEGER;
BEGIN
      TEMP1:=LENGTH(:new.ID_ACERVO);       --FORMATO: ######
      IF TEMP1>6 THEN
            :new.ID_ACERVO:=NULL;
      END IF;
END ACERVOS_FOTOGRAFICOS;
/
                                --ID DE CONVENIOS

CREATE OR REPLACE TRIGGER CONVENIOS
BEFORE INSERT ON CONVENIOS
FOR EACH ROW
DECLARE
      TEMP INTEGER;
BEGIN
      TEMP:=LENGTH(:new.NUM_CONV);  --FORMATO: ####
      IF TEMP>4 THEN
            :new.NUM_CONV:=NULL;
      END IF;
END CONVENIOS;
/
                                --ID DE CARGOS

CREATE OR REPLACE TRIGGER CARGOS
BEFORE INSERT ON CARGOS
FOR EACH ROW
DECLARE
      TEMP INTEGER;
BEGIN
      TEMP:=LENGTH(:new.ID_CARGO);  --FORMATO. ##
      IF TEMP>2 THEN
            :new.ID_CARGO:=NULL;
      ELSE
            :new.NOMBRE:=INITCAP(LOWER(:new.NOMBRE));
      END IF;
END CARGOS;
/
```

CREACIÓN DE CONSTRAINTS

```
ALTER TABLE TIPOS_IMAGEN
ADD CONSTRAINT checa_timag
CHECK(T_IMAGEN IN (01,02,03)) DEFERRABLE
/
```

```
ALTER TABLE INFORMACIONES_CRUDAS
ADD CONSTRAINT checa_inf_crudas
CHECK(TIPO_INFORMACION IN ('I_PFOTO','I_SPROC','i_pfoto','i_sproc'))
DEFERRABLE
/
ALTER TABLE TIPOS_LADAS
ADD CONSTRAINT checa_lada
CHECK(LADA_TIPO IN ('LN','LI','ln','li')) DEFERRABLE
/
ALTER TABLE USUARIOS
ADD CONSTRAINT checa_usr
CHECK(TIPO_USR IN ('U_LOF','U_ICM','U_ESP','u_lof','u_icm','u_esp'))
DEFERRABLE
/
```

PROCEDIMIENTO ALMACENADO

```
CREATE OR REPLACE PROCEDURE CARGA_IMAGEN (
id                  tipos_imagen.id_tip_imag%TYPE,
nombrei             tipos_imagen.nombre%TYPE,
horac               tipos_imagen.hora_captura%TYPE,
mesi                tipos_imagen.mes%TYPE,
anioi               tipos_imagen.anio%TYPE,
tamanioi            tipos_imagen.tamanio%TYPE,
tipoi               tipos_imagen.t_imagen%TYPE,
id_cruda            tipos_imagen.inf_crud_id_inf_cruda%TYPE,
tipo_hora           tipos_imagen.tip_hr_hora%TYPE,
nombreimagen              IN VARCHAR2,
comenti             tipos_imagen.comentario%TYPE) AS
apuntador_bfile     BFILE;
apuntador_blob      BLOB;
tamanio_imagen      INTEGER;
BEGIN
      DBMS_OUTPUT.ENABLE(1000000);
      apuntador_bfile := BFILENAME('IMAGENES',nombreimagen);
      DBMS_LOB.FILEOPEN(apuntador_bfile);
      tamanio_imagen := DBMS_LOB.GETLENGTH(apuntador_bfile);
      DBMS_OUTPUT.PUT_LINE('El tamanio es: '|| tamanio_imagen);
      INSERT INTO TIPOS_IMAGEN VALUES (
            id,nombrei,horac,mesi,anioi,tamanioi,tipoi,id_cruda,
            tipo_hora,'','','',EMPTY_BLOB(),comenti)
      RETURNING nombre_imagen INTO apuntador_blob;
      DBMS_LOB.LOADFROMFILE(
                        apuntador_blob,apuntador_bfile,tamanio_imagen);
      DBMS_LOB.FILECLOSE(apuntador_bfile);
      COMMIT;
      EXCEPTION
            WHEN OTHERS THEN
                  DBMS_OUTPUT.PUT_LINE(
                              'Tipo de Excepcion OTHERS ' || sqlerrm);
END Carga_Imagen;
/
```

SUPERTIPOS

```
                                --TIPOS_IMAGEN

CREATE OR REPLACE TRIGGER DISPARA1_TIMAG
BEFORE INSERT ON TIPOS_IMAGEN
FOR EACH ROW
DECLARE
      T_IMG NUMBER(2);
      TEMP1 INTEGER;
      TEMP2 INTEGER;
      TEMP3 INTEGER;
      TEMP4 INTEGER;
      TEMP5 INTEGER;
      TEMP6 INTEGER;
BEGIN
      T_IMG:=:new.T_IMAGEN;
      TEMP1:=INSTR(:new.ID_TIP_IMAG,'-',1,1);   -- ID TABLA: ##-$$-####
      TEMP2:=INSTR(:new.ID_TIP_IMAG,'-',1,2);
      TEMP3:=LENGTH(:new.ID_TIP_IMAG);                  --TAMAÑO ID TABLA=10
      TEMP4:=INSTR(UPPER(:new.ID_TIP_IMAG),'ID',1,1);       --IMAG.DIARIA
      TEMP5:=INSTR(UPPER(:NEW.ID_TIP_IMAG),'IS',1,1);       --IMAG.SEMANAL
      TEMP6:=INSTR(UPPER(:NEW.ID_TIP_IMAG),'IM',1,1);       --IMAG.MENSUAL

IF T_IMG=1 AND TEMP1=3 AND TEMP3=10 AND TEMP2=6 AND TEMP4=4 THEN --DIARIA
            :new.SEM_DEL_MES:=NULL;
            :new.COMENTARIO:=NULL;
            :new.NOMBRE:=INITCAP(LOWER(:new.NOMBRE));

            IF :new.DIA IS NULL OR :new.SST IS NULL THEN
                  :new.ID_TIP_IMAG:=NULL;
            END IF;
ELSIF T_IMG=2 AND TEMP1=3 AND TEMP3=10 AND TEMP2=6 AND TEMP5=4 THEN --SEM
            :new.DIA:=NULL;
            :new.SST:=NULL;
            :new.COMENTARIO:=NULL;
            :new.NOMBRE:=INITCAP(LOWER(:new.NOMBRE));
            IF :new.SEM_DEL_MES IS NULL THEN
                  :new.ID_TIP_IMAG:=NULL;
            END IF;
ELSIF T_IMG=3 AND TEMP1=3 AND TEMP3=10 AND TEMP2=6 AND TEMP6=4 THEN--MENS
            :new.DIA:=NULL;
            :new.SST:=NULL;
            :new.SEM_DEL_MES:=NULL;
            :new.NOMBRE:=INITCAP(LOWER(:new.NOMBRE));
            IF :new.NOMBRE_IMAGEN IS NULL THEN
                  :new.ID_TIP_IMAG:=NULL;
            END IF;
ELSE
            :new.ID_TIP_IMAG:=NULL;
END IF;
END DISPARA1_TIMAG;
/
```

```
                        -- INFORMACIONES CRUDAS

CREATE OR REPLACE TRIGGER DISPARA3_INF_CRUDA
BEFORE INSERT ON INFORMACIONES_CRUDAS
FOR EACH ROW
DECLARE
      TEMP varchar2(7);
      BAND1 INTEGER;
      BAND2 INTEGER;
      BAND3 INTEGER;
      TEMP1 INTEGER;
      TEMP2 INTEGER;
BEGIN
      TEMP:=UPPER(:new.TIPO_INFORMACION);              --I_SPROC,I_PFOTO

      BAND1:=INSTR(:new.ID_INF_CRUDA,'-',1,1);         -- ID: ##-$$-####

      BAND2:=INSTR(:new.ID_INF_CRUDA,'-',1,2);

      BAND3:=LENGTH(:new.ID_INF_CRUDA);                --TAMAÑO ID=10

      TEMP1:=INSTR(UPPER(:new.ID_INF_CRUDA),'SP',1,1);--IMG.SIN PROCESAR

      TEMP2:=INSTR(UPPER(:new.ID_INF_CRUDA),'IF',1,1);--IMAG. PARA FOTO

IF TEMP='I_PFOTO' AND TEMP2=4 AND BAND1=3 AND BAND2=6 AND BAND3<11 THEN
            --IMAG.PARA FOTO
            :new.TIPO_INFORMACION:=TEMP;
            :new.PARAM_ELEV_SAT:=NULL;
            :new.PARAM_ELEV_SOL:=NULL;
            :new.COMENTARIO:=NULL;
            :new.SAT_ID_SATELITE:=UPPER(:new.SAT_ID_SATELITE);

            IF :new.TER IS NULL THEN
                  :new.ID_INF_CRUDA:=NULL;
            END IF;

ELSIF TEMP='I_SPROC' AND TEMP1=4 AND BAND1=3 AND BAND2=6 AND BAND3<11
THEN
            --IMAG SIN PROCESAR
            :new.TIPO_INFORMACION:=TEMP;
            :new.TER:=NULL;
            :new.SAT_ID_SATELITE:=UPPER(:new.SAT_ID_SATELITE);

            IF :new.PARAM_ELEV_SAT IS NULL OR :new.PARAM_ELEV_SOL IS NULL
            THEN
                  :new.ID_INF_CRUDA:=NULL;
            END IF;

ELSE
            :new.ID_INF_CRUDA:=NULL;
END IF;

END DISPARA3_INF_CRUDA;
/
```

```
                                -- TIPOS_LADA

CREATE OR REPLACE TRIGGER DISPARA5_TLADA
BEFORE INSERT ON TIPOS_LADAS
FOR EACH ROW
DECLARE

      TIPOLADA VARCHAR2(2);

      TEMP1 INTEGER;

      TEMP2 INTEGER;

      TEMP3 INTEGER;

BEGIN
      TEMP1:=LENGTH(:new.CLAVE_LADA);

      TEMP2:=INSTR(UPPER(:new.CLAVE_LADA),'N',1,1);   --ID LADA NACIONAL

      TEMP3:=INSTR(UPPER(:new.CLAVE_LADA),'I',1,1);   --ID LADA INTERNAC.

      TIPOLADA:=UPPER(:new.LADA_TIPO);

      IF TIPOLADA='LN' AND TEMP3<11 AND TEMP1<11 THEN--LADA NACIONAL
            :new.NOMBRE_REGION:=NULL;
            :new.LADA_TIPO:=UPPER(:new.LADA_TIPO);
            :new.CLAVE_LADA:=UPPER(:new.CLAVE_LADA);
            :new.NOMBRE_POBLACION:=INITCAP(LOWER(:new.NOMBRE_POBLACION));

            IF :new.NOMBRE_POBLACION IS NULL THEN
                  :new.CLAVE_LADA:=NULL;
            END IF;

      ELSIF TIPOLADA='LI' AND TEMP2<11 AND TEMP1<11 THEN--LADA INTERN.
            :new.NOMBRE_POBLACION:=NULL;
            :new.LADA_TIPO:=UPPER(:new.LADA_TIPO);
            :new.CLAVE_LADA:=UPPER(:new.CLAVE_LADA);
            :new.NOMBRE_REGION:=INITCAP(LOWER(:new.NOMBRE_REGION));

            IF :new.NOMBRE_REGION IS NULL THEN
                  :new.CLAVE_LADA:=NULL;
            END IF;

      ELSE
            :new.CLAVE_LADA:=NULL;
      END IF;

END DISPARA5_TLADA;
/
```

```
                                        -- USUARIOS

CREATE OR REPLACE TRIGGER DISPARA7_USR
BEFORE INSERT ON USUARIOS
FOR EACH ROW
DECLARE
      USR VARCHAR2(5);
      TEMP1 INTEGER;
      TEMP2 INTEGER;
      TEMP3 INTEGER;
      TEMP4 INTEGER;
BEGIN
      USR:=:new.TIPO_USR;
      TEMP1:=INSTR(UPPER(:new.PASSWORD),'UI',1,1);    --ID USUARIO ICML
      TEMP2:=INSTR(UPPER(:new.PASSWORD),'UL',1,1);    --ID USUARIO LOF
      TEMP3:=INSTR(UPPER(:new.PASSWORD),'UE',1,1);    --ID USUARIO ESP.
      TEMP4:=LENGTH(:new.LOGIN);
      IF USR='U_LOF' AND TEMP2=2 AND TEMP4<9 THEN     --USUARIO LOF
            :new.LABORATORIO:=NULL;
            :new.US_ICML_COMENTARIO:=NULL;
            :new.DET_LUGAR_PAIS_ID_PAIS:=NULL;
            :new.DET_LUGAR_TIP_LADA_CLAVE_LADA:=NULL;
            :new.CONV_NUM_CONV:=NULL;
            :new.INSTITUCION:=NULL;
            :new.DIRECCION:=NULL;
            :new.TELEFONO:=NULL;
            :new.EMAIL:=NULL;
            :new.FAX:=NULL;
      ELSIF USR='U_ICM' AND TEMP1=2 AND TEMP4<9 THEN --USUARIO ICML
            :new.COMENTARIO:=NULL;
            :new.DET_LUGAR_TIP_LADA_CLAVE_LADA:=NULL;
            :new.CONV_NUM_CONV:=NULL;
            :new.INSTITUCION:=NULL;
            :new.DIRECCION:=NULL;
            :new.TELEFONO:=NULL;
            :new.EMAIL:=NULL;
            :new.FAX:=NULL;
            IF :new.LABORATORIO IS NULL THEN
                  :new.LOGIN:=NULL;
                  :new.PASSWORD:=NULL;
            END IF;
      ELSIF USR='U_ESP' AND TEMP3=2 AND TEMP4<9 THEN --USUARIO ESPECIAL
            :new.COMENTARIO:=NULL;
            :new.US_ICML_COMENTARIO:=NULL;
            :new.LABORATORIO:=NULL;
            IF :new.INSTITUCION IS NULL OR :new.DIRECCION IS NULL OR
:new.TELEFONO IS NULL OR :new.EMAIL IS NULL THEN
                  :new.LOGIN:=NULL;
                  :new.PASSWORD:=NULL;
            END IF;
      ELSE
            :new.LOGIN:=NULL;
            :new.PASSWORD:=NULL;
      END IF;
END DISPARA7_USR;
/
```

ARCOS

```
                              -- IMAGENES_BITACORAS

CREATE OR REPLACE TRIGGER ARCO_IMG_BITACORAS
BEFORE INSERT ON IMAGENES_BITACORAS
FOR EACH ROW
DECLARE

      ID_TIPO VARCHAR2(10);

      TEMP INTEGER;

      TEMP2 INTEGER;

      TEMP4 INTEGER;

      TEMP5 INTEGER;

      TEMP6 INTEGER;

BEGIN

      TEMP:=LENGTH(:new.T_IMAGEN_ID_TIP_IMAG);        --TAMAÑO ID FORANEA

      TEMP2:=INSTR(:new.T_IMAGEN_ID_TIP_IMAG,'-',1,1);--ID TIPOS_IMAGEN

      TEMP4:=INSTR(:new.T_IMAGEN_ID_TIP_IMAG,'-',1,2);--: ##-$$-####

      TEMP5:=INSTR(UPPER(:new.T_IMAGEN_ID_TIP_IMAG),'IS',1,1);--SEMANAL

      TEMP6:=INSTR(UPPER(:new.T_IMAGEN_ID_TIP_IMAG),'IM',1,1);--MENSUAL

      IF TEMP=10 AND TEMP2=3 AND TEMP4=6 AND  TEMP5=4 THEN
            SELECT ID_TIP_IMAG INTO ID_TIPO FROM TIPOS_IMAGEN
            WHERE ID_TIP_IMAG=:new.T_IMAGEN_ID_TIP_IMAG;

      ELSIF TEMP=10 AND TEMP2=3 AND TEMP4=6 AND  TEMP6=4 THEN
            SELECT ID_TIP_IMAG INTO ID_TIPO FROM TIPOS_IMAGEN
            WHERE ID_TIP_IMAG=:new.T_IMAGEN_ID_TIP_IMAG;

      ELSE
            :new.BIT_MESTUD_ID_BITACORA:=NULL;
      END IF;

      EXCEPTION
      WHEN NO_DATA_FOUND THEN
            :new.BIT_MESTUD_ID_BITACORA:=NULL;
END ARCO_IMG_BITACORAS;
/
```

```
                                -- IMAGENES_USUARIOS

CREATE OR REPLACE TRIGGER ARCO_IMG_USUARIO
BEFORE INSERT ON IMAGENES_USUARIOS
FOR EACH ROW
DECLARE
BAND1 INTEGER;
BAND2 INTEGER;
BAND3 INTEGER;
BAND4 INTEGER;
TEMP INTEGER;
TEMP1 NUMBER(6);
TEMP2 VARCHAR2(10);
TEMP3 VARCHAR2(8);
TEMP5 INTEGER;
TEMP6 INTEGER;
TEMP7 INTEGER;
TEMP8 INTEGER;
TEMP9 INTEGER;
BEGIN
      TEMP:=LENGTH(:new.NUM_CONSULTA);    --ID DE IMAGS.USUARIOS: ####
      BAND1:=LENGTH(:new.ID_TIPO_INF);    --ID DE ACERVO: ######
      BAND2:=INSTR(:new.ID_TIPO_INF,'-',1,1);   --ID INF.CRUDA: ##-$$-####
      BAND3:=INSTR(:new.ID_TIPO_INF,'-',1,2);
      BAND4:=INSTR(:new.ID_TIPO_INF,'/',1,1);   --ID REC.REGION: ###/####
      TEMP5:=INSTR(UPPER(:new.ID_TIPO_INF),'SP',1,1);--IMAG SIN PROCESAR
      TEMP6:=INSTR(UPPER(:new.ID_TIPO_INF),'IF',1,1);--IMAG PARA FOTO
      TEMP7:=INSTR(UPPER(:new.ID_TIPO_INF),'IS',1,1);--IMAG SEMANAL
      TEMP8:=INSTR(UPPER(:new.ID_TIPO_INF),'IM',1,1);--IMAG MENSUAL
      TEMP9:=INSTR(UPPER(:new.ID_TIPO_INF),'ID',1,1);--IMAG DIARIA
      IF TEMP<5 AND BAND1<7 THEN          -- ID DE ACERVO
            SELECT ID_ACERVO INTO TEMP1 FROM ACERVOS_FOTOGRAFICOS
            WHERE ID_ACERVO=:new.ID_TIPO_INF;

      ELSIF TEMP<5 AND BAND1=10 AND BAND2=3 AND BAND3=6 AND TEMP5=4 THEN
            -- IMAG SIN PROCESAR
            SELECT ID_INF_CRUDA INTO TEMP2 FROM INFORMACIONES_CRUDAS
            WHERE ID_INF_CRUDA=:new.ID_TIPO_INF;

      ELSIF TEMP<5 AND BAND1=10 AND BAND2=3 AND BAND3=6 AND TEMP6=4 THEN
            -- IMAG PARA FOTO
            SELECT ID_INF_CRUDA INTO TEMP2 FROM INFORMACIONES_CRUDAS
            WHERE ID_INF_CRUDA=:new.ID_TIPO_INF;

      ELSIF TEMP<5 AND BAND4=4 THEN       --ID RECORTE
            SELECT ID_RECORTE INTO TEMP3 FROM RECORTES_DE_REGIONES
            WHERE ID_RECORTE=:new.ID_TIPO_INF;

      ELSIF TEMP<5 AND BAND1=10 AND BAND2=3 AND BAND3=6 AND TEMP9=4 THEN
            --IMAG DIARIA
            SELECT ID_TIP_IMAG INTO TEMP2 FROM TIPOS_IMAGEN
            WHERE ID_TIP_IMAG=:new.ID_TIPO_INF;

      ELSIF TEMP<5 AND BAND1=10 AND BAND2=3 AND BAND3=6 AND TEMP7=4 THEN
            SELECT ID_TIP_IMAG INTO TEMP2 FROM TIPOS_IMAGEN
            WHERE ID_TIP_IMAG=:new.ID_TIPO_INF;       --IMAG SEMANAL
```

```
      ELSIF TEMP<5 AND BAND1=10 AND BAND2=3 AND BAND3=6 AND TEMP8=4 THEN
            --IMAG MENSUAL
            SELECT ID_TIP_IMAG INTO TEMP2 FROM TIPOS_IMAGEN
            WHERE ID_TIP_IMAG=:new.ID_TIPO_INF;

      ELSE
            :new.NUM_CONSULTA:=NULL;
      END IF;
      EXCEPTION
            WHEN NO_DATA_FOUND THEN
            :new.NUM_CONSULTA:=NULL;
END ARCO_IMG_USUARIO;
/

                              -- RECORTES_DE_REGIONES

CREATE OR REPLACE TRIGGER ARCO_RECORTES
BEFORE INSERT ON RECORTES_DE_REGIONES
FOR EACH ROW
DECLARE
      BAND1 INTEGER;
      BAND2 INTEGER;
      BAND3 INTEGER;
      TEMP1 VARCHAR2(10);
      TEMP3 INTEGER;
      TEMP4 INTEGER;
      TEMP5 INTEGER;
      TEMP6 INTEGER;
      TEMP7 INTEGER;
      TEMP8 INTEGER;
      TEMP9 INTEGER;
BEGIN
      :new.INF_GEGRAF_MASTER:=UPPER(:new.INF_GEGRAF_MASTER);
      TEMP3:=INSTR(:new.ID_RECORTE,'/',1,1);    --ID FORMATO: ###/####
      TEMP4:=LENGTH(:new.ID_RECORTE);
      BAND1:=INSTR(:new.INFCRUDA_TIMAGEN,'-',1,1);    --FORMATO ##-$$-####
      BAND2:=INSTR(:new.INFCRUDA_TIMAGEN,'-',1,2);
      BAND3:=LENGTH(:new.INFCRUDA_TIMAGEN);           --TAMAÑO=10
      TEMP5:=INSTR(UPPER(:new.INFCRUDA_TIMAGEN),'ID',1,1); --IMAG DIARIA
      TEMP6:=INSTR(UPPER(:new.INFCRUDA_TIMAGEN),'IS',1,1); --IMAG SEMANAL
      TEMP7:=INSTR(UPPER(:new.INFCRUDA_TIMAGEN),'IM',1,1); --IMAG MENSUAL
      TEMP8:=INSTR(UPPER(:new.INFCRUDA_TIMAGEN),'IF',1,1); --IMAG.FOTO
      TEMP9:=INSTR(UPPER(:new.INFCRUDA_TIMAGEN),'SP',1,1); --IMG SINPROC

IF BAND1=3 AND BAND2=6 AND TEMP3=4 AND TEMP4<9 AND BAND3=10 AND TEMP8=4
THEN
            --IMAGEN PARA FOTO
            SELECT ID_INF_CRUDA INTO TEMP1 FROM INFORMACIONES_CRUDAS
            WHERE ID_INF_CRUDA=:new.INFCRUDA_TIMAGEN;

ELSIF BAND1=3 AND BAND2=6 AND TEMP3=4 AND TEMP4<9 AND BAND3=10 AND
TEMP9=4 THEN
            --IMAGEN SIN PROCESAR
            SELECT ID_INF_CRUDA INTO TEMP1 FROM INFORMACIONES_CRUDAS
            WHERE ID_INF_CRUDA=:new.INFCRUDA_TIMAGEN;
```

```
ELSIF BAND3=10 AND TEMP3=4 AND TEMP4<9 AND BAND2=6 AND TEMP5=4 THEN
            --IMAGEN DIARIA
            SELECT ID_TIP_IMAG INTO TEMP1 FROM TIPOS_IMAGEN
            WHERE ID_TIP_IMAG=:new.INFCRUDA_TIMAGEN;

ELSIF BAND3=10 AND TEMP3=4 AND TEMP4<9 AND BAND2=6 AND TEMP6=4 THEN
            --IMAGEN SEMANAL
            SELECT ID_TIP_IMAG INTO TEMP1 FROM TIPOS_IMAGEN
            WHERE ID_TIP_IMAG=:new.INFCRUDA_TIMAGEN;

ELSIF BAND3=10 AND TEMP3=4 AND TEMP4<9 AND BAND2=6 AND TEMP7=4 THEN
            --IMAGEN MENSUAL
            SELECT ID_TIP_IMAG INTO TEMP1 FROM TIPOS_IMAGEN
            WHERE ID_TIP_IMAG=:new.INFCRUDA_TIMAGEN;

ELSE
            :new.ID_RECORTE:=NULL;
END IF;

      EXCEPTION
            WHEN NO_DATA_FOUND THEN
            :new.ID_RECORTE:=NULL;

END ARCO_RECORTES;
/
```

ANEXO

Oracle 8i

Oracle8i se basa en los puntos fuertes de sus predecesores, Oracle7 y Oracle8. Oracle7, lanzado originalmente a principios de 1993, definió unos estándares muy exigentes para los sistemas de gestión de base de datos relacionales más sofisticados. La gran cantidad de funciones de Oracle7 lo convirtieron en un servidor de base de datos polivalente para todo tipo de aplicaciones comerciales, incluyendo [Bobrowski, 2001]:

- **Procesamiento de transacciones en línea (OLTP, Online Transaction Processing).** Son aplicaciones que procesan muchas transacciones de actualización pequeñas, como los sistemas bancarios, de reservas y de entrada de pedidos.
- **Sistemas de toma de decisión (DSS, Decisión Support Systems).** Son aplicaciones que consultan la información buscada desde una base de datos para fines de análisis de datos.
- **Grandes Almacenes de Datos.** Son aplicaciones que acceden a las grandes bases de datos de sólo lectura, optimizadas específicamente para un acceso rápido incluso para la información más impenetrable.

Oracle8.0, que apareció en el verano de 1997, añadió muchas características nuevas para aumentar la capacidad de Oracle7 y hacer de Oracle un instrumento, más adecuado para los entornos de aplicaciones más exigentes y complejos. Las características nuevas de Oracle8.0 incluían el particionamiento de datos, los tipos de objetos y los métodos, tipos de datos Objetos grandes (LOB, *Large Objects*), la gestión de contraseñas, la utilidad Recovery Manager, etc.

Oracle8i, que apareció en la primavera de 1999, es de los lanzamientos más recientes de Oracle. Oracle8i mejora el lanzamiento original de Oracle8 en dos áreas primarias:

- Los grandes almacenes de datos
- El desarrollo de aplicaciones en web.

Para grandes almacenes de datos, Oracle8i incluye muchas características nuevas específicamente diseñadas para mejorar el rendimiento del procesamiento de consultas complejas, como las vistas materializadas, la reescritura automática de consultas y los índices basados en funciones. Para el desarrollo de aplicaciones basadas en web, Oracle8i viene con una VM (máquina virtual, *Virtual Machine*) de Java, de modo que los desarrolladores pueden construir todos los componentes de aplicación utilizando Java (inclusive los procedimientos almacenados basados en Java, las funciones y los paquetes), o acceder a la información de base de datos utilizando las aplicaciones de Java. Un próximo lanzamiento de Oracle8i incluirá también Sistema de archivos Internet (IFS, *Internet File System*) de Oracle, que es básicamente una interfaz de arrastrar y colocar para la manipulación de la información de base de datos.

OPCIONES DE LICENCIA DE ORACLE8i

Oracle 8i está disponible para los sistemas operativos más populares así cómo en varios formatos distintos de licencia [Bobrowski, 2001]:

- **Oracle8i.** La versión básica de Oracle8i, que incluye las opciones y funciones más utilizadas en Oracle8i.
- **Oracle 8i Enterprise Edition.** La versión completa de Oracle8i que ofrece el acceso multiusuario a todas las funciones, inclusive las que sirven para el procesamiento de base de datos más sofisticado, el acceso a las bases de datos basadas en web y los grandes almacenes de datos.
- **Oracle8i Personal Edition.** Una licencia de base de datos de desarrollo de usuario sencillo que ofrece acceso a la mayoría de las funciones de Oracle8i Enterprise Edition.
- **Oracle8i Lite.** Una base de datos compatible con Oracle8i diseñada para su uso en los entornos de computadora portátil.

Una base de datos en Oracle es una recopilación de archivos de sistemas operativos relacionados que utiliza Oracle8i para almacenar y manipular un sistema de información relacionada. Desde un punto de vista estructural, una base de datos de Oracle tiene tres tipos de archivos primarios: archivos de datos, archivos log y archivos de control.

Una instancia de base de datos es el juego de los procesos y áreas de memoria del sistema operativo utilizadas por Oracle8i para manipular el acceso a la base de datos. No se puede acceder a una base de datos de Oracle hasta después de iniciar una instancia asociada con los archivos de la base de datos física.

SQL*PLUS

Es un tipo de aplicación que se puede utilizar para introducir los comandos SQL e interaccionar con un sistema de base de datos de Oracle es una herramienta de consulta *adhoc*, como SQL*PLUS de Oracle. SQL*PLUS le proporciona una interfaz de línea de comandos muy sencilla que se puede utilizar para introducir las sentencias SQL y después ver los resultados de la ejecución de cada sentencia. En realidad SQL*PLUS permite dirigirse a un servidor de base de datos de Oracle de modo que se pueda consultar la base de datos en busca de información, o introducir, actualizar y eliminar datos de la base de datos.

SQL

Para trabajar con Oracle8i, las aplicaciones deben utilizar los comandos SQL. SQL es un lenguaje de comandos relativamente sencillo utilizado por los administradores de base de datos, los desarrolladores y los usuarios de aplicaciones para:

- Recuperar, introducir, actualizar y eliminar los datos de base de datos.
- Crear, cambiar y colocar los objetos de base de datos.
- Restringir el acceso a los datos de base de datos y a las operaciones del sistema.

La única manera en que una aplicación puede interactuar con un servidor de base de datos de Oracle es emitiendo un comando SQL. Las interfaces gráficas de usuario sofisticadas podrían ocultar los comandos SQL a los usuarios y desarrolladores, pero una aplicación siempre se comunica entre bastidores con Oracle utilizando SQL.

TIPOS DE COMANDOS SQL

Las cuatro categorías primarias de SQL son DML, control de transacciones , DDL y DCL [Alonso, 1998].

- Los comandos de manipulación de datos o del *lenguaje de modificación de datos* (DML, *Data Modification Language*) son comandos de SQL que recuperan, insertan, actualizan y suprimen filas de tablas en un a base de datos de Oracle. Los cuatro comandos básicos de DML son SELECT, INSERT, UPDATE y DELETE.

- Las aplicaciones que utilizan SQL y las bases de datos relacionales realizan el trabajo utilizando transacciones. Una *transacción de base de datos* es una unidad de trabajo realizada por una o más sentencias SQL relacionadas. Para preservar la integridad de la información en una base de datos, las bases de datos relacionales como Oracle aseguran que todas las tareas de una transacción se validan o se deshacen. Una aplicación utiliza los comandos SQL de *control de transacciones* COMMIT y ROLLBACK para controlar el resultado de una transacción de base de datos.

- Los comandos de *Lenguaje de definición de datos* (DDL, *Data Definition Languaje*) crean, cambien y colocan los objetos de base de datos. Casi todos los tipos de objeto de datos tienen comandos correspondientes de CREATE, ALTER y DROP`s.

- Una palicación administrativa utiliza los comandos del *Lenguaje de control de datos* (DCL, *Data Control Languaje*) para controlar el acceso del usuario a una base de datos de Oracle. Los tres comandos DCL más habituales son GRANT, REVOKE y SET ROLE.

PL/SQL

PL/SQL es un lenguaje de programación de procedimientos incorporados en la mayoría de los productos Oracle.

Con PL/SQL se pueden construir programas para procesar información combinando las sentencias de procedimientos de PL/SQL que controlan el flujo de programas con sentencias SQL que acceden a una base de datos de Oracle.

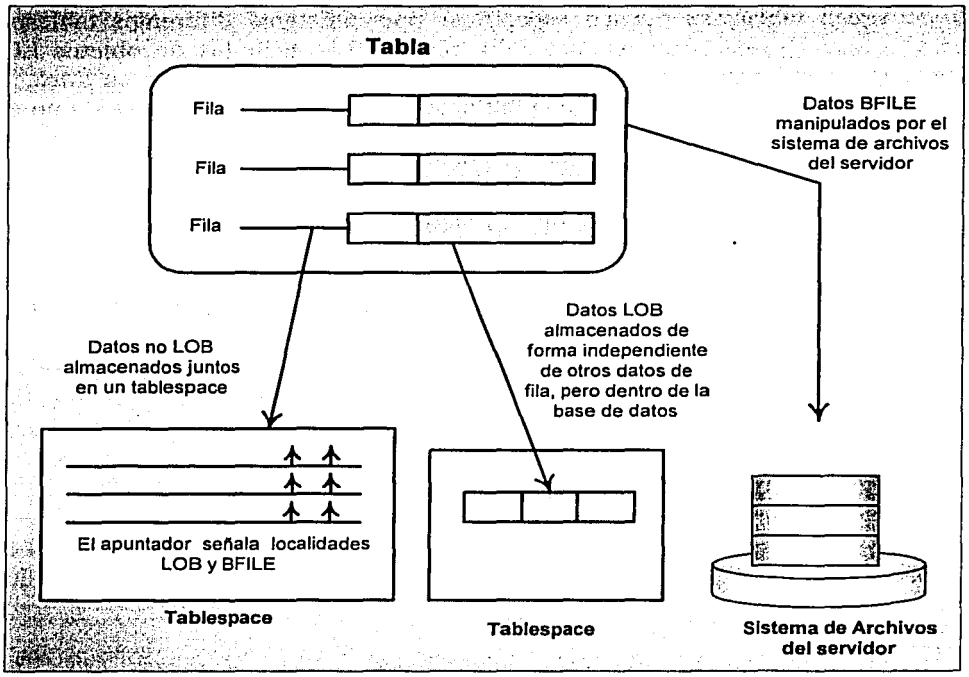
ASPECTOS FUNDAMENTALES DE LA CODIFICACIÓN PL/SQL

Todos los lenguajes de procedimientos, como PL/SQL, tienen unos elementos de lenguaje fundamentales y una funcionalidad que deben estudiarse antes de poder construir programas utilizando el lenguaje. A continuación se presentan los elementos básicos de PL/SQL, incluyendo los siguientes:

- Cómo declara las variables de programa y asignarles valores.
- Cómo controlar el flujo de programas con los bucles y la lógica condiciona.
- Cómo embeber las sentencias SQL e interactuar con las bases de datos de Oracle.
- Cómo declarar y utilizar subprogramas (procedimientos y funciones) dentro de los bloques PL/SQL.
- Cómo declarar los tipos definidos por el usuario, como los registros y las tablas anidadas.
- Cómo declarar y utilizar los cursores para procesar las consultas que se vuelven filas múltiples.
- Cómo utilizar los manejadores de excepciones para manejar las condiciones de error.

ALMACENAMIENTO DE DATOS ÚNICO PARA DATOS MULTIMEDIA

Cuando una tabla en una base de datos incluye una columna que utiliza un tipo de datos LOB (por ejemplo: CLOB, BLOB o NCLOB) o un tipo de datos BFILE, Oracle sólo almacena un pequeño localizador en línea con cada fila de la tabla. Un *localizador* es un puntero hacia la ubicación de los datos LOB o BFILE de la fila. Para las columnas CLOB, BLOB y NCLOB, un localizador LOB apunta a una ubicación de almacenamiento en la base de datos; en el caso de una columna BFILE, un localizador BFILE apunta a un archivo externo gestionado por el sistema operativo del servidor. A continuación la siguiente figura describe el concepto de los localizadores para las columnas LOB y BFILE [Bobrowski, 2001].



Distribución del almacenamiento de datos de la tabla primaria y datos multimedia relacionados a distintas ubicaciones físicas para reducir la contención de disco.

Se puede observar en la figura que una columna LOB puede tener características de almacenamiento independientes de las de la tabla que la contiene. Esto facilita cumplir los requisitos de los discos grandes normalmente asociados con los LOB. En este ejemplo , la tabla almacena todos los datos no LOB y no BFILE en cada fila juntos en un espacio de la tabla (técnicamente se llama tablespace), los datos de una columna LOB en otro tablespace y los datos de una columna BFILE en el sistema de archivos del servidor. Con ello podrá distribuir el almacenamiento de los datos principales de la tabla y los datos multimedia relacionados para ubicaciones físicas diferentes para reducir la contención de disco y mejorar el rendimiento global del sistema.

TIPOS DE DATOS LOB

La construcción en tipos de datos LOB, BLOB, CLOB y NCLOB (almacenados internamente) y el tipo BFILE (almacenado externamente), puede ser para guardar datos grandes y no estructurados, por ejemplo texto, imágenes, video y datos especiales de hasta 4 gigabytes de tamaño.

Cuando se crean tablas, se pueden especificar opcionalmente diferentes tablespaces y características de almacenamiento para columnas LOB o atributos de objetos LOB.Las columnas contienen apuntadores que pueden referenciar fuera de línea o en línea, valores LOB. Al seleccionar un tipo LOB de una tabla, se obtiene el apuntador LOB y no el valor completo LOB. El paquete DBMS_LOB y las operaciones de la Interfaz de llamadas de Oracle (OCI), utilizan los apuntadores para trabajar con estos tipos de datos LOB.

Los tipos LOB son similares a los tipos LONG y LONG RAW, pero difieren en lo siguiente:

- Los tipos LOB pueden ser atributos de un tipo de dato definido por el usuario.
- El apuntador LOB es almacenado en una columna de la tabla, con o sin el valor actual LOB. BLOB, NCLOB y CLOB pueden ser almacenados en tablespace separados. El tipo BFILE es almacenado en un archivo externo en el servidor.
- Cuando se accesa a una columna LOB, el apuntador LOB es regresado.
- Un tipo LOB puede tiene la capacidad de almacenar hasta 4 gigabytes. El tamaño máximo de los BFILEs depende del sistema operativo, pero no puede exceder los 4 gigabytes.
- Los LOBs permiten eficiencia, aleatoriedad, acceso a piezas específicas de información y manipulación de datos.
- Con la excepción del NCLOB, se pueden definir uno o más atributos LOB en un objeto.
- Se puede definir más de una columna LOB en una tabla.
- Se pueden declarar variables LOB bind.
- Se pueden seleccionar (SELECT) columnas y atributos LOB.
- Es posible insertar nuevos renglones o actualizar renglones existentes que contengan una o más columnas LOB y/o un objeto con uno o más atributos LOB. Se puede inicializar el valor interno LOB a NULL, vacío o reemplazar un LOB completo por datos. Es posible colocar un BFILE a NULL o hacer que apunte a un archivo diferente.
- Se puede actualizar una intersección renglón/columna LOB o atributo LOB con otra intersección renglón/columna LOB o atributo LOB.

- Es posible borrar un renglón conteniendo una columna o atributo LOB y también se borrara el valor LOB (una columna LOB almacenada en la base de datos). Se debe utilizar primero la sentencia INSERT para inicializar el valor interno LOB a empty (vacío). Una vez que el renglón es insertado, se puede seleccionar el empty_lob y manejarlo utilizando el paquete DBMS_LOB o la OCI.

El siguiente ejemplo crea una tabla con columnas LOB:

```
CREATE TABLE nombre_tabla (campo1 CHAR (40),
                           campo2 CLOB,
                           campo3 BLOB)
    LOB (resume) STORE AS
                ( TABLESPACE resumes
                STORAGE (INITIAL 5M NEXT 5M) );
```

TIPO DE DATO BFILE

Este tipo de dato posibilita el acceso a archivos binarios LOBs, que son almacenados en el sistema, fuera de la base de datos. Una columna o atributo BFILE almacena un localizador BFILE, el cual sirve como apuntador a un archivo binario en el sistema del servidor. El apuntador mantiene el alias del directorio y el nombre del archivo.

Los archivos binarios LOBs no pueden ser parte de transacciones y no pueden ser recuperables. Sin embargo, las capas del sistema operativo proveen integridad de archivos y durabilidad. El máximo tamaño soportado de un archivo es de 4 gigabytes.

El administrador de la base de datos debe asegurarse de que el archivo existe y que los procesos de oracle tienen permisos de lectura por parte del sistema operativo.

El tipo BFILE permite soporte de solo lectura para archivos binarios grandes. No se pueden modificar o copiar como un archivo. Oracle provee APIs para acceder a archivos de datos. Las interfaces primarias que se utilizan para acceder a archivos de datos son el paquete DBMS_LOB y la OCI.

TIPO DE DATO BLOB

El tipo de datos BLOB almacena objetos binarios de gran tamaño no estructurados. Los BLOBs pueden ser un flujo de bits sin semántica y pueden almacenar hasta 4 gigabytes de datos binarios.

Tienen alto soporte a transacciones. Los cambios pueden realizarse a través de SQL, el paquete DBMS_LOB o la OCI, participando completamente en transacciones. Las manipulaciones de valores BLOB pueden ser cerradas (commit) o deshechas (roll-back). Notar, sin embargo, que no se puede salvar un apuntador BLOB en una variable de PL/SQL u OCI, para una transacción y entonces utilizar estasvariables para otra transacción.

TIPO DE DATO CLOB y NCLOB

Este tipo de dato almacena datos de caracteres de un solo byte. Ya sean de tamaño fijo o variable, ambos utilizan CHAR para ser inicializados. Pueden almacenar hasta 4 gigabytes de datos de caracteres.

INTRODUCCIÓN AL TRABAJO CON LOBs

Los tipos LOB internos están divididos en los persistentes y los temporales.

LOBs INTERNOS

Como su nombre lo indica, son almacenados dentro de los tablespace de la base de datos de una manera que optimiza espacio y provee acceso eficiente. Utilizan semánticas de copia y participan en el modelo transaccional del servidor. Se pueden recuperar LOBs internos al ocurrir una falla en medio de una transacción y cualquier cambio en el valor interno LOB puede ser cerrado o deshecho. En otras palabras, todas las propiedades ACID que se utilizan en objetos de una base de datos, también se pueden utilizar en LOBs internos.

Los siguientes tres tipos de datos SQL, sirven para definir instancias de LOBs internos:

- BLOB, un valor LOB que se compone de datos binarios no estructurados.
- CLOB, un valor LOB que se compone de caracteres de datos que corresponden al conjunto de caracteres de la base de datos definido para Oracle 8.
- NCLOB, un LOB que su valor es formado de caracteres de datos que corresponden al conjunto de caracteres nacionales definido para Oracle 8.

PAQUETE DBMS_LOB

La utilización de LOBs esta sujeta a algunas restricciones:

No son soportados LOBs distribuidos. Específicamente, esto significa que el usuario no puede usar un localizador remoto en las cláusulas SELECT y WHERE. Esto incluye la utilización de funciones del paquete DBMS_LOB. En adición, referencias a objetos en tablas remotas con o sin atributos LOB, no son permitidas.

Por ejemplo, las siguientes operaciones son invalidas:

- SELECT col_lob from table1@lugar_remoto;
- INSERT INTO tabla_lob select type1.lobattr from table1@lugar_remoto;
- SELECT dbms_lob.getlength(col_lob) from table1@lugar_remoto;

Operaciones validas para columnas LOB en tablas remotas incluyen:

- CREATE TABLE as select * from table1@lugar_remoto;
- INSERT INTO t select * from table1@lugar_remoto;
- UPDATE t set col_lob = (select col_lob from table1@lugar_remoto);
- INSERT INTO table1@lugar_remoto .....
- UPDATE table1@lugar_remoto....
- DELETE table1@lugar_remoto...

Cuando se une un LOB interno con la utilización de cláusulas INSERT/UPDATE, la variable de unión puede ser de tipo SQLT_CHR o SQLT_LBI, pero es limitada a 4k. No se puede un SQLT_LNG a un LOB o a un SQLT_LBI más grande de 4k.

También, los tipos LOBs no son permitidos en los siguientes lugares:

- LOBs no son permitidos en tablas agrupadas y tampoco pueden ser llaves compuestas.
- LOBs no son permitidos en GROUP BY, ORDER BY, SELECT DISTINCT, resumenes y JOINs. Sin embargo, UNION ALL es permitida en tablas con LOBs. UNION, MINUS y SELECT DISTINCT son permitidas en atributos LOB si el tipo de objeto tiene una función MAP u ORDER.
- Los LOBs no pueden ser analizados con las sentencias ANALYSE.....COMPUTE / ESTIMATE STATISTICS.
- LOBs no son permitidos en tablas organizadas con indices particionados, pero son permitidos en tablas organizadas con indices no particionados.
- LOBs no son permitidos en VARRAY.
- NCLOBs no son permitidos como atributos en tipos de objeto, pero son permitidos en metodos.
- Se puede utilizar la columna/atributo LOB en el cuerpo de un trigger con las siguientes condiciones. En general los valores LOB :new y :old son limitados a solo lectura, lo que significa que no se puede escribir al LOB.

UTILIZANDO SQL DML PARA OPERACIONES BÁSICAS EN LOBS.

SQL DML provee operaciones básicas –INSERT, UPDATE, SELECT, DELETE – que permiten hacer cambios a valores completos LOBs internos dentro del ORDBMS Oracle. Para trabajar con partes de LOBs internos, se necesitara utilizar las interfaces que sean desarrolladas para manejar requerimientos más complejos.

Oracle 8 soporta operaciones de solo lectura en LOBs externos. Así, si se necesita actualizar o escribir a LOBs externos, se tienen que desarrollar aplicaciones del lado del cliente acorde a sus necesidades.

Oracle ofrece 6 diferentes ambientes para trabajar con LOBs:

- El lenguaje PL/SQL como resultado del paquete DBMS_LOB:
- El lenguaje C como resultado de la interfaz de llamadas de Oracle (OCI).
- El lenguaje C++ como resultado del precompilador PRO*C/C++.
- El lenguaje COBOL como resultado del precompilador PRO*COBOL
- El lenguaje VISUAL BASIC como resultado de Oracle Objects for OLE (OO40).
- El lenguaje Java como resultado del API JDBC.

UTILIZANDO EL PAQUETE DBMS_LOB PARA TRABAJAR CON LOBS

El paquete DBMS_LOB puede ser utilizado para leer y modificar LOBs internos (persistentes y temporales) ya sea completamente o en solo una parte. Este paquete puede ser utilizado para operaciones de lectura en BFILEs.

Como se describe a mayor detalle en los siguientes puntos, las rutinas DBMS_LOB trabajan basándose en los apuntadores LOB. Para la ejecución exitosa de las rutinas del DBMS_LOB, se debe proveer un apuntador de entrada que represente el LOB existente en los tablespace de la base de datos o en el sistema de archivos externo, antes de invocar la rutina.

Para LOBs internos, se debe definir un objeto DIRECTORY que mapea a un directorio físico valido, conteniendo los LOBs externos que se necesiten acceder. Estos archivos deben existir y debe tener permisos de lectura para los procesos del servidor Oracle.
Si el sistema operativo utiliza nombre de ruta (path) sensible a las mayúsculas o minúsculas, se debe asegurar de especificar el directorio en el formato correcto.

Una vez que el LOB es definido y creado, se debe entonces SELECcionar el apuntador LOB en una variable local LOB de PL/SQL y utilizarla como un parámetro de entrada para el DBMS_LOB, y así acceder al valor LOB. Ejemplos provistos con cada una de las rutinas DBMS_LOB se ilustraran en las siguientes secciones.

Las rutinas que pueden modificar BLOB, CLOB y NCLOB son:

Función/Procedimiento	Descripción
APPEND ()	Agrega el valor LOB a otro LOB.
COPY ()	Copia una porción de un LOB a otro LOB.
ERASE ()	Borra parte de un LOB, iniciando en un offset especifico
LOADFROMFILE ()	Carga un dato BFILE en un LOB interno
TRM ()	Ordena el valor LOB a una longitud corta especifica.
WRITE ()	Escribe datos al LOB especificando un offset.
WRITEAPPEND ()	Escribe datos al final del LOB
COMPARE ()	Compara el valor de dos LOBs
GETCHNKSIZE ()	Obtiene el tamaño del pedazo para leer y escribir
GETLENGTH ()	Obtiene la longitud de el valor LOB.
INSTR()	Regresa la posición de la n-ava ocurrencia in un patrón en el LOB
READ()	Lee datos del LOB, inicando a un offset especifico
SUBSTR()	Regresa parte del valor LOB, iniciando en un especifico offset.

Tabla Rutinas que pueden modificar valores BLOB,CLOB y NCLOB

TRES FORMAS DE CREAR UNA TABLA CONTENIENDO UN LOB

a. Un LOB puede ser una columna en una tabla.
b. Pueden ser atributos de un tipo de objeto.
c. Pueden estar contenidos dentro de una tabla anidada.

En todos los casos, SQL DLL es utilizado, para definir columnas LOB en una tabla y atributos LOB en un tipo de objeto.

INICIALIZAR LOBS INTERNOS A NULL O EMPTY

Se puede colocar un LOB interno – esto es, una columna LOB en una tabla, o un atributo LOB en un tipo de objeto definido por el usuario— a NULL o empty (vacío). Un LOB inicializado a NULL no tiene apuntador. En contraste, un LOB empty almacenado en una tabla es un LOB de longitud cero, que tiene apuntador. Así, si se SELECCiona una columna/atributo empty LOB se obtendrá un apuntador el cual se puede utilizar parea colocar datos en el tipo LOB vía la OCI o las rutinas del DBMS_LOB.

Alternativamente, las columnas LOB, no los atributos, pueden ser inicializados a un valor. Lo que quiere decir – los atributos LOB internos difieren de las columnas LOB internas, en que los atributos LOB no pueden ser inicializados a un valor diferente de NULL o empty.

Se pueden inicializar los LOBs en Multimedia_tab utilizando la siguiente sentencia INSERT:

INSERT INTO Multimedia_tab VALUES (1001, EMPTY_CLOB(), EMPTY_CLOB(), NULL, EMPTY_BLOB(), EMPTY_BLOB(), NULL, NULL, NULL, NULL);

Esta sentencia inicializa el valor de story, flsub, frame y sound a un valor empty (vacío) y coloca photo y music a NULL.

INICIALIZANDO UN LOB A NULL

En caso de querer colocar el valor de un LOB interno a NULL o que se desee utilizar la sentencia INSERT, pero aún no se tenga el valor que se ingresara al LOB, sino hasta después, se deberá hacer con la sentencia SELECT, por ejemplo:

SELECT COUNT (*) FROM Voiced_tab WHERE Recording IS NOT NULL;

Si es que se desea ver todos los segmentos de voz que han sido grabados,

SELECT COUNT (*) FROM Voiced_tab WHERE Recording IS NULL;

Si se desea ver que segmentos de voz, aún no han sido grabados.

Se debe utilizar la sentencia UPDATE para colocar el valor NULL de una columna LOB – a EMPTY_BLOB() / EMPTY_CLOB() o a un valor para LOBs internos o a un nombre de archivo para LOBs externos. No se puede utilizar la OCI o las funciones DBMS_LOB de PL/SQL en un LOB que es NULL. Estas funciones solo trabajan con un apuntador, y si la columna LOB es NULL, esta no tiene apuntador en el renglón.

COLOCANDO UN LOB INTERNO A EMPTY.

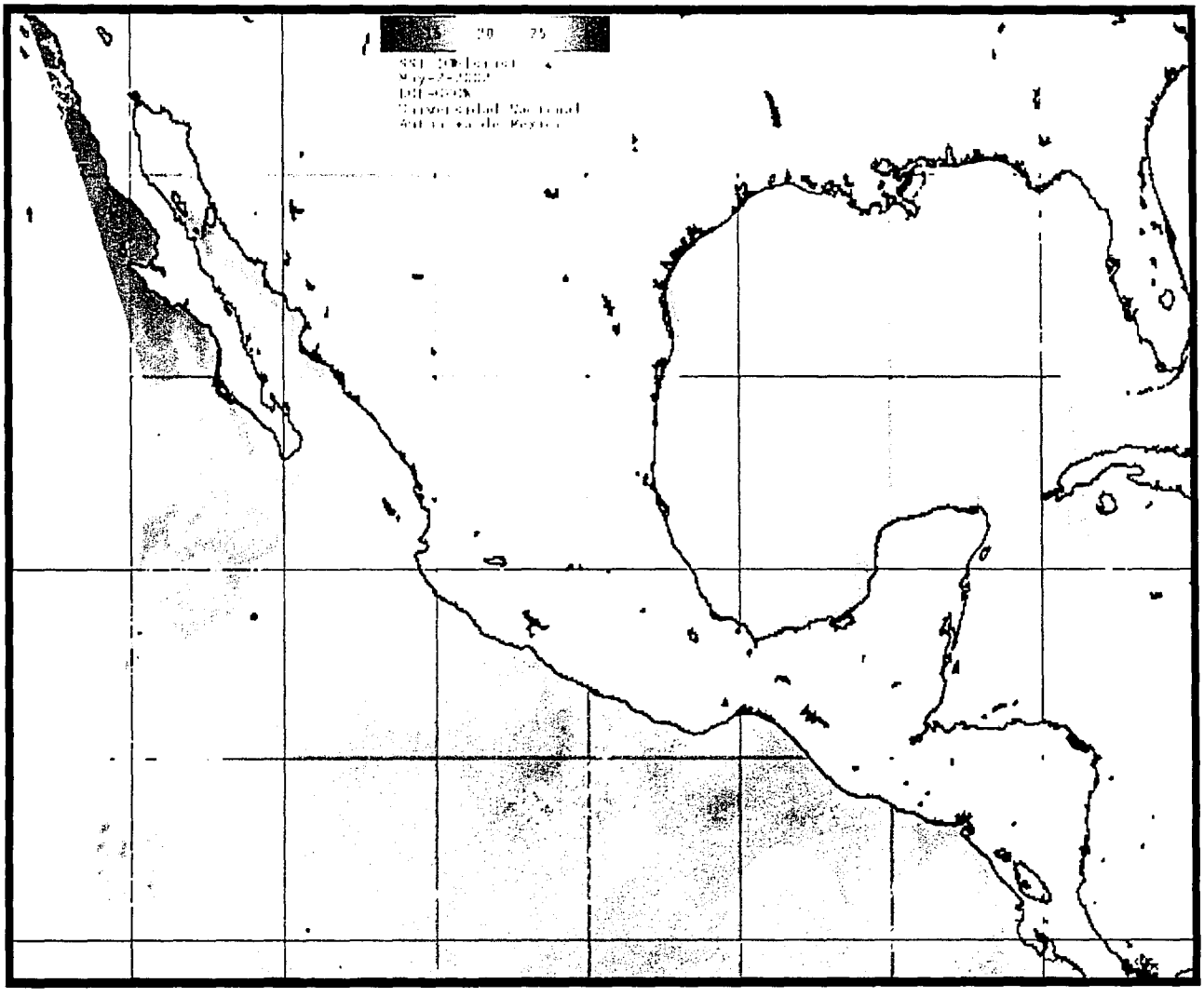
Si no se desea colocar una columna LOB a NULL, otra opción es colocarlo a un valor empty utilizando la función EMPTY_BLOB() / EMPTY_CLOB()en la sentencia INSERT:

```
INSERT INTO una_tabla VALUES (EMPTY_BLOB());
```

Otra forma de hacerlo es la siguiente:

```
DECLARE
        apunt_lob BLOB;
BEGIN
    INSERT INTO una_tabla VALUES (EMPTY_BLOB()) RETURNING blob_col INTO
apunt_lob;
/*      Ahora utilizar el apuntador apunt_lob para llenar el BLOB con datos  */
END;
```

TIPO DE IMAGEN MANEJADA EN LA BASE DE DATOS
(Imagen Diaria Procesada SST)



TIPO DE IMAGEN MANEJADA EN LA BASE DE DATOS
(Imagen Mensual)