



UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

MIDDLEWARE PARA LA EXTRACCION, VALIDACION
E INTEGRACION DE UN DATA WAREHOUSE
BANCARIO

T E S I S
QUE PARA OBTENER EL TITULO DE:
INGENIERO EN COMPUTACION
P R E S E N T A N :
CAROLINA JESSICA BRISEÑO CORTEZ
VERONICA CHAVEZ FLORES
EDUARDO ORDUÑEZ SEGURA
ANDRES RODRIGUEZ GUZMAN
KARINA TORRES REYES

DIRECTOR DE TESIS: M.I. LAURO SANTIAGO CRUZ



CIUDAD UNIVERSITARIA

2000

275827



Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Agradecimientos

A mi padre

Gracias por enseñarme la responsabilidad que implica cada trabajo, por tu cariño y apoyo.

A mi madre

Para ti me faltarían las palabras para describir lo mucho que has hecho por mí, te doy las gracias por ser mi amiga, mi consejera y mi madre. Este trabajo es el fin de un proyecto que tu me ayudaste a realizar y es gracias a ti que ahora soy una profesionalista.

A mi esposo e hijos

A ustedes les doy las gracias por el apoyo, paciencia y amor que me han dado para realizarme como profesionalista, madre y esposa. A ti Antonio además por toda la enseñanza académica y profesional que me has dado.

Jéssica Briseño Cortez.

A mi madre

Por que gracias a ti logré alcanzar esta tan importante meta en mi vida.

Gracias por tu ejemplo de esfuerzo, tenacidad, amor y dedicación para todos los grandes y pequeños detalles de la vida.

Por ese gran apoyo incondicional.....gracias mamá.

A mi hijo Arturo

Para que te sirva como ejemplo de que nunca es tarde para lograr tus objetivos, a pesar de lo difícil que éstos puedan resultar.

A mi hermano Ramiro

Por su gran apoyo y respaldo en los momentos difíciles.

A los compañeros de la facultad

"... Con quienes compartí un largo e importante camino, especialmente a Carmen y Rosina por su valioso apoyo...."

"... A las quienes de manera insistente propiciaron el dar este importante paso en mi vida: Ing. Gilberto Ruíz Durán y C P. Fernando Hernández..."

"... A mis hermanos, amigos y a todas las personas que de una u otra forma me ayudaron a alcanzar este objetivo"

A Olivia

Por su apoyo y comprensión.

Andrés Rodríguez Guzmán.

Gracias a **mis papás**, que me dieron todo lo que fuera necesario (lo que fuera y cuando fuera) para llegar a este punto, que me dieron una familia amorosa que me ha apoyado en todo momento y que siempre lo hará, que me dieron la inconformidad para seguir buscando y que me han dado un **Dios** para agradecerle los papás que tengo.

Gracias a **mis amigos** y compañeros de escuela (especialmente a la banda güevoncia "Victor Vigüevas", "Carlos Perez-oso" y "Pizarro Tapia") que hicieron de la escuela un juego didáctico que volvería a jugar con gusto en su compañía.

Al "Topo de los Cielos (Carlos **csanche**)" que prácticamente me obligó a iniciar este proyecto.

Y especialmente a **mis hermanos** por estar siempre de mi lado sin importar nada.

Eduardo Ordúñez Segura.

Índice

Índice

Prefacio

Capítulo I.- Introducción

1.1.-	¿Qué es un Data Warehouse?	1
1.2.-	Sistemas para Soporte de Decisiones, Sistemas Operacionales y Sistemas de Data Warehouse.	4
1.2.1	Diferencia entre Data Warehouse y bases de datos operacionales (OLTP).	5
1.2.2	Data Warehouse frente a sistemas operacionales.	6
1.2.3	Patrones de Utilización del CPU.	8
1.2.4	Información variable con respecto al tiempo y otras diferencias.	10
1.2.5	La teoría del "Big Bang" contra la teoría de la evolución.	10
1.2.6	Evolución del Ciclo de Vida del desarrollo de sistemas hacia atrás.	12
1.3.-	Factores que Determinan un Proyecto de Data Warehouse Exitoso.	13
1.3.1	Obtener el compromiso de la Dirección.	13
1.3.2	Iniciar con un proyecto controlable.	14
1.3.3	Comunicar claramente expectativas realistas.	16
1.3.4	Asignar a un Gerente de proyecto orientado a usuarios	17
1.3.5	Utilizar métodos comprobados	17
1.3.6	Diseñar con base en preguntas más que en transacciones.	18
1.3.7	Cargar los datos que son necesarios.	19
1.3.8	Definición de la fuente de datos adecuada.	20
1.3.9	Definir claramente temas únicos.	21
1.3.10	Obligar la referencia de todos los aspectos decisivos de los datos.	21
1.3.11	Evaluar el motor de la base de datos	22
1.3.12	Considerar soluciones creadas.	22
1.3.13	Construir con un alto grado de escalabilidad.	22
1.3.14	Construir en un ambiente operativo abierto.	23

Capítulo II.- Metodología de Data Warehouse

2.1	Arquitectura.	24
2.1.1	Planeación de la Arquitectura.	25
2.1.2	Requerimientos.	27
2.1.3	Evaluación de la Arquitectura existente.	31
2.1.4	Diseño de la Arquitectura.	33
2.1.5	Definición de los componentes individuales.	38
2.2	Procesos.	42
2.2.1	Extracción de datos y validación.	42
2.2.2	Características y problemática de la extracción.	42
2.2.3	Características de validación.	49
2.2.4	Distribución de datos y carga	50
2.2.5	Diseño de Bases de Datos.	53
2.2.6	Metadata.	57
2.2.7	Publicación y acceso a los datos.	69

Capítulo III.- Herramientas de Data Warehouse

3.1	Herramientas de apoyo a la construcción Data Warehousing	77
3.2	Herramientas OLAP.	81
3.3	Herramientas para Minería de Datos.	88
3.4	Diversos Proveedores de Herramientas para Data Warehouse.	92

Capítulo IV.- Análisis del Middleware

4.1	Problemática del Sistema Financiero.	111
4.2	Alcance del Proyecto.	113
4.2.1	Objetivo.	113
4.2.2	Características y restricciones.	113
4.3	Arquitectura Global.	119
4.3.1	SA Local con acceso a bases de datos operacionales.	120
4.3.2	SA Local sin acceso a bases de datos operacionales de Entidades Supervisadas.	121
4.3.3	Envío de información directo a la Entidad Normativa utilizando el CLIENTE.	121
4.3.4	Funcionalidad.	122

Capítulo V.- Diseño del Middleware

5.1	Funcionalidad.	127
5.1.1	Extracción de información.	128
5.1.2	Validación de sensibilidad y congruencia.	130
5.1.3	Integración de información.	132
5.2	Definición de Estructuras de datos.	139
5.3	Definición de Eventos y Callbacks.	148
5.4	Definición de Threads Internos.	149
5.5	Definición de Procedimientos Registrados.	150
5.6	Definición de Stored Procedures.	155
5.7	Archivos.	170
5.8	Parámetros en la Línea de Comandos.	171
5.9	Algoritmos.	171

5.9.1	Event handlers.	171
5.9.2	Procedimientos registrados.	177
5.9.3	Thread handlers.	182

Capítulo VI.- Diseño de la Base de Datos Metadata

6.1	Modelo Conceptual.	187
6.1.1	Diseño conceptual de la Metadata.	188
6.2	Modelo Físico.	226
6.2.1	Diseño Físico de la metadata.	226

Capítulo VII.- Implementación del Middleware

7.1	Fase de Implementación.	247
7.2	Fase de Pruebas.	265
7.3	Fase de Mantenimiento.	275
	Conclusiones	279
	Bibliografía.	281
	Apéndice A. Ingeniería de Software.	A1
	Apéndice B. Conceptos Básicos de Bases de Datos.	B1
	Apéndice C. Simbología.	C1
	Apéndice D. Glosario.	D1

Prefacio

En general, para satisfacer las necesidades informáticas de una empresa bancaria, se hace uso de sistemas de transacciones diarias. Este tipo de sistemas cubren las necesidades en cuanto al manejo de la información y la seguridad que sobre la misma se proporciona, sin embargo, debido a la importancia que alcanza la información, actualmente se requiere una gran concentración de la información proveniente de estos sistemas, la cual no necesariamente se encuentra bajo una sola plataforma ni mucho menos bajo un mismo formato. La necesidad actual de obtener esta concentración surge de brindar auxilio eficaz y oportunamente para la toma de decisiones. Estas necesidades llevaron al desarrollo de la arquitectura *DataWarehouse* (**DW**), esta arquitectura es un concepto relativamente nuevo, sin embargo, a pesar de su corta vida, su éxito le dio la difusión suficiente para llegar a las aulas de las principales Universidades del mundo en menos de dos años desde su aparición.

El presente trabajo trata sobre la creación de un componente de la arquitectura *DataWarehouse*; primero describiremos ampliamente el concepto y la arquitectura de *DataWarehouse* y todas sus implicaciones, veremos el enorme alcance que este concepto tiene como una herramienta de apoyo para la toma de decisiones y que como concepto unitario, el **DW** es una gran extensión de actividad informática, durante estos primeros capítulos hemos de sentar las bases para que el desarrollo de la aplicación quede bien fundamentado.

El objetivo del trabajo es "realizar un *middleware* cuya función será automatizar los procesos de extracción, validación e integración de la información enviada por varias entidades a una entidad central, mediante una arquitectura flexible, escalable, configurable y adaptable, apoyada en las definiciones y conceptos del *DataWarehouse*". Estos conceptos serán explicados con más detalle en los capítulos del presente trabajo.

La segunda parte está dirigida al análisis, diseño e implementación del *middleware*. Una de las tareas más complejas fue el análisis y diseño de la *metadata*, que es donde se definen las reglas y el alcance del negocio.

No se pretende plantear una nueva versión de esta arquitectura de reciente creación, sólo se aprovechan las diversas propuestas de la arquitectura del *DataWarehouse* para aplicarlas dentro de la problemática de un sistema financiero. Dicha arquitectura es bastante compleja, ya que involucra varios componentes importantes como las bases de datos de donde se desea extraer la información, herramientas de extracción, herramientas visuales, herramientas para consultas, técnicas de programación, estrategias de desarrollo, y el *middleware* que es el tema de este trabajo de tesis.

Finalmente presentamos la bibliografía consultada y los apéndices generados durante el desarrollo del presente trabajo.

Capítulo 1

Introducción

En este capítulo presentamos la definición básica del *Data Warehouse*, sus objetivos, componentes y características. Así como las diferencias entre *Data Warehouse* y Sistemas Operacionales, factores importantes en la toma de decisiones.

1.1 ¿Qué es un Data Warehouse?

Un *Data Warehouse (DW)* se define como una colección de información corporativa derivada directamente de los sistemas operacionales y de algunos orígenes de datos externos. De acuerdo con W. H. Inmon, quien es considerado como el padre del *DW*: "Un *DW* es un conjunto de datos integrados orientados a una materia, que varían con el tiempo y que no son transitorios, los cuales soportan el proceso de toma de decisiones de una administración."

Como su nombre lo indica, el *DW* es una enorme colección de datos provenientes de sistemas operacionales y otras fuentes, después de aplicarles los procesos de análisis, selección y transferencia de datos. El objetivo principal es que el uso adecuado de esos datos, nos lleve a la obtención de información útil que sirva como soporte fundamental para la toma de decisiones, tarea casi imposible de lograr con la sola utilización de sistemas operacionales. Por lo anterior, un *DW* es considerado como un concepto que viene a resolver los problemas de manejo y uso adecuado de grandes y diversas fuentes de datos.

De acuerdo con algunas organizaciones, el *DW* es una *Arquitectura*. Para otras, es un depósito semánticamente consistente de datos (separados y que no interfieren con los sistemas operacionales y de producción existentes), que llenan por completo los diferentes requerimientos de acceso y reporte de datos de varias fuentes heterogéneas, para soportar la constante necesidad de consultas estructuradas, reportes analíticos y soporte de decisiones, incluyendo datos históricos y adquiridos.

Podemos apreciar una gran divergencia de criterios para establecer una definición precisa de un *DW*, pero existe un claro consenso de que la tecnología del *DW* es un ingrediente esencial en el conjunto de soluciones para el soporte de decisiones de una empresa

Un *DW* organiza y almacena los datos necesarios para la realización de procesamientos analíticos de la información a través de una perspectiva histórica. Es por ello importante mencionar que la meta de "Obtener la información correcta para las personas adecuadas en el momento conveniente, para que tomen decisiones que puedan valer millones de pesos", debe estar presente en la mente de todas las empresas cuyo volumen y manejo de información así se lo exijan.

Para lograr lo anterior habría que resolver algunos problemas que aún no han sido solventados, por ejemplo:

- La integración de datos y metadatos de varias fuentes
- Calidad de información: limpiar y refinar.
- Condensación y adición de datos.
- Sincronización de las fuentes con el *DW*, para asegurar una actualización constante con el *DW*, conforme se crean nuevos datos dentro de las fuentes.
- Aspectos del desempeño relacionados con compartir la misma computadora y plataformas RDBMS, tales como la base de datos y las herramientas del *DW*.
- Administración de los metadatos.

Es importante mencionar que el *DW* no es ni un producto de software ni un motor o tecnología de bases de datos en particular, sino una serie de componentes y procesos que en conjunto forman la *Arquitectura del Data Warehouse*.

La arquitectura de un *DW* es una forma de representar los componentes de una solución *DW* y su interrelación. En este sentido podemos mencionar los siguientes componentes:

- Base de datos operacional.
- Procedimientos de carga.
- Data Warehouse.
- Metadata
- Middleware.
- Sistemas estratégicos / Herramientas de Extracción.

Objetivos del Data Warehouse

En el planteamiento original de Data Warehouse presentado por William H. Inmon y popularizado por IBM como "Information Warehousing", el uso de *DW* como modelo de infraestructura para el soporte a la toma de decisiones tiene cuatro objetivos principales:

Proteger los sistemas de producción del acceso indiscriminado de usuarios ávidos de información, que podrían afectar el rendimiento de estos sistemas.

Proveer un ambiente de información protegido y bien administrado para la toma de decisiones, ya que representa un activo importante para la organización.

Construir un modelo de datos corporativo que permita una estandarización en el manejo de información, tanto de los sistemas de producción como de los sistemas para el soporte a la toma de decisiones.

Mantener independencia entre las aplicaciones de acceso del usuario y la administración de la información, manejándolas como problemáticas independientes.

Estos objetivos marcaron el punto de partida para que se desarrollaran de manera práctica y comercial las diversas estrategias de implantación de este tipo de ambientes.

Características de un Data Warehouse

Formalmente un *DW* es una colección de datos:

- Organizada por Entidades de la Institución.
- Integrada.
- Variable en el tiempo.
- No volátil.

Las características mencionadas en la definición de un *DW* determinan un ambiente muy distinto del ambiente operacional clásico.

Es importante diferenciar los componentes de la arquitectura de las tareas necesarias para llevar adelante un Proyecto *Data Warehousing* y las herramientas que pueden apoyar estas tareas.

La fuente de la mayoría de los datos que componen un *DW* es el ambiente operativo y se tiende a pensar en la existencia de una gran redundancia de datos entre el ambiente operativo y el ambiente del *DW*. De hecho, existe una redundancia mínima de datos entre ambos ambientes debido a:

- Los datos son filtrados cuando pasan de un ambiente a otro. Muchos de los datos existentes en el ámbito operacional nunca pasan al *DW*. Únicamente los datos que resultan necesarios para el procesamiento de toma de decisiones son migrados.
- Los datos son transformados al ingresar al *DW* utilizando los procesos de integración.
- El manejo del tiempo es muy distinto de un ambiente a otro y sólo existe un pequeño traslape entre ambos ambientes.
- El *DW* contiene información sumariada que no existe en el ambiente operacional.

1.2 Sistemas para Soporte de Decisiones, Sistemas Operacionales y Sistemas de Data Warehouse

Desde la década de los 60's los departamentos de sistemas han sido los responsables de los datos transaccionales de las organizaciones. Para ello implementaron sistemas del tipo OLTP (On Line Transactional Processing, Procesamiento de Transacciones en Línea) que permitían recabar, manipular y reflejar los resultados operativos del negocio. El costo de alcanzar estos objetivos ha sido por lo general la falta de flexibilidad de los sistemas obtenidos. Típicamente, los gerentes han sido restringidos a recibir información del negocio en formatos preestablecidos o inflexibles, obtenidos a través de consultas guiadas por menús. El análisis de los datos o reportes personalizados no están soportados por este tipo de sistemas, o son obtenidos luego de un largo proceso, generalmente solicitados al departamento de sistemas. Es por este motivo que, a pesar de la efectividad y eficiencia de estos sistemas en el ámbito operacional, en pocos casos han llegado a ser populares a nivel gerencial. De alguna manera están orientados a contestar preguntas sobre el ayer más que sobre el hoy o el futuro de una organización. Los sistemas OLTP describen muy bien qué, cuándo, a dónde, quién y cómo pero frecuentemente no pueden responder preguntas tales como ¿por qué? ¿O qué ocurrirá si?.

Como resultado, los usuarios finales optaron por el empleo de nuevas metodologías de análisis de datos, tales como las hojas de cálculo, *front-ends* para realizar consultas, etc. reescribiendo la información obtenida de los sistemas transaccionales en la mayoría de los casos, con las desventajas que esto trae aparejado, la posibilidad de errores, el manejo de información sumariada, etc. De cualquier manera, las aplicaciones desarrolladas por usuarios finales no poseen un nivel de análisis de datos profundo, derivado en la mayoría de los casos del desconocimiento de la forma física en que estos sistemas representan los datos. Un usuario final conoce la información en el ámbito transaccional, pero no reconoce en ella tablas, *primary keys*, *foreign keys*, etc. Es por ello que la flexibilidad obtenida mediante estas aplicaciones de usuarios finales ha sido alcanzada a costo de confiabilidad o dificultad de mantenimiento.

En la década de los 80's surgen herramientas orientadas a satisfacer estas necesidades. Estos productos, conocidos como EIS (Executive Information Systems, Sistemas de Información Ejecutiva) o DSS (Decision Support Systems; Sistemas para el Soporte de Decisiones) permiten una mayor flexibilidad en el manejo de los datos, evitan reescribir la información, permiten sumarizar la información a analizar (*Roll-up*) o disgregar la misma llegando a los niveles operativos (*Drill-down*), etc.

Si bien estas herramientas brindaron una solución temporal a los problemas iniciales, al poco tiempo, la dispersión de la captura de datos y el volumen alcanzado por los mismos llevó a la aparición de nuevas tecnologías. Es así como surgen los DW y los sistemas OLAP (On Line Analytical Processing, Procesamiento Analítico en Línea).

Para comprender mejor estas tecnologías se puede realizar una analogía con otras industrias. En la mayoría de las industrias, independientemente de su producto final, se puede dividir el negocio en tres áreas: Producción, Depósito/Distribución y Venta. Cuando

analizamos cómo los datos llegan al usuario final, también podemos distinguir estas tres áreas.

La “producción” de datos es donde capturamos la información y ocurre a nivel de nuestros sistemas operacionales mediante el empleo de OLTP. Estos sistemas están optimizados para almacenar grandes volúmenes de datos a través del tiempo. Para que estos datos estén disponibles a nuestros usuarios finales, deben ser reempacados y almacenados. Es ahí donde surge el “Depósito/Distribuidor” de datos o *Data Warehouse*.

El último paso de la estrategia de llegada de los datos al usuario final es la “Venta” de los mismos. Para ello necesitamos una tecnología que permita localizar los datos que el usuario precisa, con un nivel de detalle variable, que sea capaz de determinar excepciones, realizar análisis de tendencias, modelado y estadísticas, de forma transparente para el usuario. Estos “vendedores” de datos son los llamados sistemas OLAP.

1.2.1 Diferencia entre Data Warehouse y bases de datos operacionales (OLTP)

Un *DW* es diferente de las bases de datos operacionales que soportan las aplicaciones de OLTP. Un *DW* está caracterizado principalmente por lo siguiente:

- **Está orientado a una materia.** Organiza y orienta los datos desde la perspectiva del usuario final. Muchos sistemas operativos organizan sus datos desde la perspectiva de la aplicación, de modo que el acceso de la aplicación a los datos tenga la mayor eficiencia posible. Con frecuencia, la información que está organizada para que una aplicación del negocio la recupere y actualice con facilidad no está organizada necesariamente de modo que un analista con herramientas gráficas inteligentes de consulta pueda formular las preguntas empresariales correctas. Esto se debe al enfoque del diseño de la base de datos —la eficiencia de recuperación y actualización de la aplicación— al momento en que se implementó por primera vez.
- **Administra grandes cantidades de información.** La mayoría de los *DW* contienen información histórica que se retira con frecuencia de los sistemas operacionales porque ya no es necesaria para las aplicaciones operacionales y de producción. Por el volumen de información que un *DW* debe manejar, también debe de ofrecer opciones para la adición y la condensación que clasifican esta inmensa cantidad de datos. En resumen, los actuales usuarios de un *DW* buscan “árboles en el bosque”. Por lo tanto, un *DW* maneja información a diferentes niveles de granularidad. Por la necesidad de administrar toda la información histórica y además los datos actuales, un *DW* es mucho mayor que las bases de datos operacionales.
- **Guarda información en diversos medios de almacenamiento.** Por los volúmenes de información que deben manejarse, un *DW* frecuentemente guarda información en diferentes medios de almacenamiento
- **Comprende múltiples versiones de un esquema de base de datos.** Debido a que el *DW* tiene que guardar información histórica y administrarla, y como la

información histórica ha sido manejada en diferentes momentos por diferentes versiones de esquemas de bases de datos, en ocasiones el *DW* tiene que controlar información originada en organizaciones de bases de datos diferentes.

- **Condensa y agrega información.** Con frecuencia, es muy alto el nivel de detalle de la información guardada por bases de datos operacionales para cualquier toma de decisiones sensata. Un *DW* condensa y agrega la información para presentarla en forma comprensible a las personas. La condensación y adición son esenciales para retroceder y entender la imagen global.
- **Integra y asocia información de muchas fuentes de información.** Debido a que las organizaciones han administrado históricamente sus operaciones utilizando numerosas aplicaciones de *software* y múltiples bases de datos, se requiere de un *DW* para recopilar y organizar en un solo lugar la información que estas aplicaciones han acumulado al paso de los años. Esta es una tarea desafiante por la diversidad de tecnologías de almacenamiento, de técnicas de administración de bases de datos y de la semántica de los datos.

Conforme las organizaciones confían más y más en la disponibilidad de información comienzan a utilizar cada vez más la información del *DW* en sus actividades cotidianas, su disponibilidad comienza a ser cada vez más necesaria, y en el acceso a un *DW*, éste se convierte en un *Sistema de Misión Crítica*, ya que genera dependencia del *DW* para la toma de decisiones importantes.

1.2.2 Data Warehouse frente a sistemas operacionales

Hasta ahora, el propósito principal de los primeros sistemas de base de datos era cumplir las necesidades de los sistemas operacionales, que son, característicamente, de la naturaleza transaccional. Ejemplos clásicos de sistemas operacionales son:

- Libro general de cuentas.
- Pagos de cuentas.
- Gestión financiera.
- Procesamiento de órdenes.
- Entrada de órdenes.
- Inventario.

Los sistemas operacionales, por naturaleza, están principalmente ligados al manejo de una única transacción. Observemos el sistema de un banco: cuando el cliente hace un depósito en su cuenta, el sistema operacional del banco es el responsable de dejar constancia de la transacción para asegurar que aparecerá el correspondiente débito en el registro de la cuenta

El típico sistema operacional trata con una orden, un pedido, un elemento del inventario, etc. Un sistema operacional, por lo general, trata con eventos predefinidos y, debido a la naturaleza de tales eventos, necesita un acceso muy rápido. Cada transacción, normalmente, trabaja con pequeñas cantidades de datos.

La mayor parte del tiempo, el negocio necesita que el sistema operacional no cambie en exceso. La aplicación que graba la información, así como la que controla su acceso a ella, o sea, los informes de un negocio bancario, no cambia mucho con el tiempo. En este tipo de sistemas, la información necesaria cuando un cliente inicia una transacción debe ser corriente. Antes de que un banco conceda un retiro, debe de estar seguro del balance actual del cliente.

Por otro lado, una aplicación de *DW* tiene una naturaleza distinta. Una transacción típica maneja grandes cantidades de datos. Una aplicación de *DW* responde a preguntas como:

- ¿Cuál es el depósito medio por sucursal?
- ¿Qué día de la semana es el más ocupado?
- ¿Qué clientes con un promedio alto en balances no están participando en la actualidad de una cuenta maestra?

Dado que estamos manejando preguntas del tipo ¿Qué pasaría si?, cada petición es única. La interfaz que soporta al usuario final debe tener un diseño flexible. Hay muchas aplicaciones diferentes que acceden a la misma información, cada una de ellas de forma particular. Existen herramientas OLAP que ayudan a componer y procesar este tipo de consultas en el *DW* y en los *datamarts*. Un *datamart* es, generalmente, un subconjunto de *warehouse* con un propósito específico. Por ejemplo, puede que tengamos un subconjunto financiero y un subconjunto de *marketing*, cada uno de ellos diseñado para proporcionar información a una determinada parte del negocio corporativo. Una frase clave en la industria actual es: “¿Qué aproximación se debe tomar cuando se construye un sistema de toma de decisión?”.

Tenemos constancia de dos aproximaciones al proceso de construcción:

- Una aproximación consiste en gastar un tiempo extra en construir primero un núcleo de *DW*, para, luego, usarlo como base para realizar rápidamente muchos *datamarts*. La construcción inicial de esta aproximación dura más, ya que se ha empleado tiempo en analizar las necesidades de datos del *DW* completo, identificando los elementos de información que se usarán en una gran cantidad de mercados. La ventaja consiste en que, una vez construido el *datamart*, ya tenemos los planos del *DW*.
- La otra aproximación consiste en construir primero un *datamart* específico para trabajo en grupo. Esta aproximación pone rápidamente los datos en manos de los usuarios, pero el trabajo para poner la información en el *datamart* puede no ser rentable cuando se mueven dichos datos a un *DW* o cuando se intenta usar datos similares en un *datamart* diferente. Se gana velocidad pero no portabilidad.

Un sistema operacional se optimiza para procesos transaccionales, mientras que un *DW* se optimiza para procesos con amplios conjuntos de datos. Estas diferencias se hacen notorias cuando se realiza un seguimiento del uso de la unidad central de proceso (CPU) en una computadora de *DW* y en otra con un sistema operacional.

1.2.3 Patrones de utilización del CPU

Un *DW* es la pesadilla de un administrador de sistemas desde la perspectiva de utilización del *hardware*. Al contrario de un sistema operacional, donde la utilización del CPU es predecible por naturaleza, en un *DW* la utilización es esporádica. Cuando se inicia un procesamiento de tipo analítico en grandes conjuntos de datos, nunca sabremos cómo o cuándo va a afectar esto a los recursos del sistema. Si nos tomamos la molestia de hacer un seguimiento del CPU en un sistema operacional, observaríamos una gráfica como la mostrada en la figura 1.1, en la que hay actividad durante todo el día, con algunos máximos locales.

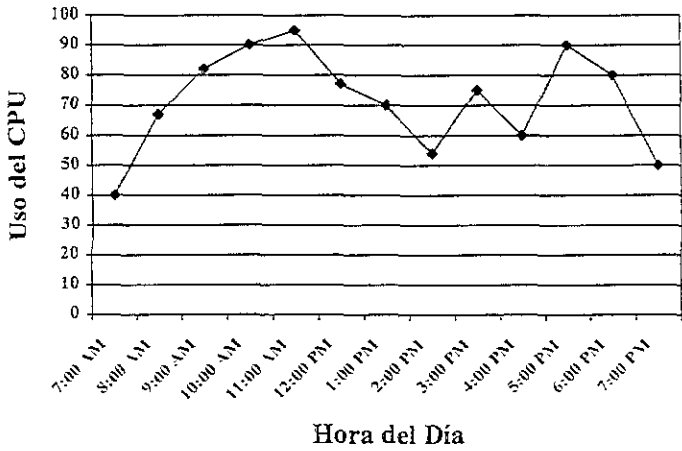


Figura 1.1 Uso de CPU para sistemas OLAP.

Si analizáramos esta gráfica, se podrían predecir las ocurrencias de puntos altos y bajos durante todo el día. Por ejemplo, a la hora de comida, se vería un punto bajo que se iniciaría todos los días como a las 14:30 y duraría hasta las 16:30. Veríamos que cada mañana la zona de mayor actividad se inicia a las 9:00 y llega a lo más alto como a las 10:30. También se podría ver que la actividad comienza a bajar a partir de las 18:00. A las 19:00 se vería que comienza la actividad nocturna del procesamiento por lotes. Claramente se ve que la actividad sigue un patrón predecible durante todo el día.

En el lado opuesto se encuentra *DW*, cuya gráfica de utilización del CPU es muy diferente. Como podemos ver a la figura 1.2, hay una utilización esporádica con vanos altibajos. De hecho se usa al 100%, o no se usa nada. Debido a la complejidad de las

consultas, las cuales determinan la utilización, y dado que las consultas están en relación a las necesidades de gestión que se basan en una prioridad variante, dictada por las siempre cambiantes presiones del mercado, tendremos dificultades a la hora de intentar predecir las tendencias del CPU.

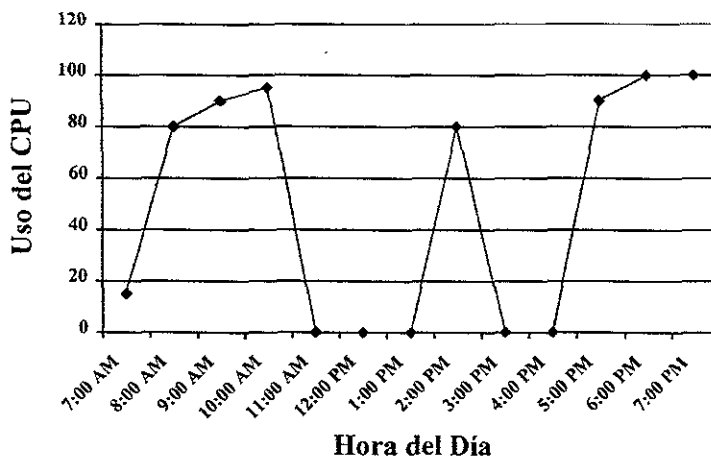


Figura 1.2 Uso de CPU en sistemas DW

Uno de los trabajos más importantes que hay que realizar al construir un DW consiste en intentar comprender cuáles serán las consultas más típicas. Como dice William Inmon: "Es imposible conocer por adelantado cuáles serán las consultas típicas, ya que la gestión desplaza el foco de atención constantemente según las cuestiones relevantes que conciernen al negocio en cualquier punto del tiempo. Pero es posible tener una idea sobre los problemas generales que conciernen a la gestión de manera frecuente. Pero, incluso entendiendo los asuntos de la gestión, seríamos incapaces de predecir el uso que le va a dar la gestión al DW".

Dado que es difícil determinar el patrón de uso del CPU por parte del DW, e incluso si lo determináramos, éste sería inconsistente, lo cual nos lleva a una conclusión importante: no se deben mezclar los sistemas de DW con los sistemas operacionales. Se puede configurar satisfactoriamente una computadora para manejar un entorno de DW o para optimizar la configuración de un sistema operacional, ¡pero no ambos! Una tentación típica es querer que un mismo sistema de computadoras realice ambas actividades, pero la experiencia muestra que en la práctica esto no funciona.

1.2.4 Información variable con respecto al tiempo y otras diferencias

Como ya se ha dicho, la configuración de un sistema de computadoras que se necesita para soportar una operación de DW es distinta a la configuración necesaria para soportar un sistema operacional, así como el típico usuario de un DW es distinto del típico usuario de sistema operacional. Veamos una aplicación de nóminas. ¿Son iguales las necesidades de nóminas que tiene un empleado a las que tiene el vicepresidente de una gran corporación? Aunque los dos necesitan los datos de las nóminas, ambos los requieren por razones completamente diferentes.

La sensibilidad al tiempo de los datos es otra de las grandes diferencias. Un sistema operacional necesita datos corrientes. Esta información, generalmente, no puede ser de hace más de 90 días. En la aplicación de un banco, el empleado del banco está interesado en el balance de la cuenta en el momento actual, y no hace tres días. Por otro lado, el vicepresidente del banco puede estar interesado en las tendencias de los balances de todas las cuentas durante el pasado año. Así, la línea de tiempo del DW es, mucho más grande. De hecho, en un entorno de DW, la línea de tiempo podría variar desde un día hasta varios años.

Las herramientas que se usan para construir un DW son distintas a las utilizadas para construir un sistema operacional. Muchos de los principales expertos en DW creen que intentar usar tecnología CASE en la construcción de un DW es un camino seguro al desastre, y muchas de las herramientas tradicionales asumen que se tiene una idea clara de lo que se está construyendo. En un DW, éste no es el caso, lo que significa que o se meditan las aproximaciones que se toman, o nos arriesgamos a ese desastre.

El proceso evolutivo por el que construimos un DW es diferente al de un sistema operacional. Veamos ahora algunas ideas sobre la evolución de un DW.

1.2.5 La teoría del “Big Bang” contra la teoría de la evolución

La diferencia entre un DW y un sistema operacional ocurren ya desde la fábrica en la que se tejen estas tecnologías. Esto se hace más claro cuando se observa la aproximación de diseño necesaria para construir cada una de las tecnologías. Como veremos, estas aproximaciones son bastante diferentes.

La tecnología de diseño que solemos llamar “Big Bang”, más comúnmente conocida como Ciclo de Vida del Diseño del Sistema (SDLC), es la típica metodología que se usaría para construir un sistema operacional. Se le llama “Big Bang” porque uno se sienta y junta todos los requisitos; cuando el usuario tiene la oportunidad de usar el sistema, se ha avanzado demasiado en el proceso para hacer muchos o incluso algún cambio. Haz el modelo y constrúyelo. Y luego ¡Bang!. Esto existe y se hace, creando un pequeño hueco para posibles cambios del sistema central. La mayoría de los cambios que se permite hacer están relacionados con la presentación de los datos. Para tener una mejor visión de

esta aproximación, la figura 1.3 contrasta el ciclo de vida de un sistema operacional con el de un *DW*.

Sistema operacional, Big Bang	Data Warehouse, Teoría de la evolución
1 Definición de los recursos de usuario.	1 Escuchar al usuario.
2 Análisis y planes basados en definiciones.	2 Implementar una primera visión comercial del <i>DW</i> .
3 Realización de Modelo.	3 Escuchar al usuario
4 Diseño Físico.	4 Desarrollar mecanismos de soporte analítico
5 Programación	5 Escuchar al usuario.
6 Asegurar la calidad y la aceptación del usuario.	6 Hacer más visiones comerciales.
7 Implementación.	7 Escuchar al usuario.
	8 Ir al paso 3

Figura 1.3 Ciclo de vida para Sistemas Operacionales y Data Warehouse.

Quando se construye un sistema operacional se comienza con la fase de especificaciones de requisitos. Esto es fácil de hacer, ya que tenemos unas necesidades claramente definidas. Una transacción bancaria es una transacción bancaria. La responsabilidad principal está en optimizar la capacidad del sistema para permitir el procesamiento de estas operaciones de transacción. Igual que el cajero de una entidad financiera, es necesario hacer constar de manera correcta un depósito realizado por el señor X. Dado que el procesamiento de eventos, o sea, un depósito, está claramente definido en un sistema operacional, la fase de especificaciones de requisitos es ajustada, lo cual es una razón de peso para poder construir de manera satisfactoria un sistema operacional usando la teoría del *Big Bang*.

La diferencia clave en un sistema operacional es que puede desarrollar una lista de requisitos ajustada en un día, aunque éste es un proceso iterativo. Ya que es un sistema operacional, se pueden reunir los requisitos correctos haciendo un análisis adecuado. Una vez realizado el análisis, se puede comenzar a construir el plan. Es ahora cuando se modela el sistema operacional. Existe una alta probabilidad que el modelo que se desarrolla se convierta en el producto final, a no ser que se descubra que se cometió un gran error en la fase de descubrimiento. Este proceso continúa hasta que el sistema está completo, ¡*Bang!*, ya lo tenemos.

Comparemos esto con una aproximación heurística. Heurística se refiere a un proceso en el que la salida de una etapa corresponde a la entrada de la siguiente.

En un proyecto *DW* es difícil, si no imposible, hacer otra cosa que no sea un análisis heurístico. Con esto se quiere decir que nadie pone atención a lo que queremos en un *DW*. La experiencia ha demostrado que el proceso es absolutamente diferente.

1.2.6 Evolución del ciclo de vida del desarrollo de sistemas hacia atrás

Al ciclo de vida del desarrollo de sistemas hacia atrás conocido como BSDLC (Backwards System Development Life Cycle, Ciclo de Vida del Desarrollo hacia atrás) de *DW* se le suele llamar "La teoría de la Evolución". En la figura 1.3, las etapas parecen algo que va de arriba hacia abajo comparado con el ciclo de vida de los sistemas operacionales. Debido a que los usuarios finales no conocen todavía como analizarán los datos, es imposible comenzar con una especificación de requisitos.

Recordemos que un *DW* es un repositorio de información. Su objetivo es ser lo bastante flexible como para soportar los cambios en las necesidades de un negocio, de tal forma que, cuando se propongan preguntas tipo ¿Qué pasaría si...?, Se puede profundizar más y más en el *warehouse*. Cada pregunta descompone la información de manera distinta. Si se hace correctamente, se encontrarán las respuestas buscadas.

Cuanto más rápido se pueda poner un mercado a funcionar con el usuario final, antes se podrá escoger un *datamart*. Un *datamart* es un subconjunto de un *DW* que se extrae para satisfacer las necesidades de una clase particular de usuario. Por ejemplo, sería útil establecer un mercado de *marketing*, un mercado financiero, etc. Luego, se deja que los usuarios que pertenecen a una de estas clases trabajen con un determinado subconjunto de datos. Estas cuestiones descubrirán cualquier hueco que haya en los conjuntos de datos.

Al principio del proceso de construcción de un *DW*, el mejor objetivo que se puede esperar obtener es comprender los intereses generales de la gestión. En un sistema financiero, la gestión tendrá interés en saber por qué se tardan 90 días en pagar las facturas. Se pueden comparar los tipos de clientes y su correlación con las facturas que se están pagando, y también se pueden comparar los segmentos en una industria y su correlación con las facturas que se están pagando.

Por lo anterior llegamos a la conclusión que: La clave para tener éxito en una iniciativa de *DW* está en poner los *datamarts* en su sitio de forma rápida, aunque esto signifique que la iniciativa de *DW* se alargue más. Hasta que los usuarios empiecen a analizar los datos, seremos incapaces de determinar los conjuntos principales de datos que faltan.

Si los usuarios finales no pueden verlo, probarlo, tocarlo o intentarlo, los desarrolladores del sistema no hacen nada. Muchos proyectos de *DW* fallan por que los empleados del sistema intentan crear un *warehouse* demasiado poblado antes de presentarlo a los usuarios finales, éste es un gran error. Se puede pasar mucho tiempo juntando y filtrando los datos sin presentarlos primero al usuario final. *Filtrar* los datos es el proceso de purificar los datos en un *warehouse* o en un negocio,

En resumen, lo que hace diferente al *DW* de las demás aplicaciones operacionales de negocios es el modo como se organizan los datos, además de la cantidad de datos que se pueden almacenar. Los datos de transacciones que apoyan una aplicación operacional están organizados para un rápido almacenamiento y recuperación de maneras *predecibles* y *bien conocidas*. La *información dentro del DW* está organizada para sustentar una gran variedad de tareas de análisis, las rutas de acceso que se toman a fin de recuperar datos para tomar decisiones y sustentarlás no pueden predecirse o controlarse con anticipación.

1.3 Factores que Determinan un Proyecto de Data Warehouse Exitoso

Data Warehousing es una tecnología vital que revoluciona el modo en que las empresas accesan a su información y la utilizan para crear soluciones de negocios, sin embargo, *DW* no es un hechizo que al leerse actuará y se aplicará por sí solo; es muy importante planear bien las acciones de la implementación antes de iniciar un proyecto *DW* (más que cualquier otro) y estar muy conscientes de nuestras capacidades y de los alcances del proyecto *DW*.

Un *DW* almacena datos históricos que se obtienen de los sistemas operacionales y de fuentes externas de datos, todos estos datos se limpian para quitar inconsistencias y se integran para crear nuevas bases de datos orientadas por temas, que son más adecuadas para los *front-ends*. Debido a que el *DW* está separado de las bases de datos operacionales, los procesos informativos se pueden descargar en sistemas abiertos más flexibles y eficientes. De esta manera se permite que el usuario maneje una buena cantidad de la información por sí solo, usando herramientas gráficas de acceso y análisis de datos. Así sus preguntas (*queries*) no afectan a los Sistemas Operacionales, y los Datos Operacionales quedan a salvo de las operaciones de los usuarios del *DW*, a la vez que éstos últimos pueden navegar en la información en un sistema interactivo y *ad hoc*.

Al igual que en toda tecnología, la implementación, la aplicación y los métodos adecuados determinan que se tenga éxito o no en el proyecto, además, tal como lo indican los estándares de calidad para productos y servicios, la planificación es un aspecto muy importante a considerar en el desarrollo de un proyecto de *DW* debido a que la mayoría de las decisiones asociadas con el diseño, implementación y entregas se determinan en la planeación del proyecto. Sin embargo, hay muchos otros aspectos (tal vez miles) que se deben considerar para que la implementación del *DW* sea exitosa, a continuación veremos algunos de ellos.

1.3.1 Obtener el compromiso de la Dirección

Tener el soporte adecuado es el punto más crítico para el éxito de un proyecto *DW*, de nada sirve diseñar y planificar el mejor proyecto si no se tiene el soporte adecuado, digamos el patrocinio adecuado, tenemos que buscar quien respalde el proyecto dentro de la empresa, esta persona tiene que ser un ejecutivo de alto rango dentro de la compañía; por ejemplo, el Vicepresidente de Ventas, de Mercadotecnia, de Manufactura, de Operaciones, etc. Esta persona se convertirá en el logro de una credibilidad más allá de un proyecto de desarrollo más, le dará al proyecto la visión estratégica de negocios necesaria al *DW* y el respeto de toda la organización.

Este soporte nos ayuda a obtener información que se pueda considerar estratégica, o a acelerar la obtención de información y a obtener una visión general de lo que la compañía puede considerar como estratégico.

Con base en la importancia estratégica del patrocinador, es importante notar que no siempre se tendrá quien respalde el proyecto con el que se pueda tratar directamente, sino que seguramente el designará o nos ayudará a designar un Comité Controlador que

le represente. El Comité Controlador o el propio Patrocinador serán quienes den el soporte al proyecto para llegar a un final exitoso.

Igual como el Patrocinador nos ayudará a obtener información y cooperación de los usuarios, también nos servirá en aspectos relacionados con las finanzas del proyecto, debido a que los proyectos *DW* normalmente no suelen demostrar beneficios en corto plazo. Entonces, el Protector elegido nos puede ayudar o incluso pelear los recursos financieros que el proyecto requiere. Y nos podrá ayudar a realizar el plan financiero de desarrollo correcto para justificar los costos del proyecto *DW* y así aseguramos que no nos van a faltar recursos tales como equipo, lugar, *software*, etc.

Por observación y experiencia, podemos asegurar que la falta de recursos en cualquier punto del desarrollo de una aplicación resulta en el fracaso de la misma, si consideramos que un proyecto *DW* no sólo es caro, sino que es lento, el factor del patrocinio comprometido de un Alto Dirigente (para que no lo cambien a mitad del proyecto) puede ser el más peligroso.

1.3.2 Iniciar con un proyecto controlable

De ninguna manera se recomienda iniciar un proyecto *DW* como tal como primer esfuerzo, esta pretensión es demasiado para cualquier equipo de desarrollo, y casi seguramente no se terminará jamás.

Para ejemplificar, recordemos que, generalmente, un *DW* se aplica para empresas que controlan y dependen de cantidades de información medibles en *gigabytes*, *gb*, (como unidad mínima), generalmente transnacionales, y además, en un *DW*, la información se maneja en forma histórica.

En este contexto, pensemos en la tarea de análisis (solamente) de requerimientos para una compañía con una sola filial fuera del país, y con dos dentro de la república, que controlan almacenes, subdistribuidores, proveedores, nóminas, seguros y su contabilidad (para no ahondar), que desean controlar su información en forma histórica, compararla con la información que se pueda conseguir de proveedores de información y finalmente, utilizar todos estos recursos o fuentes de datos para alimentar alguna aplicación que además de ser amigable al usuario (que en general serán de nivel gerencial o superior) les debe ayudar a tomar decisiones estratégicas para la empresa.

Aunque para muchos el párrafo anterior es suficientemente claro, tratemos de imaginar la cantidad de tiempo, recursos humanos y recursos financieros que necesitaríamos para lanzar este proyecto completo (análisis, diseño, implementación, documentación, pruebas, capacitación, mantenimiento) y finalmente, busquemos por todos lados una empresa que estuviera dispuesta a realizar la inversión tan grande y con resultados a tan largo plazo.

Alguien de pensamiento rápido podría preguntarse ¿Entonces, para qué sirve todo esto si no se puede aplicar por ser una inversión muy poco viable? Y con justa razón, aunque seguramente ha habido en la historia compañías y corporaciones que puedan darse estos lujos, sobre todo instituciones gubernamentales, pero ahora los expertos nos recomiendan que apliquemos la táctica "divide y vencerás" para iniciar un proyecto *DW*, es decir, dividir la organización por áreas o temas de interés para la toma de decisiones gerenciales.

Esto resulta muy conveniente, sobre todo, porque la base de datos *DW* está orientada a estos temas (áreas, materias) de interés gerencial.

Y no sólo eso, sino que al implementar un *DW* virtual (*DW* para el tema, ó *Subject DW*), podemos iniciar el análisis, desarrollo de métodos, capacitación, generación de código, etc. Todo lo cual nos va a servir durante el desarrollo de los siguientes *DW* virtuales, es decir, al dividir esta gran aplicación en lo que para los Sistemas Operacionales (SO) serían las áreas funcionales, obtenemos los mismos beneficios que en los SOs, nos familiarizamos con las áreas de la empresa, desarrollamos métodos de prueba, de implantación, escribimos código, conocemos la logística de las diferentes operaciones de la empresa, nos relacionamos con el personal, etc. Así, cubrimos cuatro necesidades, iniciamos con un proyecto que podemos terminar exitosamente, hacemos una parte de la base de datos del *DW* completo y además, nos ahorramos tiempo de programación del desarrollo de los subsecuentes *DW* virtuales y finalmente, esta división nos da la oportunidad de elegir un área específica de la organización que nos dé ventajas.

Si analizamos la organización un poco, podremos observar que existen áreas funcionales que si bien no son más importantes que las demás, si manejan procesos críticos que serían capaces de colapsar la organización por algún periodo de tiempo, de tal manera que si por alguna razón se nos ocurriera iniciar el proyecto en una de estas áreas y falláramos, generaríamos muchos problemas para la organización.

Por lo tanto, debemos, además, elegir con cuidado un área de la organización que no sea de misión crítica, y de preferencia, que nos dé buen prestigio cuando terminemos nuestro proyecto *DW* con éxito. En la gráfica de la figura 1.4 podemos ver esto con mayor claridad si consideramos el eje vertical como la media del beneficio que representa para la compañía el avance del sector, y en el eje horizontal el riesgo que involucra fallar en la implantación en el sector respectivo de la compañía.

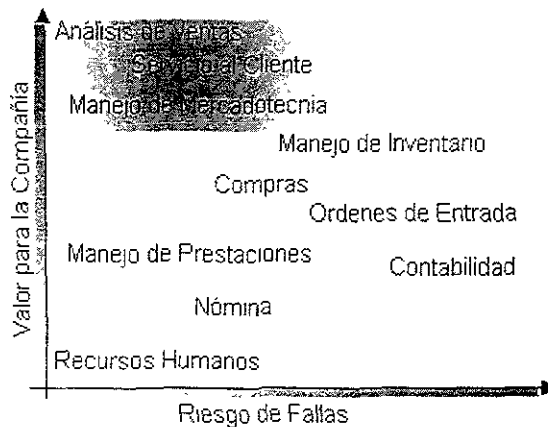


Figura 1.4 Evaluación del Sector a desarrollar.

Los costos iniciales serán altos para el proyecto, pero con las ventajas descritas, éstos irán disminuyendo como efecto de reutilizar códigos, formatos, estándares, etc. hasta estabilizarse en un punto bajo, como vemos en la gráfica de la figura 1.5

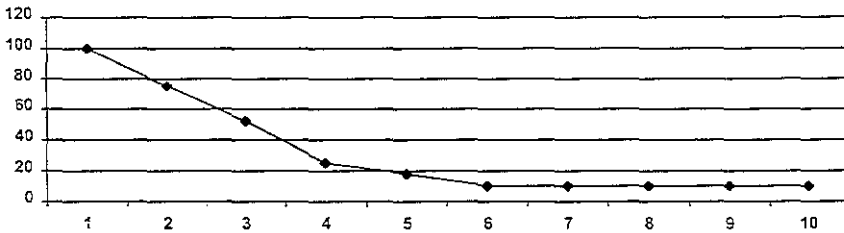


Figura 1.5 Porcentaje del costo de desarrollo sobre el tiempo

Reutilizar los elementos para el desarrollo de nuevos *DW* orientados a temas disminuye los costos de producción hasta que nos encontremos con un área de negocios a la que parecen no ajustarse las *DW* anteriores, lo cuál obliga a que se vuelvan a realizar esfuerzos de análisis y desarrollo para adecuar los *DW* desarrollados anteriormente con el área de negocios que se está analizando.

Además, habría que realizar un esfuerzo especial para verificar que los cambios que se lleguen a realizar en los *DW* no afecten a las áreas ya desarrolladas.

Esto provocaría un pico al final de la gráfica anterior, además de que el tiempo en nuestros proyectos se tendría que alargar con un nuevo ciclo de desarrollo.

Por lo tanto, la estrategia que se recomienda para el desarrollo de *DW* es dividir el *DW* corporativo en *DW* virtuales que se irán integrando conforme se van terminando, y usar una metodología que se enfoque en el desarrollo de ciclos de desarrollo pequeños y entregas rápidas para los *DW* orientados a Temas, y en reutilizar entidades de datos. Esto hace que los usuarios tengan entregas con mayor velocidad, y que el equipo del proyecto aumente su experiencia con base en los ciclos más rápidos de revisión.

1.3.3 Comunicar claramente expectativas realistas

De entre las cosas que se involucran al realizar un proyecto (el que sea), una muy importante es la comunicación. A nadie le gustan las sorpresas cuando éstas involucran excesos, sobre todo en tiempos y presupuestos, y probablemente, es todavía peor cuando la sorpresa es que la funcionalidad está por debajo de lo esperado. La comunicación clara y precisa nos ayuda a no comprometernos en tiempos ni capacidad del sistema que no sean reales.

Una diferencia importante en el desarrollo de un proyecto *DW* es que los clientes no son como los que siempre tratamos, sino que son de un nivel gerencial hacia arriba, para ellos, los tiempos, la funcionalidad y el presupuesto deben ser manejados y comunicados a los clientes (y al Patrocinador o su comité) con toda claridad y oportunidad.

Es importante asegurarse de que no se sobrestimen las capacidades del *DW*, finalmente, las computadoras son sólo máquinas, y dependen de un cierto grado de inteligencia del usuario, se espera que el *DW* le ayudará al cliente a realizar mejores decisiones, pero no hay garantía de ello, menos se debe esperar que el *DW* diga que decisiones se deben tomar; el *DW* sólo da a los usuarios mejor información para que ellos puedan tomar sus decisiones.

Si todo lo anterior se comunica con claridad y oportunidad, evitaremos sorpresas, decepciones y obtendremos un estatus de éxito conforme vayamos desarrollando nuestro proyecto *DW*.

1.3.4 Asignar un Gerente de proyecto orientado a usuarios

La selección de un Gerente de Proyecto tiene que hacerse con un poco más de cuidado, si pensamos que cualquier líder de proyecto podrá llevar a buen término un proyecto *DW* tenemos problemas.

El gerente de proyecto debe ser capaz de trabajar en el proyecto desde afuera, es decir, tiene que ser capaz de colocarse en el lugar del usuario, con completo entendimiento de cómo se va a ver y sentir el producto (lo que se entrega). Porque es desde esta perspectiva desde donde podrá guiar correctamente a su equipo para que no haya desviaciones durante el desarrollo, a menos, que éstas representen una mejora al producto que se traduzca en posibilidades de éxito para el usuario del *DW*.

Los gerentes de proyecto orientados a tecnología (tecnócratas) tienden a perder de vista el final del hilo, el producto en que el usuario se apoyará para tomar mejores decisiones, a medida que aparecen nuevas tecnologías en el mercado desean establecer un nuevo plan (o por lo menos cambiar algunas fases) para ajustarse a las nuevas tecnologías.

Para entregar un *DW* exitoso, el gerente de proyecto debe entender con claridad lo que el usuario desea, tiene que ser capaz de entender las lógicas del negocio en cuestión y algunos de los procesos mentales de los usuarios y tiene que entender el tipo de negocio en el que la compañía trata de ser la mejor.

Y finalmente, el gerente de proyecto tiene que interactuar frecuentemente con los usuarios, por lo que su capacidad de comunicar es esencial y es altamente deseable una alta capacidad de desarrollo de empatía y socialización. Arriesgarse con un líder de proyecto que no cumpla todos estos requisitos es como jugar a la Ruleta Rusa con nuestro *DW*.

1.3.5 Utilizar métodos comprobados

Cuando se iniciaron los desarrollos en arquitectura Cliente-Servidor observamos un fenómeno muy especial, los profesionales del desarrollo de *software* comenzaron a perder la memoria, en esa época, todo lo que se había hecho fue de alguna u otra manera eliminado por obsolescencia, con excepción de los ciclos de desarrollo de sistemas, manejo de proyectos y las políticas de gobernabilidad de estas tareas, sin embargo, fue

un largo camino para llegar hasta ese punto, y muchas lecciones valiosas se tuvieron que aprender en el camino, éstas deben formar las bases donde se construyan los sistemas futuros.

Así, aunque el *DW* puede ser un nuevo tipo de sistema, las tareas a las que hay que enfrentarse no son muy diferentes de las que había en el pasado y finalmente, el *DW* no es otra cosa que una aplicación. A través de décadas de desarrollo de sistemas se han aprendido varias técnicas para desarrollo y manejo de sistemas de información. Utilizar estas técnicas nos ayuda a elevar la probabilidad de que el proyecto *DW* llegue a buen término, es necesario tomar todas esas técnicas que funcionaron para diferentes plataformas y para tantos sistemas diferentes y fusionarlas en un nuevo estándar que nos ayude a *entregar sistemas de calidad a nuestros usuarios*.

Entre otros procesos del *DW*, hay que adoptar políticas, procedimientos y estándares para tener una base de comparación para lograr el tamaño, contenido, valor y calidad de una actividad. Entre los artículos que deben tener políticas, procedimientos y estándares documentados están:

- Manejo del proyecto
- Planes de desarrollo
- Documentos de diseño
- Técnicas de modelado
- Técnicas de desarrollo de prototipos
- Planes de pruebas
- Planes de control de calidad
- Procesos de revisión y auditoría
- Pruebas de funcionalidad
- Control de cambios (modificaciones)
- Reporte y rastreo de problemas
- Control de configuraciones

Todo esto es especialmente funcional en aquellas situaciones en donde muchas personas tienen que cooperar usando diferentes productos, y coexisten diferentes herramientas, no se puede pensar que todo esto funcione si se permite que los gerentes y desarrolladores trabajen a sus ritmos, con sus propios estándares y definiendo sus requerimientos individualmente. En cambio, es fácil deducir que la capacidad de lograr desarrollos *DW* se incrementará si todos conocen un método uniforme de realizar las tareas.

1.3.6 Diseñar con base en preguntas más que en transacciones

Las técnicas de modelado de procesos, tan utilizadas en el desarrollo de sistemas operacionales no resultan muy útiles cuando se trata de desarrollo de *DW*.

Un *DW* es un sistema guiado por datos y guiado por cuestionarios, en donde el desarrollo se debe enfocar al modelado de datos, enfocándose principalmente a que el producto que se entrega debe ser fácil de entender y rápido de navegar. Es necesario considerar en que el usuario no va a cambiar su forma de pensar, por lo tanto, no es posible hacer que las preguntas (*queries*) queden bien definidas y en algunos casos no se puede costear el

ajuste de cada pregunta que el usuario enviará al sistema, por lo tanto el sistema debe tener cierta inteligencia implícita para adaptarse a esa forma de actuar e inclusive fomentarla.

Por lo tanto, es necesario que se tenga muy presente la organización y disposición de los datos, los índices, el uso de memoria y los patrones de acceso a preguntas (*query access patterns*). Esto significa romper muchas de las reglas que hemos aprendido en el transcurso de la historia del modelado de datos, específicamente, el problema de la desnormalización se convertirá ahora en un gran auxiliar.

DW ha ido añadiendo desnormalización a sus técnicas de modelado e implementaciones físicas porque esto mejora la posibilidad que tiene el usuario de navegar por los datos e históricos del negocio, sin tener que recurrir a la intervención de un experto, así se puede explicar mejor por qué la redundancia en estos datos o desnormalización es una técnica que ayuda a que logremos facilidad de uso y eficiencia en el procesamiento de queries en un *DW*.

1.3.7 Cargar los datos que son necesarios

Hay que distinguir entre una técnica de desnormalización y un ingreso indiscriminado de datos en la base *DW*. Si se le pregunta a un usuario que datos necesita tener en el sistema, seguramente responderá que todos. Esto es debido a que el sentimiento generalizado de los usuarios es que los sistemas les han tenido aislados de los datos que tanto necesitan para realizar sus labores.

Muchos *DWs* se iniciaron ante esta ansiedad de datos de los usuarios, al cambiar la filosofía del departamento de sistemas para *“démosle todo lo que piden, y luego vemos que es lo que usan”*. No debemos descuidar que finalmente, sólo son usuarios, y por ellos somos los profesionales en sistemas.

Debemos cuidar qué datos entran al *DW*, los datos no deben entrar en el *DW* sin justificación plena, hay que estar atentos de la tenue línea que divide el “querer” y “necesitar”. El uso operacional de datos, o su existencia en una base de datos no es justificación para que éstos se incluyan en el *DW*.

Posiblemente, si las computadoras fuesen comparativamente mucho más capaces en todos sentidos, podría diseñarse el *DW* para que almacenara toda esa información, pero hay limitantes de todo tipo, y la inclusión indiscriminada de datos (o el acceso indiscriminado a búsquedas) provoca un muy notorio alentamiento en los procesos del *DW*. Por lo tanto, hay que ser muy escrupuloso en cuanto a qué datos vamos a dejar entrar al *DW* y que datos no.

Una forma de acercarse a este requerimiento es entrevistarse con los usuarios, luego realizar un diseño y volver a presentarlo a usuario, esta vez para validarlo, aclarando las perspectivas de información (no de datos) que el modelo le puede ofrecer, luego hay que darle la oportunidad de que retroalimente este modelo. Y cuando lo haga, conviene que se le haga clasificar los datos en cuatro categorías: necesario, importante, bueno e innecesario.

El diseño inadecuado con mucha información que no corresponde (externa) deja al *DW* inflexible e inflado, con datos sin significado que ocupan lugar, tienen relaciones, alentan los procesos y deben recibir mantenimiento. Estos datos, además, enterran los datos importantes, lo cuál seguramente pondrá a los usuarios en contra nuestra y del *DW*.

1.3.8 Definición de la fuente de datos adecuada

Normalmente vamos a encontrar que las organizaciones tienen la información dispersa en muchas partes, esto es normal y consecuencia de la rigidez de los sistemas de información que orilla a los usuarios a crear y conservar sus propias bases de datos hechas a su medida y modo.

Por lo tanto, cuando se escarba para obtener las fuentes de información, se puede encontrar, en el mejor de los casos, con diversas formas de tablas hechas en hojas de cálculo. Sin embargo, lo normal al desarrollar *DWs* es que las organizaciones han crecido mucho, algunas veces por medio de fusiones, esto nos lleva a una organización con diversos sistemas de información, y con sus subsecuentes bases de datos dispersas.

Aunque a pesar de esto las empresas se han logrado coordinar para sobrevivir e incluso crecer, sus sistemas de información aún se encuentran enfocados a algo menos que toda la organización, y además, no hay una arquitectura en común.

Esto supone un problema para quienes deben analizar el negocio como un todo, integralmente, porque tienen que ver los datos a través de diferentes fronteras funcionales, e incluso a través de diferentes fronteras de arquitectura. Y por si esto fuera poco, muchas veces hay información vital en proveedores externos, información que sirve para comparar contra la información de la propia organización y poder determinar tendencias, ventajas y problemáticas, algunos ejemplos de estas fuentes externas son:

- Reportes de marcas, compras e investigación de mercados.
- Reportes demográficos y crediticios.
- Diarios de negocios populares.
- Revistas de la industria.
- Reportes tecnológicos.

Algunas de las complicaciones especiales que tendrán estas fuentes externas son: Referencias legales de las fuentes de datos, Uso ordenado de las entradas de datos, Frecuencia de actualización y el Formato no estructurado de la información.

Hay que rastrear la información desde su punto de captura hasta la fuente original de la misma y almacenar las referencias en un repositorio de *metadatos*. Metadatos significa datos sobre los datos, en ella se describe la estructura, el contenido y el origen de los datos contenidos en un *DW*. De esta manera, es un elemento primordial en donde todos los datos internos o externos quedarán definidos y registrados para su uso en un *DW*. Esto ayudará a los miembros del equipo de desarrollo y usuarios del *DW* a través de los procesos de la transformación a las áreas de acceso de datos del *DW*. Podemos considerar a los *metadatos* como equivalentes metafóricos de un catálogo de libros en una biblioteca.

1.3.9 Definir claramente temas únicos

Como se expuso anteriormente, un *DW* se debe dividir en temas únicos, claramente diferenciados entre sí y que definan áreas clave de la organización en cuestión.

Cada área y cada indicador de los negocios debe quedar contenido y biunívocamente definido en un *DW* en estándares que se deben manejar en toda la organización y que deben estar organizados sobre una dimensión estandarizada del tiempo.

Uno de los ejercicios más interesantes del desarrollo del *DW* será realizar las definiciones de temas de alto nivel e indicadores estratégicos del negocio, junto con su información aledaña. El reto será definir y documentar estas áreas e indicadores en términos tales que sean uniformes para toda la organización. No debe haber definiciones que resulten confusas, o que permitan traslapes de tiempo.

Como dijimos anteriormente, estas definiciones se utilizarán durante el desarrollo de los *DW* orientados a temas (áreas de la empresa), y se debe considerar en el plan de desarrollo un tiempo para las fases de definición de temas e índices estratégicos de la empresa.

1.3.10 Obligar la referencia de todos los aspectos decisivos de los datos

Uno de los aspectos más decepcionantes de la información es la manipulación y la poca credibilidad de los datos, en las primeras implementaciones de los sistemas de soporte de decisiones involucraban la extracción de grandes cantidades de datos de mainframes para colocarlas en hojas extendidas (spreadsheets). Entonces los usuarios manipulaban los datos para justificar sus causas, las colocaban en una gráfica y se dirigían a las juntas.

De esta manera la mayor parte del tiempo de la reunión se perdía en determinar la procedencia y confiabilidad de los datos que se presentaban. A tal grado que en algún caso específico, se logró la venta de un proyecto *DW* con sólo presentar en la reunión los tiempos que se lograría ahorrar en reuniones al implementar el *DW*.

El *DW* permite evitar tales desviaciones y juegos de números tan comunes en los negocios actuales. Si se logran las características de un *DW* completo, se tendrá un repositorio de datos fidedignos. Los reportes estarán documentados respecto a sus fuentes, además, al haber implementado completamente el *DW* todas las fuentes de datos para los reportes serán el propio *DW*, y si no, se debe poder obtener la fuente a partir de un reporte, que debe ser fácil de encontrar asociado a los datos; y si esto es muy importante para el negocio, el *DW* se debe poder expandir para incluir la nueva fuente.

Entre otras cosas, es conveniente que se diseñen e implanten políticas en la empresa para que cada fuente de información utilizada sea documentada, y que los datos que de ella provengan sean identificados con la fuente, y finalmente, que los datos de cada reporte que se presente tengan indicada el origen de los datos.

1.3.11 Evaluar el motor de la base de datos

Al igual que cuando queremos competir en autos, es muy importante tratar de conocer las capacidades y limitaciones de nuestro motor, en el caso del *DW* hay que considerar varios aspectos interesantes:

Primero, hay que recordar que las bases de datos dependen de actualizaciones regulares de la información, por lo tanto hay que asegurarnos que el tiempo de respuesta a las preguntas (*queries*) sea aceptable.

Como mencionamos anteriormente, las bases de datos *DW* tienden a crecer sin medida, por lo tanto, hay que considerar que los procedimientos del producto (como el *backup*) resulten en un grado suficiente de eficiencia, además, hay que tener presente siempre que la base *DW* siempre será más grande que los datos sin procesar (*raw data*), estadísticamente, el triple del tamaño, y posteriormente, con sumarios de tablas, se vuelve a duplicar.

Y finalmente, cuando una base de datos *DW* sobrepasa los 100 gb es necesario evaluar las características de disponibilidad de datos de nuestros productos.

1.3.12 Considerar soluciones creadas

- 1) Hay herramientas de extracción de datos que nos permiten vaciar los sistemas operacionales en el *DW*.
- 2) Las Bases de Datos Relacionales proveen el manejo para las bases de datos subyacentes del *DW*.
- 3) El *Middleware* de administración de *DW* ayuda a manejar y afinar las funciones del *DW* para obtener un máximo rendimiento y seguridad.
- 4) Las herramientas de acceso y reporte de datos permiten al usuario convertir con facilidad los datos en reportes de negocios y diagramas.
- 5) Las herramientas OLAP permiten al usuario navegar en los datos en vez de preguntar cosas específicas y además ayudan a manejar funciones analíticas complejas.
- 6) Las herramientas de Internet permiten que se accese la información desde cualquier punto con sencillez y seguridad.
- 7) Herramientas de minería de datos que buscan automáticamente patrones y relaciones en los datos.
- 8) Herramientas y utileras de desarrollo que simplifican el desarrollo y manejo de los datos y metadatos.

1.3.13 Construir con un alto grado de escalabilidad

Como explicamos anteriormente, las bases de datos inician con un tamaño pequeño (relativamente), y tienden a ir creciendo, en apariencia hasta infinito. Lo cuál nos obliga a considerar que el *hardware* que soporte estas aplicaciones tiene que ser altamente escalable, pues de otra manera las primeras etapas del desarrollo mostrarían un rendimiento considerablemente superior al de las subsecuentes, y las etapas finales

podrían llegar a ser inoperantes al cargarse toda la información y los nuevos usuarios del sistema.

A menudo, un componente subestimado del desarrollo *DW* es el sistema que manejará el almacenamiento físico de los datos, su movimiento, respaldo y recuperación. Si se diseña mal o no está completamente planificado, se puede convertir en un factor que frene el desempeño de las aplicaciones.

1.3.14 Construir en un ambiente operativo abierto

Debido a la diversidad de los datos que antes mencionamos, y a la diversidad de las herramientas, utilerías y componentes de bases de datos que hay y que se pueden manejar, es muy conveniente que no se case el *DW* con una marca o producto específico, lo mejor es considerar un ambiente que permita el uso de una variedad de productos y soluciones para que nos permita utilizar diferentes herramientas conforme se pongan a disponibilidad (y el cambio sea conveniente). Pero sobre todo, para que podamos aprovechar la experiencia de diversos especialistas y no quedemos a merced o expensas de un único proveedor.

En este capítulo se han revisado los aspectos y componentes básicos de un *DW*, a continuación se presentan los pasos formales para la construcción de un *DW*.

Capítulo 2

Metodología de Data Warehouse

En este capítulo se detalla desde la metodología hasta los procesos involucrados en la arquitectura del Data Warehouse.

2.1 Arquitectura

2.1.1 Planeación de la Arquitectura

Debido a que las empresas requieren manejar muchas transacciones en forma práctica y a muy bajos costos, tienen la imperiosa necesidad de integrar su información en *DWs*. Este proceso resulta sumamente difícil cuando nos enfrentamos a la compleja tarea de integrar los sistemas y las tecnologías operacionales existentes, con las nuevas herramientas de *DW*. La estructura que vincula todos los elementos y los mantiene integrados es conocida como una *Arquitectura*.

Las áreas de definición y planeación de la arquitectura de sistemas ya se ha discutido y comentado al paso de los años. La definición y la planeación de la arquitectura son procesos para muestrear toda la información de la empresa, así como sus metas y objetivos. La experiencia nos dice que la mayoría de las empresas no se han tomado el tiempo necesario para canalizar todo el esfuerzo requerido con la finalidad de mapear correctamente los requerimientos del negocio hacia la *Arquitectura*.

Enfocarse hacia una arquitectura errónea resulta muy costoso, y no sólo en dinero, ya que el costo de interrumpir con las actividades de la empresa, puede ocasionar un gran contratiempo en la operación del negocio.

Por ejemplo, imaginemos el impacto de una arquitectura errónea en un escenario diferente al informático. Imaginemos que le pedimos a un constructor que nos construya una casa. Se realizan algunas pláticas donde el constructor entiende que la necesidad primordial radica en contar con mayor espacio de almacenamiento del que tenemos actualmente. Supongamos que el constructor interpreta que más espacio lo darán unos closets, no un sótano. El constructor procede con el diseño y a la construcción de una

casa de concreto con varios closets para nosotros. Una vez terminada descubrimos el gran error ya que nuestra necesidad de espacio era un sótano. Una vez que le notificamos al constructor que más espacio quiere decir un sótano, el constructor nos presenta una nueva cotización con un costo estimado para la modificación.

¿Qué tan difícil y costoso resultará agregar un sótano a la casa?, ¿cómo podemos comparar el costo de un diseño adecuado desde el inicio? Si el constructor crea una maqueta mostrando la planeación de la casa y todas las consideraciones necesarias, y si además, revisamos todo conjuntamente no tendremos este tipo de sorpresas. Ya que la casa será construida correctamente desde el principio, colocando los cimientos necesarios para soportar el proyecto.

Los *DWs* no son diferentes. El desarrollo basado en una Arquitectura proporciona las bases necesarias para la integración. La definición de una arquitectura en la etapa inicial de un proyecto de *DW* resulta crítica. Lo primero que se requiere es desarrollar una arquitectura, posteriormente canalizaremos todo el esfuerzo hacia el desarrollo del *DW*. La arquitectura definirá un esquema laboral apropiado, proporcionando un conjunto de métodos balanceados, una metodología para la definición de las aplicaciones y de los datos, así como los aspectos tecnológicos correspondientes para el *DW*. Además también nos permitirá definir los estándares de calidad, el diseño general y las técnicas de soporte para el proyecto.

El proceso para definir la arquitectura permitirá incrementar grandemente la tasa de éxito del proyecto *DW*.

Para construir una arquitectura para un *DW*, se requiere que el equipo de trabajo del *DW* realice lo siguiente:

- Entender la información que presenta la tecnología existente.
- Entender la problemática, especialmente el modelo de datos y las prioridades de la empresa.
- Mapear los datos operacionales existentes (y los sistemas que los administran) de las áreas seleccionadas.

Como podemos interpretar, una arquitectura prepara el camino para un *DW* estable, útil y extensivo. Como sea, por sí sola la arquitectura no resuelve los problemas de calidad de los datos en las bases de datos actuales. Corregir dichos problemas requiere de sistemas que actualicen la arquitectura de los sistemas operacionales. Un *DW* ayudará a comprender los requerimientos para lograr las mejoras de los sistemas existentes, aunque implementar esas mejoras requiere de un esfuerzo adicional.

Definición de la Arquitectura

El término Arquitectura tiene muchos significados entre las personas que se dedican al desarrollo de sistemas computacionales

Presentaremos una definición literal del término:

Arquitectura: Es un estilo, un método de diseño y construcción, un arreglo ordenado de partes.

En un *DW* la arquitectura facilita la creación de recursos de datos precisos, que se puedan compartir, y de muy fácil acceso para toda la empresa.

Una arquitectura proporciona un bosquejo que explica cómo la visión, objetivos y metas del *DW* serán entregados. Los componentes incluyen datos compartidos, infraestructura tecnológica, y el código existente con posibilidades de reutilizarse. Debido a que una arquitectura presenta algunos beneficios clave a la organización, incluiremos los siguientes:

- **Datos más consistentes.** Como datos comunes y estandarizados, así como modelos que empiezan a utilizarse, la consistencia de los datos será la norma en la empresa.
- **Desarrollo de aplicaciones simplificadas.** Usando un diseño inteligente y una implementación cohesiva que se acerque a la forma en que la empresa crea, accesa y modifica datos, el tiempo para implementar nuevas aplicaciones o cambiar las existentes tenderá a reducirse.
- **Responsabilidades de la empresa.** La integración de los datos, aplicaciones y arquitecturas tecnológicas influirán en la habilidad de la empresa para responder a las necesidades del negocio de una manera consistente, oportuna y de muy alto nivel.

Para construir un *DW* primero debemos de establecer una arquitectura y no después de haber empezado el proyecto, ésta debe establecerse desde el principio por toda la empresa para proceder con el desarrollo. Sin la aprobación y el correcto soporte, todos los esfuerzos encaminados al desarrollo de un *DW* en una organización, inevitablemente fallarán.

2.1.2 Requerimientos

¿Cómo comenzar a formular los requerimientos para un *DW*? Partamos de la diversidad de formas en que se considera un *DW* presentando los siguientes ejemplos:

Un *DW* es simplemente un sistema de aplicación empresarial con su propia base de datos. Esta base de datos se genera a partir de otras bases de datos operacionales, no de información inicial que se introduce. El *DW* ofrece una serie de características y funciones para implementar procesos empresariales y enlazarlos con otros

procesos fuera del ámbito del *DW*. De manera muy similar a otros sistemas empresariales, se requiere que el *DW* proporcione al usuario final estos grupos de características y funciones muy eficientemente.

Un *DW* es una capacidad latente. Almacena información resumida que se organiza de acuerdo con temas empresariales, tales como clientes y productos, para analizar la información con más facilidad. La labor de mostrar, organizar y reportar la información que guarda el *DW* corresponde a las herramientas que deben incorporarse. En esta visión, un *DW* tiene una capacidad latente que sólo se vuelve útil cuando las herramientas de análisis y reporte se aplican con inteligencia a los datos que conserva el *DW*; además, se requiere que soporte un gran número de herramientas de acceso, operadas por una extensa gama de usuarios finales. También debe guardar y administrar un ámbito de información grande para servir a una gran clientela.

Un *DW* es una base de datos histórica, la cual es una acumulación de muchos años de información transaccional en línea, organizada para eficientar el almacenamiento y facilitar la recuperación. Se requiere que el *DW* organice grandes cantidades de información de manera compacta y eficiente. También es necesario que proporcione técnicas para resumir, a fin de que los usuarios finales lo comprendan con más facilidad.

Algunas veces un *DW* es una tienda de datos operacionales. Entrega información operacional a un gran número de usuarios copiando información de bases de datos de los sistemas operacionales. En este caso, se requiere que el *DW* distribuya información operacional de manera eficiente a muchos usuarios. También se debe considerar que el *DW* realice las adecuaciones tecnológicas necesarias para trasladar la información de su base de datos operacional a la tecnología de almacenamiento que se emplea en el *DW*.

Resulta evidente que los requerimientos para un *DW* son tan variados y diversos como las clases de usuarios que lo utilizan para obtener beneficios empresariales. Por lo tanto, es necesario clasificar los requerimientos del *DW* utilizando técnicas clásicas.

Una de estas técnicas clásicas es, por ejemplo, el diagrama de Zachman, la cual es una de las formas más eficientes de visualizar un sistema desde muchas perspectivas. Los requerimientos que son visibles desde cada una de las perspectivas son los que proponen las personas que tienen esas perspectivas.

En un documento histórico de 1987, John Zachman comparó la construcción de un sistema de información grande y complejo, con la construcción de un edificio enorme e igualmente complejo. Hacer el intento de dónde comenzar la tarea edificadora es un gran reto. Existe una gran variedad de personas involucradas en la construcción de un edificio. A los dueños o inversionistas les interesa un edificio que cumpla con las necesidades y estilo de vida de sus residentes. El arquitecto pretende diseñar un edificio que incorpore las necesidades del dueño en cuanto a eficiencia y estética. El constructor quiere recibir las especificaciones precisas para realizar su tarea. Cada una de estas personas visualiza

la construcción de una manera muy diferente. El dueño lo ve como una inversión financiera y mide la ganancia con el costo. El arquitecto lo ve como planos y especificaciones para los materiales de construcción. Es probable que el constructor considere la actividad como planos eléctricos y de plomería para el edificio.

Zachman también mostró que cada una de estas personas tenían intereses de diferentes dimensiones sobre el edificio. Al plomero del constructor le interesa desde el punto de vista de la plomería, mientras que el electricista se concentra en los cables y las instalaciones eléctricas.

Después, Zachman mostró cómo combinar estos puntos de vista y dimensiones para formar una matriz a la que llamó *Diagrama de Zachman* (figura 2.1). Este diagrama representa la perspectiva de arriba hacia abajo, conducida por el negocio, o el dueño, al igual que el punto de vista de abajo hacia arriba, conducido por la construcción o del implementador, de un sistema de información.

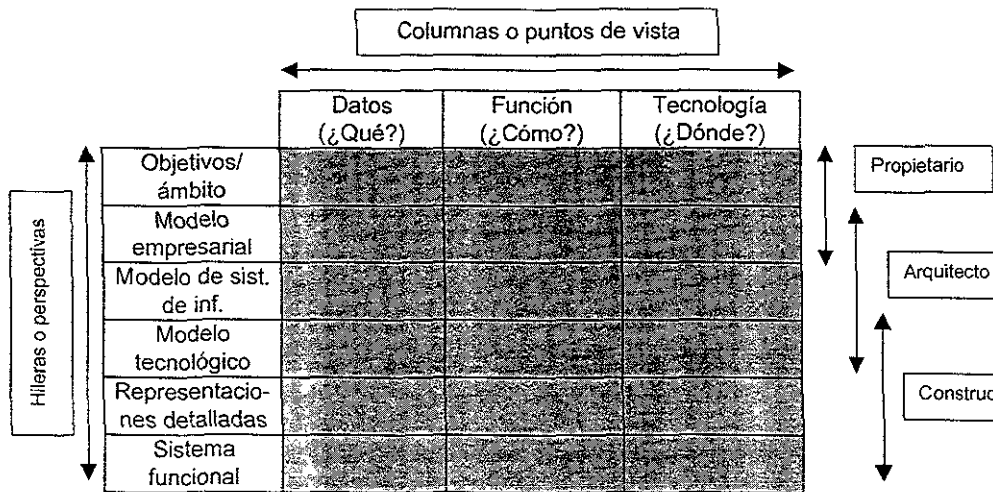


Figura 2.1 Diagrama de Zachman para arquitecturas de sistemas de información

Cada celda del diagrama de Zachman (intersección fila/columna) representa una percepción arquitectónica (modelo) de un tipo de participante en el sistema. Formalmente las filas representan puntos de vista o perspectivas y las columnas un aspecto o visión del sistema.

En su documento Zachman se refirió a las siguientes personas que tenían interés en un Sistema de Información:

- El dueño o inversionista en el Sistema de Información, quien intenta resolver un problema empresarial utilizando la Tecnología de la Información.

Por lo general, el punto de vista del dueño se mide en dinero y la perspectiva que tiene del negocio es periodos y procesos empresariales. En general dicho interés en la parte de la tecnología se limita al análisis de inversiones y pagos, y a la evaluación de riesgos y programas. El inversionista considera que el *DW* forma parte del DSS (Decision System Support, Sistema de Soporte de Decisiones) dentro de la perspectiva empresarial total (ver figura 2.2)

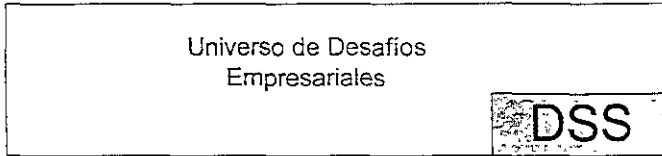


Figura 2.2 Ámbito de los desafíos empresariales que solventa la tecnología del *DW*.

En el DSS el dueño confía que el *DW* ofrece análisis y razones para tomar decisiones empresariales. Gran parte de la capacidad de análisis con la que cuenta el *DW* se basa en la información histórica y la extrapolación de las tendencias. Las áreas donde el inversionista requiere soporte de decisiones son las que afectan el negocio en forma integral (Ventas, Mercadotecnia, Finanzas, Administración estratégica, Planeación y desarrollo de productos, Atención a clientes, Recursos humanos, etc.).

- El usuario que debe utilizar el *DW* con regularidad para realizar funciones empresariales. La perspectiva que tiene el usuario del *DW* está muy bien definida y prescrita como un conjunto de procesos y pasos que debe seguir para llevar a cabo el análisis y la función empresarial. Normalmente no se interesa en la manera de estructurar el *DW* o cómo se forma, sólo requiere que realice las funciones señaladas correcta y eficientemente.
- El arquitecto que debe entender tanto las necesidades empresariales del *DW*, como la tecnología de implementación que se necesita para solventarlas, además de ser el negociador entre el inversionista que ha encargado un sistema basado en un punto de vista de alto nivel de las necesidades empresariales y el constructor que necesita especificaciones detalladas para preparar un *DW*.
- El constructor, quien es el responsable de la instalación e integración física de diversos componentes del *DW*. Debe incorporar los requerimientos operacionales y de despliegue en el *DW*, como son la disponibilidad, la capacidad de recuperación y la seguridad.

Recordando el diagrama de Zachman, el cual representa dos dimensiones de un sistema, las hileras representan las distintas perspectivas de los depositarios (el dueño, el arquitecto y el constructor), y las columnas representan la visión del sistema (los datos, el

proceso, la red). La siguiente figura es un ejemplo de cómo tres tipos de perspectivas de depositarios pueden tener seis diferentes puntos de vista.

	QUÉ	COMO	DÓNDE	QUIEN	CUANDO	PORQUE
Dueño	<ul style="list-style-type: none"> DSS para análisis de ingresos del cliente 	<ul style="list-style-type: none"> Analizar datos históricos 	<ul style="list-style-type: none"> Oficina del departamento de mercadotecnia 	<ul style="list-style-type: none"> Departamento de mercadotecnia 	<ul style="list-style-type: none"> Mensualmente Trimestralmente Anualmente 	<ul style="list-style-type: none"> Negocio Objetivos Decisiones de mercadotecnia promocional
Usuario empresarial	<ul style="list-style-type: none"> Áreas tema Dimensiones Granularidad 	<ul style="list-style-type: none"> Profundizar Reportar Consultar 	<ul style="list-style-type: none"> Escritorio Sala de conferencias del departamento 	<ul style="list-style-type: none"> Administradores Analistas 	<ul style="list-style-type: none"> Información fluida 	<ul style="list-style-type: none"> Objetivos operacionales Soporte de decisiones
Implementador de tecnología de la información	<ul style="list-style-type: none"> Áreas tema Metamodelo de DW 	<ul style="list-style-type: none"> DW y funciones del mercado de datos 	<ul style="list-style-type: none"> LAN Trabajo corporativo en red 	<ul style="list-style-type: none"> DA/DBA Analista de dominio Programador de aplicaciones 	<ul style="list-style-type: none"> DW y ciclo de actualización de carga 	<ul style="list-style-type: none"> Objetivos técnicos

Figura 2.3 Perspectivas del DW establecidas mediante el diagrama de Zachman.

2.1.3 Evaluación de la Arquitectura existente

Cualquier iniciativa tecnológica debe considerar la infraestructura de cómputo existente. Si el equipo del proyecto contará con un ambiente limpio de consideraciones para desarrollar sistemas, la vida sería infinitamente más sencilla. Pero esto raramente ocurre. Una arquitectura debe permitir a los negocios migrar hacia nuevas tecnologías de modelado, bajo controles estrictos con un mínimo de interferencias. Por ello hay que ser muy cuidadosos a la hora de integrar en nuevos DWs las aplicaciones y sistemas patrimoniales.

Para soportar la integración de los sistemas patrimoniales, resulta crucial un entendimiento estable del medio ambiente actual. Los sistemas patrimoniales impactan el desarrollo de una nueva arquitectura. Los problemas heredados por los sistemas y su infraestructura incluyen lo siguiente:

- Los sistemas que tienen información que no puede ser accesada sin un significativo y personalizado desarrollo de aplicaciones.
- La carencia de una base adecuada de conocimientos que soporte e integre los componentes del sistema.
- La disminución de los recursos computacionales, incluyendo equipo y personal.

Además de estos problemas, los sistemas patrimoniales son el respaldo de los sistemas de información que mantienen a la compañía funcionando y por lo tanto, deben ser tomadas en cuenta todas las precauciones para evitar interrupciones en la operación.

Modelando a la empresa actual

Cuando documentamos las tecnologías existentes, los esfuerzos deben ser enfocados a comprender los datos, aplicaciones y áreas tecnológicas de los actuales sistemas de información. Es importante definir el soporte a la organización detrás de los sistemas y tecnologías existentes. Esta etapa dará una nueva luz a los procesos de la organización, así como lo que se ha hecho y lo que se ha dejado de hacer históricamente para soportarlos. El equipo encontrará respuestas a las preguntas como las que se presentan a continuación:

- ¿Cuáles de las funciones empresariales se integrarán al proyecto?
- ¿Qué información se requiere para soportar dichas funciones?
- ¿Dónde se realizan esas funciones?
- ¿Con qué frecuencia se ejecutan?
- ¿Quién ejecuta las funciones empresariales?
- ¿Qué se requiere para mejorar la ejecución de dichas funciones?

La información reunida durante esta etapa, deberá ser capturada, almacenada y organizada en un modelo automático. Probablemente ésta será la primera vez que se realice una evaluación de esta magnitud en la compañía, por ello la información reunida en esta etapa deberá ser altamente resumida, lo que resultará en un fundamento sólido en el que análisis posteriores podrán basarse. La colección formal de esta información será de gran ayuda en esfuerzos posteriores, incluyendo los *DWs* orientados a un tema. Al finalizar esta etapa, el equipo habrá producido un conjunto de modelos que definan a los sistemas y tecnologías de la empresa tal y como se encuentran soportando las operaciones empresariales.

Revisión de los sistemas y tecnologías existentes

Cuando se evaluó la arquitectura de la empresa, el equipo debió inventariar cada una de las funciones empresariales que se requieren dentro del ámbito del *DW*. Estas funciones deberán modelarse y colocarse en los sistemas y subsistemas aplicativos. El resultado del conjunto de modelos definirá las relaciones e interdependencias entre dichos sistemas.

Debido a que la mayoría de los sistemas de información de las organizaciones están basados en funciones empresariales, no encontraremos ninguna organización típicamente sencilla que entienda todo el proyecto. Es por ello que se requieren programar entrevistas individuales con cada una de las organizaciones de soporte para entender toda la información referente a cada una de ellas. Los mejores candidatos para las entrevistas y

recopilación de información deben incluir grupos de soporte de las aplicaciones del departamento de sistemas de la compañía, así como “meta usuarios” de los sistemas que puedan ser representativos de los usuarios, quienes deben poderse encontrar en grupos de coordinación de sistemas.

Modelando la arquitectura de las aplicaciones

Es común empezar el análisis arquitectónico con las aplicaciones que soportan los negocios, debido a que hay mejor entendimiento de los datos asociados a estos sistemas. Podremos de una manera rápida determinar los orígenes de los datos que requieren ser analizados en la fase de la arquitectura de los datos, como objetivo de la aplicación.

Con cada aplicación trataremos de captar la información de los sistemas en general, tan completos como su relación con los demás sistemas y que nos conduzca a la construcción de la arquitectura de las aplicaciones. Un diagrama hipotético de un sistema y sus relaciones es descrito en la figura 2.4, donde claramente podemos apreciar que el sistema de “Control de Manufactura” es el elemento concentrador y regulador de todos los flujos de información, así como sus interrelaciones con todos los demás procesos.

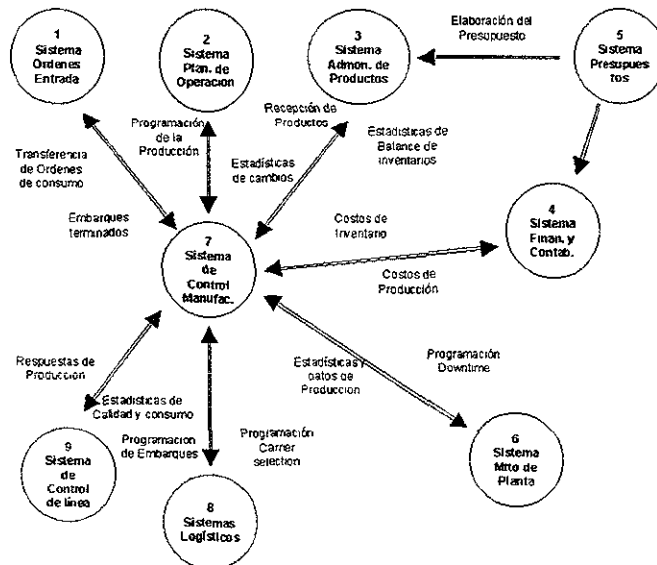


Figura 2.4 Arquitectura de una aplicación de alto nivel de una empresa manufacturera

2.1.4 Diseño de la arquitectura

El proceso de desarrollo involucra mucha investigación y discusiones extensivas sobre el contenido de la arquitectura. La creación de la arquitectura se realizará después de muchas revisiones de arquitecturas de la organización con representantes de todas las áreas del negocio.

Para cada organización se tiene que determinar qué componentes son críticos, se recomienda basarse en una arquitectura que incluya definiciones para las aplicaciones, los datos y la tecnología. Frecuentemente, la arquitectura tendrá que ser subdividida en arquitecturas más pequeñas que van definiendo las estructuras funcionales de la empresa. Igualmente, puede ser que éstas deban ser subdivididas en sus propias sub-arquitecturas, que le darán soporte a cada punto particular de la estructura general del negocio.

Para entregar todo lo que se requiere (u ofrece) de un *DW*, se tiene que establecer una sólida fundamentación, por ello el diseño de la arquitectura es crítico para el éxito general del *DW*. Sin una arquitectura adecuada, la conectividad de los componentes y las tecnologías subyacentes de un *DW* no ocurrirán, en cambio, con una arquitectura bien definida, es más sencillo lograr los beneficios e impactos de los *datamarts* y su posterior expansión a *DW*.

Las fases que a continuación se desglosan no se consideran en modo alguno una guía para diseñar una arquitectura adecuada al negocio, pero sí como un compendio empírico de desarrolladores:

- Fase I: Planeación de la arquitectura.
 - Crear un enfoque.
 - Adoptar una metodología.
 - Crear un equipo para la arquitectura.
 - Definir el alcance del proyecto.
 - Definir las metas y objetivos del negocio.
 - Planear la estructura de terminación del trabajo.

- Fase II. Construcción del plano de la arquitectura.
 - Definir los requerimientos.
 - Definir la arquitectura para los datos.
 - Definir la arquitectura para las aplicaciones.
 - Definir la arquitectura para la tecnología.
 - Determinar el conjunto de herramientas del *DW*.
 - Generar una documentación estándar de la arquitectura (planos).
 - Revisar

Debido a la naturaleza tan grande en tamaño, y tan precisa en cuanto a objetivos del *DW*, si no hacemos toda esta etapa de diseño, podemos, metafóricamente hablando, obtener una gran plataforma de lanzamiento en un punto equivocado tal, que no nos permita acertar a donde queremos llegar.

Ahora desarrollaremos brevemente algunos de los puntos enumerados en las fases I y II del diseño de la arquitectura.

De la fase I

- **Crear un enfoque**

La importancia del enfoque (paradigma u orientación) en las empresas fue especialmente notoria cuando en los 80's las empresas cambiaron su enfoque al modelo de entregas a tiempo, y al de fabricar bajo pedido, se redujeron mucho los costos de almacenaje y al mismo tiempo se logró una mayor satisfacción de los clientes. Igualmente, el enfoque del DW debe estar en el soporte de decisiones de negocios, a través de la documentación de las mismas, no en la automatización de procesos específicos de negocios.

Para ello, se recomienda que el sistema aporte:

- Reportes donde el usuario pueda definir el tipo de información que quiere.
- Un panel de control donde los usuarios puedan observar en tiempo real el estado de las variables de negocio que elijan.
- Agentes delatores de excepciones que alerten a los usuarios para realizar correcciones.

- **Adoptar una metodología**

Según su definición, la metodología es un conjunto de prácticas, procedimientos y reglas utilizadas por quienes trabajan en una disciplina, o realizan alguna investigación. La metodología debe abarcar todas las tareas y actividades que se tienen que realizar junto con la información que se debe capturar para definir e implementar una nueva arquitectura.

Las áreas que actualmente se consideran son:

- La organización de las fuentes y almacenes de la información de negocios en la empresa.
- La definición y soporte de los procesos de *software* para la implementación de la funcionalidad requerida.
- La infraestructura de cómputo que habilite a las aplicaciones y los datos a interactuar por toda la empresa.

La metodología adoptada en el desarrollo da lineamientos para el plan de trabajo del proyecto y para la capacitación, define los pasos, junto con su propósito, entregas, documentación, lineamientos, roles, responsabilidades y estimaciones.

Se recomienda hacer la selección de herramientas durante esta etapa del proceso, ya que hay que verificar que soporten la metodología definida y den facilidades para el reporte de datos, incluyendo: selectividad, almacenamiento, preguntas *ad-hoc* y facilidad de reporte.

- **Crear un equipo para la arquitectura**

Si bien la metodología dará la estructura general para la arquitectura, las personas que trabajen en ella harán la investigación y la estructuración del contenido de la arquitectura, este equipo de personas debe formarse de:

- Una persona que controle las actividades diarias del proyecto y sus miembros, que haga reportes, evalúe avances, programe entregas y encuentre los factores que impiden las entregas.
- Una persona que dé una visión de usuario y conocimientos del negocio al equipo.
- Una persona que construya las áreas en que se va expandiendo el negocio, que las domine y que aporte su experiencia al equipo.
- Una persona que supervise la instalación, preparación y uso de los conjuntos de herramientas.

Hay que considerar que una buena parte de la efectividad del equipo dependerá de que no esté compuesto totalmente de "Sistemólogos", sino que se recomienda una relación 50-50 para una máxima efectividad, para lograr un buen balance se recomienda tomar en cuenta a las personas que construyeron las áreas dentro de la organización, y al propio Patrocinador, o algún miembro del comité que designe.

- **Definir el alcance del proyecto**

Además de todo lo que ya se ha mencionado anteriormente, es necesario definir claramente el alcance del proyecto para poder justificar los gastos de diseño de una arquitectura.

- **Definir las metas y objetivos del negocio**

Las metas y sus correspondientes objetivos deben establecer y documentar claramente las razones del negocio que guían las iniciativas del *DW*. Igual de claro hay que definir los usos del *DW* en toda la empresa, y los beneficios de la aplicación para el negocio en su totalidad. Lo más importante de estas definiciones es que deben ser claras, concisas y no técnicas.

- **Planear la estructura de terminación del trabajo**

Un compromiso de entrega bien planeado lleva a entregas de calidad y evita que se cancelen futuros planes o puntos relativos al *DW*.

De la Fase II

- **Definir la arquitectura para los datos**

Como hemos mencionado, los datos de un *DW* pueden estar dispersos, y además ser muy numerosos, lo cual obliga al equipo de desarrollo del *DW* a analizar la disposición de los datos, como se puede analizar una red, y de igual manera, se tiene que diseñar una arquitectura que nos permita localizar y definir en general los datos (los repositorios, los datos operacionales, los metadatos) en esta red, para que con base en ella, podamos decidir la correcta implementación del *DW*, o los cambios necesarios para la implementación del *DW*.

- **Definir la arquitectura para las aplicaciones**

La arquitectura de las aplicaciones definirá como van a interactuar los usuarios con el *DW*, esto varía desde un ambiente estándar de reportes de datos, hasta uno de herramientas de reportes personalizados y a la medida altamente dinámicos. Para decidir, conviene hacer una evaluación de las habilidades técnicas de los usuarios, como veremos en el siguiente capítulo.

- **Definir la arquitectura para la tecnología**

Es necesario hacer notar que es tan importante la logística que se implica en todo el *DW*, como son importantes las máquinas que harán posible el funcionamiento de todo este engranaje de información junto con sus arquitecturas de *hardware* y *software*.

- **Generar una documentación estándar de la arquitectura (planos)**

Estos "planos arquitectónicos" del *DW* deben contener los siguientes puntos:

- Descripción de lo que el negocio espera del *DW*.
- Cómo se va a entregar lo que el negocio espera.
- Definición clara de ciclos de entrega por fases, incluidos procesos de refinamiento y revisión de la arquitectura.

Y se deben desglosar en la siguiente forma:

- Componentes del cliente.
- Componentes del servidor.
- *Hardware*.
- Sistema operativo.
- Protocolos de conectividad.
- Herramientas de usuario final (EIS/EDS, reportes *ad-hoc*, OLAP, minería de datos).
- Herramientas de desarrollo para clientes.
- *Software* de manejo de datos.
- *Software* de manejo del *warehouse*.
- Herramientas de desarrollo para servidores.
- Repositorio para metadatos.
- Herramientas de conectividad.
- *Software* de calendarización.
- Distribución de la información.
- Herramientas de archivo
- Herramientas de transformación.
- Manejo de sistema y procesos.
- Carga de grandes volúmenes de datos.
- *Software* de respaldo y recuperación de datos.
- Resistencia a fallas de sistema.
- Seguridad.
- Almacenamiento y recuperación de textos y documentos.
- Almacenamiento y recuperación de elementos multimedia.
- Interfaces WWW y otras de *Internet*.

Grosso modo, los planos esencialmente, traducen la misión, las metas y los objetivos de la empresa para el DW, en una arquitectura tecnológica lógica compuesta de sub-arquitecturas individuales para los componentes de aplicaciones, de datos y de tecnología de un DW. En ellos, los requerimientos del negocio son mapeados en requerimientos de arquitectura, que en su momento se mapean en requerimientos de producto en grados tecnológicos.

2.1.5 Definición de los componentes individuales

La arquitectura de un DW es una forma de representar los componentes de una solución DW y su interrelación.

En este sentido podemos mencionar los siguientes componentes.

- Bases de datos operacionales.
- Procedimientos de carga.
- Data Warehouse.
- Metadatos.

- Middleware.
- Sistemas estratégicos/Herramientas de extracción.

Bases de datos operacionales

Los sistemas OLTP son sistemas normalmente optimizados para el manejo de un conjunto predefinido de transacciones, y de ellos se transferirá la información seleccionada, pueden haber sido construidos utilizando manejadores de datos relacionales (RDBMS, Relational Database Management System, Sistemas de Administración de Bases de Datos Relacionales), manejadores de archivos jerárquicos, de archivos planos u otro tipo de manejadores. Por lo anterior es necesario analizar y definir cuidadosamente aquellos datos de los sistemas operacionales que representen la esencia o filosofía del negocio que se pretenda manejar, para que al transferir los datos al *warehouse*, ese conocimiento primordial se capture en lo que se conoce como *metadatos*.

Los datos extraídos de los diferentes sistemas pueden, a su vez, ser manejados por un RDBMS. Normalmente los sistemas operacionales de los cuales se extraen los datos son muy diversos y cada uno de ellos aporta varios gigabytes, por lo que es normal que las bodegas de datos contengan del orden de 20, 50, 100, 200 o más gigabytes, aunque es posible, y en algunos casos puede ser deseable, construir un *DW* tan pequeño como 200 ó 500 megabytes.

Procedimientos de carga

Básicamente sólo existen dos formas de realizar esta tarea, la Acumulación simple” y el *Rolling*.

La Acumulación simple es sin duda la más sencilla y común, y consiste en realizar una sumariazación o resumen de todas las transacciones comprendidas en el periodo de tiempo seleccionado y transportar el resultado como una única transacción hacia el *DW*.

El proceso de *Rolling* por su parte, se aplica en los casos en que se opta por mantener varios niveles de granularidad. Para ello se almacena información sumariazada (resumida) a distintos niveles, correspondientes a distintas agrupaciones de la unidad de tiempo

Data Warehouse

El repositorio de datos es una enorme colección de datos provenientes de sistemas operacionales y otras fuentes, después de aplicarles los procesos de análisis, selección y transferencia de datos seleccionados. El objetivo del *DW* es el uso adecuado de esos datos para obtener información útil para el soporte a la toma de decisiones, lo que es difícil de lograr con sistemas operacionales.

Metadatos

Uno de los componentes más importantes de la arquitectura de un *DW* es la *Metadata* (Plural en latín de *metadato*). La *metadata* es la base de datos que contiene los “*Metadatos*”. Estos últimos son definidos comúnmente como “datos acerca de los datos”, en el sentido de que se trata de datos que describen cuál es la estructura de los datos y cómo se relacionan.

En una base relacional, los datos residen en las tablas que el diseñador de la aplicación definió para dar soporte a la misma, mientras que los *metadatos* se encuentra en una serie de tablas del sistema, frecuentemente bajo nombres como *systables*, *syscolumns*, *syscatalog*, etc. Los *metadatos* documentan exactamente; entre otras cosas: que tablas existen para esa aplicación, qué columnas posee cada una de ellas y qué tipo de datos se pueden almacenar. Los datos son de interés para el usuario final; los metadatos son de interés para los programas que tienen que manejar estos datos.

Sin embargo, la función que desempeñan los metadatos en un ambiente de *DW* es muy diferente a la que cumple en los ambientes operacionales.

Otra de las razones que determina la importancia de los metadatos es el horizonte de tiempo que abarca un *DW*. Un *DW* generalmente contiene información comprendida en periodos de 10 a 15 años, por lo tanto, los metadatos deberán mantener un registro de los cambios que ocurran en la estructura del *DW* a través del tiempo

Por último, debido a que los *metadatos* en los ambientes de *DW* son utilizados en forma directa por el usuario final a través del uso de las herramientas de consulta, es necesario proveer de un mapeo que permita al mismo, interactuar con ellos expresándose en términos de Negocio y no en términos computacionales.

Middleware

El *middleware* es el *software* que actúa como intermediario entre las diversas aplicaciones del *DW*, dentro de la infraestructura de red. Debido a que se sitúa en medio de estas dos grandes capas, obtiene el nombre de *middleware*.

Aunque no existe una clasificación estándar de *middleware*, en este rubro se encuentran distintos tipos de *software* como: monitores de transacciones, ligas estándares entre manejadores de bases de datos y *front-ends* (ODBC, SAG-CLI, IDAPI, etc.), distintos tipos de *gateways* (de bases de datos, de correo electrónico, de seguridad, etc.), herramientas de manipulación de datos (extractores, convertidores de formatos, calendarizadores) así como las nuevas arquitecturas de objetos distribuidos (OLE y CORBA entre las principales) y los sistemas basados en mensajes (MOM, Message Oriented Middleware)

Algunos especialistas internacionales han empezado a hablar de la conveniencia de hacer toda una arquitectura de *middleware* para responder a las necesidades de interoperabilidad de hoy y del mañana de un *DW*. Desde este punto de vista, una

arquitectura de *middleware* debe cumplir con diversas funcionalidades para garantizar que:

- Operará con un rendimiento adecuado.
- Podrá adaptarse a los cambios y adiciones que se presenten.
- Será fácil de usar y administrar.

En proyectos mayores y/o de ambientes muy heterogéneos, una arquitectura de *middleware* contemplará por lo general, diversos elementos y productos, probablemente de distintos fabricantes.

Sistemas estratégicos/Herramientas de extracción

Como parte de la arquitectura, se tienen que determinar que tipo de sistema vamos a tener en el *DW*, o dicho más al estilo del *DW*, es necesario definir la arquitectura de las aplicaciones, y principalmente, que tipo de herramientas vamos a utilizar para el desarrollo. Esto es principalmente debido a que un *DW* es realmente muy grande, y a diferencia de los sistemas tradicionales, no podemos usar la misma herramienta para programar, hacer reportes, transferir información, manejar las bases de datos, etc.

Especialmente, en un *DW* se debe tener mucho cuidado con la selección de las herramientas que se utilizan, y doblemente con las de extracción de datos, pues como ya hemos mencionado anteriormente, una buena parte del éxito del *DW* depende de que los usuarios puedan obtener la información como si se la pidieran a un "sistemólogo". Entonces, las herramientas EIS, DSS, las que nos permitan hacer reportes personalizados, los ambientes de desarrollo de aplicación, los productores de reportes y otras herramientas, resultan particularmente importantes, y además, es igualmente importante que todas ellas sean capaces de interactuar entre ellas, y de intercambiar información, archivos, datos, etc., es decir, que haya interoperabilidad.

Ahora, considerando todas estas bases para el desarrollo de aplicaciones y estando ya familiarizados con la mayor parte de la terminología implicada, podemos iniciar con la descripción de los procesos involucrados.

2.2 Procesos

2.2.1 Extracción de datos y validación

Para lograr la extracción de datos, se requiere de algunos procesos previos como son el análisis, la selección y finalmente la extracción de los datos. Estos procesos son requeridos para seleccionar datos de sistemas operacionales, extraerlos y convertirlos a un formato o formatos que permitan manejarlos en común, de acuerdo al modelo de datos de la empresa, y de acuerdo a la información para la toma de decisiones con que se desee contar.

Los datos deberán ser actualizados (extraídos de nuevo) como un proceso cíclico, periódico; por ejemplo, cada semana, cada dos semanas o cada mes, de acuerdo a las necesidades de información actualizada que el negocio requiera. Tener o pretender actualizaciones diarias es contraproducente en la mayoría de los casos, ya que la idea es trabajar con datos históricos con los que se puedan realizar comparaciones, proyecciones, etc. Tratar de comparar datos de un día contra el anterior no es una pregunta de toma de decisiones y en caso de serlo, no se requiere de un *DW* para obtener la respuesta ya que con el sistema operacional sería suficiente.

2.2.2 Características y problemática de la extracción

Para entender mejor este tema, partamos de la siguiente premisa: "Veamos qué es lo que usan los usuarios y cómo lo utilizan"

Antes que nada, en toda aplicación de sistemas el primer paso es la comunicación con los usuarios para saber qué hacen, cómo lo hacen y qué herramientas utilizan para lograrlo. Entremos al tema de la extracción de datos para lo cuál comenzaremos por mencionar dos definiciones referentes a la extracción de datos.

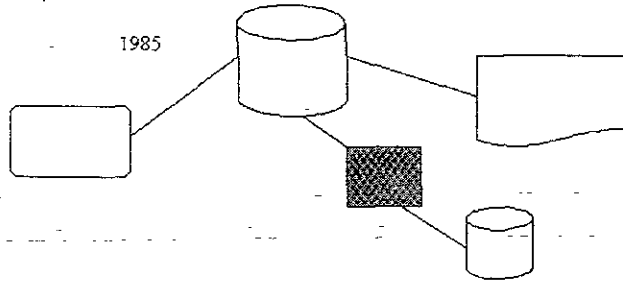
Extracción. Es la actividad relacionada con transferir datos de bases de datos operacionales (fuentes de datos) al *DW*.

Extracción de Datos. Análisis de datos a detalle para revelar relaciones, patrones y asociaciones insospechadas o desconocidas.

Como parte del proceso evolutivo de los sistemas para la toma de decisiones, en 1985 aparece el concepto de los procesos y programas de extracción.

Como se ve en la figura 2.2 el programa de extracción es el más sencillo de todos los programas (cuadro sombreado). Se ejecuta a través de un archivo o base de datos (cilindro), utiliza ciertos criterios para la selección (cuadro sin sombreado), y una vez

encontrados los datos "calificados", los transporta hacia otro archivo, o base de datos (cilindro pequeño).



Programa de Extracción

Figura 2.2 Naturaleza del proceso de extracción.

En esos años se volvió muy popular el uso de los programas de extracción, ya que formaba parte del proceso de la información. Existen dos grandes razones por las que lograba dicha popularidad:

- Podía mover los datos sin afectar el rendimiento de los procesos, ejecutándose en línea, lo cual no ocasionaba ningún conflicto en el rendimiento cuando un gran volumen de datos requerían ser analizados.
- Cuando los datos son movidos fuera del dominio operacional transaccional con un programa de extracción, se genera un cambio en el control de los datos, por lo que el usuario final se apropia de los datos una vez que toma el control de ellos.

Por estas razones, los procesos de extracción pronto se hallaron por doquier, por lo que se realizaron muchos programas de extracción, tal como se muestra en la figura 2.3.

En la figura 2.3 se muestra un sistema complejo de procesos de extracción, que al principio fueron procesos de extracción de las bases de datos centrales hacia bases de datos del primer nivel, luego extracción de la extracción del primer nivel, después extracción de extracción de extracción y así sucesivamente. Esto no quería ser escuchado por los directivos de una gran compañía que tenía que hacer más de 45,000 extracciones por día.

Este patrón de procesos de extracción en las organizaciones empezó a ser muy común, por lo que los procesos de extracción se salieron del control generando como consecuencia lo que se llamó "Arquitectura Natural Envolverte" (figura 2.3).

Esta "nueva" arquitectura ocasionó una serie de problemas trayendo como consecuencia la pérdida de credibilidad en los datos, ya que para realizar una consulta se tomaba una fotografía del estado de las bases de datos en un momento dado, dando un resultado, pero si se realizaba la misma consulta diez minutos más tarde, la fotografía resultante era diferente a la anterior y por ende el resultado de la consulta diferente a la primera.

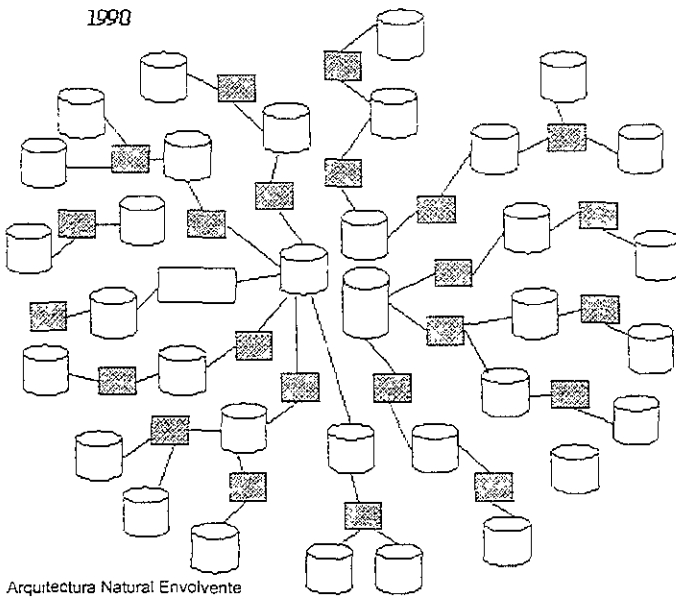


Figura 2.3 El proceso de extracción basado en la proliferación de extracciones.

Las cinco razones por las cuales se veía una crisis predecible en la proliferación de procesos de extracción son:

- No se consideraba el tiempo en la consulta de los datos.
- Existían diferentes algoritmos para la extracción.
- Varios niveles de extracción
- Datos externos.
- Diversas fuentes de datos.

Todo lo anterior llevó al desarrollo del *DW*, por lo que es necesario comenzar considerando los datos operacionales y quitarnos la idea que la creación de un *DW*, implica la extracción de los datos operacionales y su depósito en el *DW*.

En la figura 2.4 se muestra una descripción simple del depósito de datos desde un ambiente operacional existente hacia un *DW*. Aquí podemos apreciar que múltiples aplicaciones contribuyen con datos al *DW*.

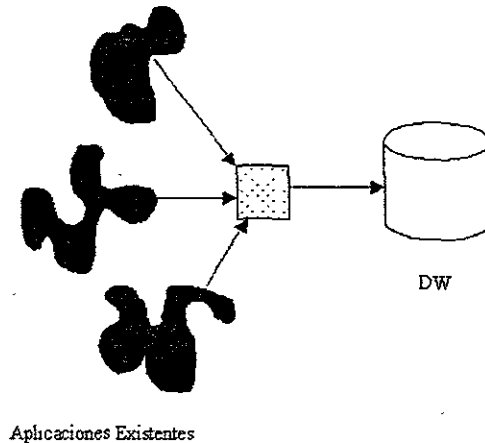


Figura 2.4 Mover datos del ambiente operacional a un *DW* no es simple proceso de extracción.

La primera falla en pensar que construir un *DW* es meramente un proceso de extracción, es que los datos en un ambiente operacional están desintegrados. Dicha falta de integración dentro del ambiente de los sistemas se muestra en la figura 2.5.

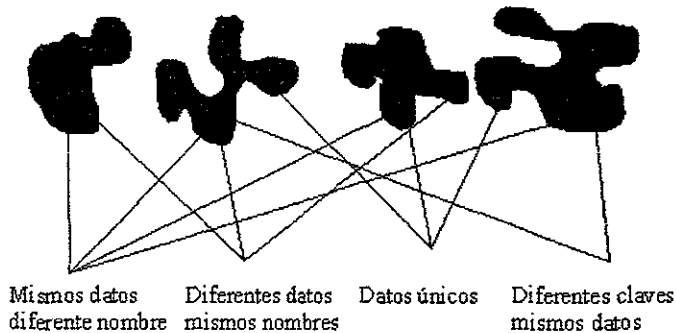


Figura 2.5 Los datos en los sistemas operacionales están terriblemente desintegrados.

Debido a que cada aplicación tiene su propio, único y privado conjunto de requerimientos, los cuales no consideraron otras aplicaciones cuando fueron desarrolladas, tratar de extraer datos de estas aplicaciones se vuelve un problema muy complicado. Está clara muestra de falta de integración, se ha convertido en una pesadilla para los

programadores. Existen mil y un detalles que se deben programar para poder traer datos desde un ambiente operacional.

Sabemos que existen muchos datos en los ambientes operacionales, por lo que tratar de rastrear en todos cada vez que se actualiza el DW es muy desgastante y poco realista.

Para evitar lo anterior, existen cinco técnicas (ver figura 2.6) que nos ayudan a limitar la cantidad de datos a rastrear.

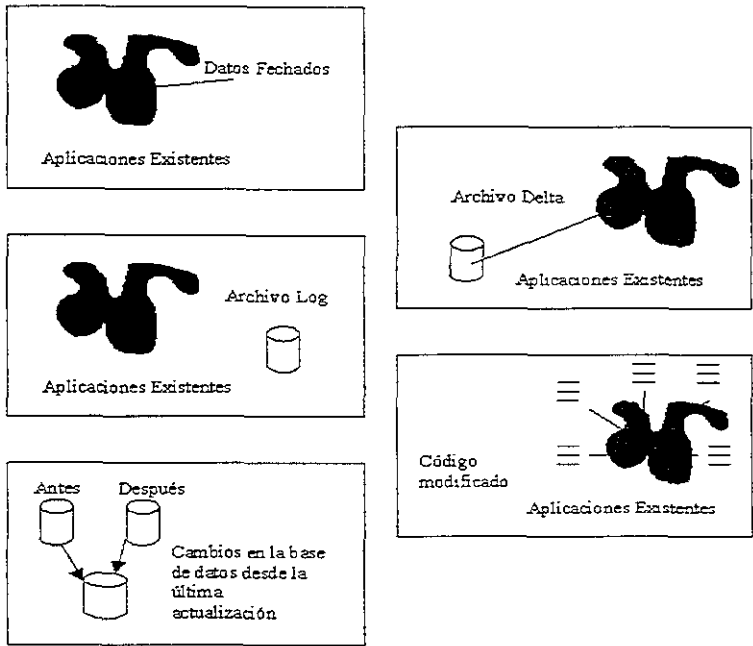


Figura 2.6 ¿Cómo sabemos que fuente de datos escanear? ¿El proceso será diario?, ¿Cada semana?, etc.

1. La primera técnica consiste en rastrear datos que han sido registrados y fechados, ya que para el DW es completamente eficiente rastrear cuando los datos ya no tienen movimientos o actualizaciones de registros.
2. La segunda técnica consiste en limitar los datos a rastrear por el DW rastreando exclusivamente un archivo "delta". El archivo delta es creado por una aplicación que contiene sólo los cambios hechos a la aplicación. Cuando existe este tipo de archivo, el proceso de rastreo se vuelve muy eficiente ya que muchos datos innecesarios en el DW nunca serán tocados. Lo malo es que no muchas aplicaciones construyen archivos delta.

3. La tercera técnica consiste en rastrear y auditar un archivo log. Este tipo de archivos contienen la misma información que los archivos delta pero con algunas mejoras. Muchas veces los procesos operativos protegen a los archivos logs, ya que son necesarios en los procesos de recuperación. Una dificultad es que el formato interno está diseñado para los propósitos del sistema, no de las aplicaciones. Esto requerirá de los servicios de un "Gurú" tecnológico para que desarrolle la interfaz entre los datos y el contenido del archivo log. Otro defecto es que frecuentemente el archivo contiene mucha más información de la deseada por el programador del DW.
4. La cuarta técnica consiste en modificar el código de las aplicaciones para que generen la cantidad de información necesaria y exclusiva que se requiera para el DW.
5. La última técnica consiste en la comparación de una imagen de "antes" y "después", en esta opción se toma una fotografía de la base de datos en el momento de la extracción. Cuando se realiza otra extracción, se toma una segunda fotografía. Ambas fotografías son comparadas para verificar la actividad que se ha tenido. Esta técnica es molesta y muy compleja además de requerir una gran cantidad de recursos.

Otra consideración a la hora de diseñar la extracción es la condensación de los datos, como se muestra en la figura 2.7.

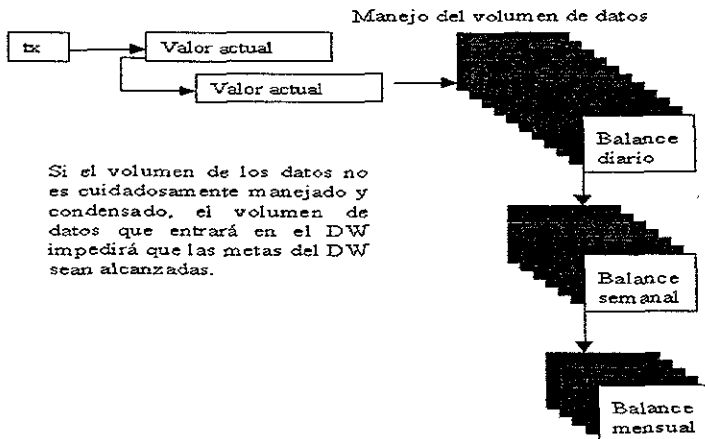


Figura 2.7 La condensación de los datos es un factor vital en el manejo del DW.

Complejidad de la transformación e integración

A primera vista, cuando los datos son movidos del ambiente operacional al DW, parece que lo único que habría que hacer es sacar los datos de un lugar para ponerlos en otro.

Debido a la sencillez de esta idea muchas organizaciones comienzan a fabricar manualmente su propio *DW*. El programador ve que hay que mover los datos de un sistema operacional antiguo a un *DW* moderno y dice "Lo puedo hacer" ya que es muy sencillo. Lo que no ha tomado en cuenta y que irá descubriendo conforme avance en su análisis es que existen diversos factores que hay que tomar en cuenta como son:

La extracción de datos de un ambiente operacional a un *DW* requiere de un cambio de tecnología. Esto incluye normalmente leer datos de tecnología DBMS (Database Management System, Sistema Manejador de Bases de Datos), como IMS y escribir los datos en una nueva tecnología DBMS, como Informix.

La selección de los datos del ambiente operacional puede ser muy compleja. Para evaluar el proceso de extracción de registros, se requieren muchos procesos de ordenamiento previos, así como lectura de las claves y la conexión lógica, etc.

Las claves operacionales deben ser leídas y reestructuradas antes de ser reescritas en el *DW*.

Los datos en la mayoría de los casos deben de ser cambiados de formato. Por ejemplo, si en el sistema operacional el formato de la fecha es *yy/mm/dd* y se requiere como *dd/mm/yy*. Otro ejemplo de cambio de formato que debe hacerse es de EBCDIC a ASCII, etc.

Los datos deben ser validados (limpiados). En algunos casos la aplicación de un algoritmo sencillo se aplica a los datos para corregirlos. En otros casos se requiere de un proceso más elaborado conocido como la aplicación de un *Middleware*.

La eficiencia en la selección de los datos para la extracción muchas veces viene a proporcionar un gran resultado.

Muchas veces se requiere la sumarización de algunos datos.

Los registros de entrada que se deben de leer pueden tener formatos "exóticos" o no estandarizados.

El diseño de los datos del *DW* debe de hacerse de acuerdo al modelo de datos corporativo.

En resumen, la información que debe elegirse para que sea extraída de los sistemas operacionales, debe ser minuciosamente elegida o en su caso reformateada para poder ser insertada dentro del *DW*. Por lo que el proceso de extracción debe definir una copia de datos uniforme, estandarizada y sencilla para que pasen al *DW*. Es por ello que se debe ser muy cuidadoso en el diseño de un método óptimo de extracción y si es posible para evitar la duplicidad de aplicaciones, eliminar los procesos obsoletos una vez que se libere el *DW*.

Especificaciones de extracción

La etapa de extracción en un *DW* tradicionalmente es un proceso de diseño. Es un flujo de datos que se alimentan desde sistemas operacionales y generan otros que llegarán al *DW*. Como sea, la clave de los procesos de extracción es: qué debe pasar para limpiar y transformar los datos en información útil dentro de un *DW* para que puedan ser utilizados para tomar decisiones.

2.2.3 Características de validación

Como se ha visto, la construcción de un *DW* comienza poniendo cuidadosa atención en su arquitectura y modelo de datos, y con la clasificación de sus componentes. Pero con los conceptos de extracción ya revisados no es suficiente para lograrlo, ya que antes de depositar la información extraída de los sistemas operacionales será necesario que pase por una serie de filtros que permitan la pureza de la información en el *DW*. Estos filtros se conocen como validación, depuración o limpieza de los datos extraídos, mediante un proceso que será conocido como *Middleware*, dicho proceso normalmente será una especie de caja negra donde entrarán datos y se les aplicará algún o algunos procesos y entregará datos al *DW*.

Tanto el proceso de extracción como el de validación deben ser sencillos, lo que debe permitir al proceso de extracción que rastrea determinar el quién, qué, cómo y dónde de los datos. Este tipo de información permite un algoritmo de extracción basado en el tiempo y el llenado de las tablas del *DW*. Cualquier diseño de transferencia de datos de un sistema operacional, debe de incluir estas características para facilitar dicha tarea.

Debido a que los procesos de extracción y limpieza implican un esfuerzo, tradicionalmente desarrollado en base a estándares como los procesos de modelado y utilizados para implementar el *DW* y sus procesos asociados, debe ser el trabajo principal del equipo de desarrollo, especialmente del administrador de la base de datos, para asegurar que tanto el proceso de extracción como el de limpieza que cargan los datos al *DW* de acuerdo a una arquitectura global.

Todo esto de acuerdo a las siguientes metas:

- Todos los datos son cargados en el *DW* de acuerdo a los estándares corporativos. Dichos estándares deben ser publicados y estar disponibles así como su significado en el repositorio de metadatos o en un manual electrónico.
- En el momento en que los datos del *DW* sean modificados, el inventario de datos de cada área deberá reflejar dichos cambios. Hay que recordar que el repositorio nos debe proporcionar la historia completa de una entidad

Lo que es importante resaltar, es que todos los procesos de validación dependerán única y exclusivamente de las necesidades de la organización, por lo que es primordial considerar todos los comentarios posibles de los usuarios a la hora de diseñar los procesos de validación.

- **Validación.** Comprobaciones de la calidad de los datos definidas por reglas empresariales, el modelo de *DW*, etc.; la congruencia y la integridad dominan la verificación.

2.2.4 Distribución de datos y carga

Distribución de datos

El proceso de distribución puede ser similar al trabajo de una persona que entrega periódicos. Después de que el periódico es impreso y se hace disponible en un centro de distribución, la persona que va a entregar el periódico, va con los suscriptores y les entrega unas copias del periódico. El concepto es el mismo en un *DW*. Los sistemas alimentan a un *DW* centralizado. El *DW* debe ser suministrado por los subsistemas con datos correctos desde los procesos de extracción, validación y carga.

Un grupo de utilerías estándar en la arquitectura en general puede ser implementada para soportar este concepto de distribución. De nueva cuenta, hoy en día el software moderno para replicación es inestimable en esta área. Sin embargo, se va a encontrar que esto no es una clara solución al problema de distribución de datos. Por lo tanto, hay que asegurarse de que las utilerías puedan trabajar en armonía con el tiempo asignado para el desempeño de la distribución de los datos y para hacer que estos estén disponibles a todos los usuarios. Estas utilerías tienen el movimiento de la carga física de los datos de un *DW* al consumidor de datos, el cual puede incluir usuarios, reportes estándares, análisis de sistemas multidimensionales o *datamarts*.

La tecnología y arquitectura de los datos pueden garantizar que la distribución de los datos logren lo siguiente:

- Asegurar que la distribución de los datos ocurra puntualmente y de manera eficiente.
- Asegurar que sólo los datos son ordenados y entregados
- Establecer servicios apropiados y requeridos para el nivel estándar.
- Obtener datos para proveer evidencias estadísticas al servicio del nivel estándar.

Carga de datos

En el proceso de carga de un *DW* existen básicamente dos tipos de carga:

- La carga inicial de los datos existentes en el ambiente operativo.
- La carga de los cambios que han ocurrido sobre el nivel operativo desde la ejecución del proceso de carga anterior.

El primero de ellos no reviste mayores problemas, salvo los concernientes a asegurar la integridad de los datos, pues sólo debe ser realizado una vez. Generalmente se realiza una recorrida sobre los datos operativos en forma secuencial, se aplican las transformaciones necesarias y se almacenan en las estructuras de la *DW*.

Por el contrario, la captura de las modificaciones que han ocurrido a nivel corporativo es una tarea mucho más compleja que la carga inicial. Para lograr esta tarea existen una serie de técnicas (descritas en el punto 2.2.2 Características y problemática de la Extracción) tendientes a identificar cuáles son los nuevos datos que se deben migrar hacia el *DW*. Esta tarea debe ser realizada de la manera más eficiente posible, y para ello es crítica la cantidad de datos que se deban procesar para lograr identificar aquellos que han sido identificados.

Incorporación del tiempo

Como forma de medir los cambios que ocurren en las distintas actividades de una organización, los datos deben ser analizados bajo una perspectiva histórica. Cuando se realiza un análisis de alguna actividad del negocio, lo más conveniente es tener grupos de transacciones que permitan tener una idea global de la evolución de dicha actividad, en vez de abocarse al estudio del detalle de cada una de las transacciones. Sin embargo, el tiempo que transcurre entre las distintas transacciones es variable, debido a que las mismas ingresan al sistema de una manera personalizada. Por lo tanto, la unidad de tiempo que se seleccione para agrupar las transacciones debe ser uniforme que habilite la comparación directa entre los distintos periodos. Estos periodos no deben ser elegidos de una manera arbitraria, sino que deberán tener como base los periodos con los cuales el nivel gerencial analiza el rendimiento de las actividades del negocio

La definición de la unidad de tiempo determina además, la frecuencia con que se deberán ejecutar los procedimientos de carga. Esta frecuencia debe ser cuidadosamente respetada, pues los datos operativos son válidos en el momento actual y no necesariamente reflejan una situación pasada. A modo de ejemplo, si se desea almacenar el estado de cuenta de un cliente en forma mensual, él mismo deberá ser tomado siempre con la misma frecuencia, supongamos todos los días primero del mes; pues de realizarse la migración del mismo unos días después éste podría haber cambiado.

El último aspecto involucrado con la incorporación del tiempo, se refiere a los casos en los que se opta por mantener varios niveles de granularidad. En estos casos se deberán definir además, cuales son los algoritmos que permiten la conversión de una unidad a

otra, es decir como los días se convierten en semanas, estos en meses los meses en cuatrimestres, años, etc., o lógica asociada a las conversiones de otras líneas de análisis

Mecanismos de carga

La última tarea a desarrollar en el proceso de carga del *DW* es la relativa a la determinación del mecanismo que se utilizará para realizar la misma, de acuerdo al nivel de granularidad que se haya seleccionado. Existen dos formas básicas de desarrollar esta tarea:

Acumulación simple. Es sin duda la más sencilla y común, y consiste en realizar una sumariación o resumen de todas las transacciones comprendidas en el periodo de tiempo seleccionado y transportar el resultado como una única transacción hacia el *DW*.

Rolling. Este proceso se aplica en los casos en que se opta por mantener varios niveles de granularidad. Para ello se almacena información sumariada a distintos niveles, correspondientes a distintas agrupaciones de la unidad de tiempo.

Por ejemplo, si se determina el día como unidad básica de tiempo, las transacciones diarias son sumariadas y almacenadas a nivel diario, correspondiente al nivel de detalle más alto. Al llegar al límite de los valores válidos de la unidad, en nuestro ejemplo al completar una semana, se resumen todos los registros del nivel y se crea el correspondiente al siguiente nivel de detalle, en nuestro ejemplo el registro semanal, y se inicializa nuevamente el nivel diario. Al completar el nivel semanal se sumariiza y se genera el nivel mensual reinicializando el nivel semanal, y así sucesivamente hasta llegar al último nivel de detalle seleccionado donde se aplican los criterios de acumulación simple. Utilizando este mecanismo, se obtiene un alto nivel de detalle para la información más reciente y a medida que la antigüedad de la misma va creciendo, decrece el nivel de detalle mantenido. Este enfoque está acompañado con la mayoría de las consultas estratégicas, las cuales generalmente involucran un alto nivel de detalle para analizar el comportamiento más reciente y requieren menos detalle cuando se analizan periodos menos recientes.

Una variante de este mecanismo, consiste en la realización de las actualizaciones a todos los niveles en forma simultánea. Es decir que las transacciones diarias se sumariizan, creando un registro del primer nivel y en forma simultánea se actualizan los registros correspondientes a los niveles de detalle más bajos. En nuestro ejemplo, la sumariización diaria generaría un nuevo registro diario, actualizando simultáneamente el registro semanal, el mensual y el anual que corresponda. Si bien esta variante permite tener reflejada en todo momento la misma información a todos los niveles, tiene el inconveniente de que, salvo para el caso de más bajo nivel de detalle, los otros niveles poseen la acumulación hasta la fecha. Esto significa que hasta no completar el periodo la información está incompleta.

2.2.5 Diseño de Base de Datos

Modelo de datos

Una vez realizado el análisis de requerimientos de la empresa, obtenido las áreas objeto de interés del negocio, identificada el área principal del negocio y descubiertas las consultas de interés para esta área, procederemos a modelar toda esta información.

Existen dos métodos y cualquiera de los dos puede ser utilizado para el modelado de datos del *DW*: modelo estrella y el modelo multidimensional. Es práctico asumir que el *DW* deberá ser modelado utilizando un modelo de datos relacional. Nosotros necesitamos identificar las áreas objeto (entidades en el lenguaje normal de base de datos) las cuales deberán existir en el modelo de datos.

Los sistemas operacionales implementan un modelo de datos en donde las principales entidades del negocio son modeladas, para el *DW* se crean áreas objeto que consolidan información de las fuentes operacionales. En un *DW* no deben existir muchas áreas objeto, sólo se deben mostrar las más importantes. Un análisis iterativo debe concentrarse en un área y demostrar el valor de cada una de ella.

El Modelo de Estrella

El modelo de estrella mejora las consultas basadas en las actividades del negocio contra el modelado tradicional de base de datos quien normaliza los modelos. El esquema de normalización contiene entidades y relaciones entre ellas. Sin embargo, este esquema mantiene una estructura irregular para el procesamiento de las consultas. Por el contrario, el modelo de estrella define entidades que están dirigidas a la toma de decisiones del negocio, así como también, las entidades reflejan la importancia de los aspectos operacionales del negocio. Un modelo de estrella optimiza las consultas y por lo tanto proporciona un diseño de base de datos que está enfocado en la rápida respuesta para los usuarios del sistema. También el diseño que es construido desde el modelo de estrella no es tan complicado como el diseño tradicional de base de datos. El modelo estrella lleva este nombre ya que visualmente tiene esta forma. El verdadero poder del modelo de estrella está basado en las estructuras de datos que permiten filtrar o reducir el tamaño de los resultados de las entidades importantes durante las consultas de los usuarios y sus búsquedas. Un modelo estrella proporciona un entendimiento y aprovechamiento de las estructuras de datos, por que los puntos de la estrella o dimensiones proporcionan un mecanismo por el cual un usuario puede realizar *drill down*.

Para crear el modelo estrella hay que categorizar cada uno de los atributos del área objeto en dos rubros:

- Dimensiones
- Métricas

Una dimensión es un atributo por el cual una consulta puede ser generada, mientras que, una métrica es un atributo del cual nosotros podemos derivar un valor numérico. Por ejemplo, imagine los siguientes atributos del área objeto relativo a ventas:

Store Id
Product Id
Date of Sale
Number of Units
Sale Price
Color
Payment Type

Figura 2.8 Área objeto de ventas.

La dimensión más común en un modelo estrella es la de tiempo (atributo *Date*), pero los otros atributos dimensiones son: *Store Id* y *Product Id*. Los atributos *Payment Type* y *Color* también pueden ser considerados dimensiones. Las métricas son los atributos *Number of Units* y *Sale Price*.

Así mismo al identificar las dimensiones simétricas probablemente no podamos contestar muchas consultas con estos atributos, pero si nosotros creamos tablas de dimensiones podemos comenzar refinando nuestro modelo. Como se muestra en la figura 2.9:

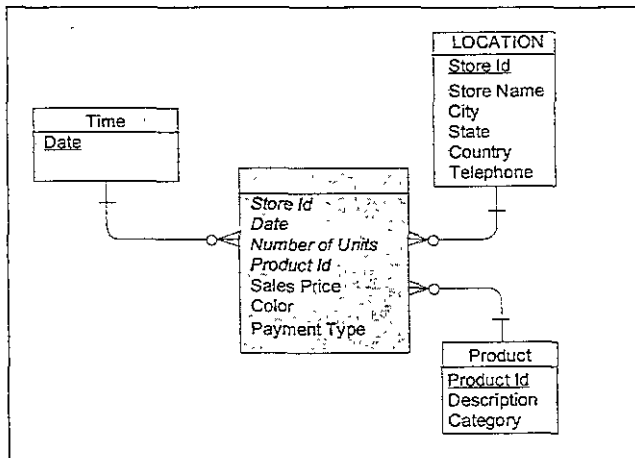


Figura 2.9 Modelo Estrella.

El diagrama que resulta es conocido como el modelo estrella, pero éste puede ser reconocido por lo que realmente es: un modelo conceptual de datos. Las tablas de dimensión guardan información normalizada acerca de los atributos de dimensión.

Cuando dibujamos el modelo de estrella, el área objeto adquiere el nombre de *tabla facto*, debido a que en ella se guardan las métricas relevantes.

El modelo estrella puede ser utilizado para comenzar a obtener requerimientos. ¿Qué tipo de preguntas puede contestar este modelo? Si nosotros vemos las tablas de dimensiones, podemos comenzar con un clásico ejemplo comparando el número de unidades vendidas (*Number of Units*, una de las métricas en la *tabla de facto*) este mes con el número de unidades vendidas el mes anterior. Otro ejemplo puede ser un análisis entre el precio de venta y el costo por unidad de cada proveedor, identificando que proveedores están contribuyendo más en nuestras ganancias.

Una vez que se genera el modelo físico del modelo estrella, las tablas de facto almacenan la mayoría de los registros de información, mientras que las tablas de dimensión tienden a tener relativamente pocos registros. El realizar el análisis de la información generalmente involucra *joins* entre una gran tabla (*tabla de facto*) y una o más tablas de dimensión, lo cual tiene el beneficio de regresar resultados rápidamente.

De hecho es posible derivar tablas de dimensiones de las tablas de dimensiones existentes. Por ejemplo la tabla *location* puede ser normalizada en *Store* y *Region* y la dimensión *product* en *Product* y *Product Category*. Esta categorización de dimensiones mantiene relaciones 1 a N entre las tablas involucradas. El modelo formado es conocido como *Snowflake Schema* (Modelo copo de nieve) y a las categorizaciones de las dimensiones se les llama jerarquías (ver figura 2.10). Las jerarquías permiten realizar *drill down* (navegar hacia adelante) o *roll up* (navegar hacia atrás).

Nosotros podemos sugerir también que los atributos *Payment Type* y *Color* pueden ser dimensiones. ¿Puede Color ser una métrica? El argumento sería el siguiente: "Yo podría contar el número de colores distintos". La respuesta es NO, ya que la métrica es "*count (distinct color)*", "no Color por sí mismo."

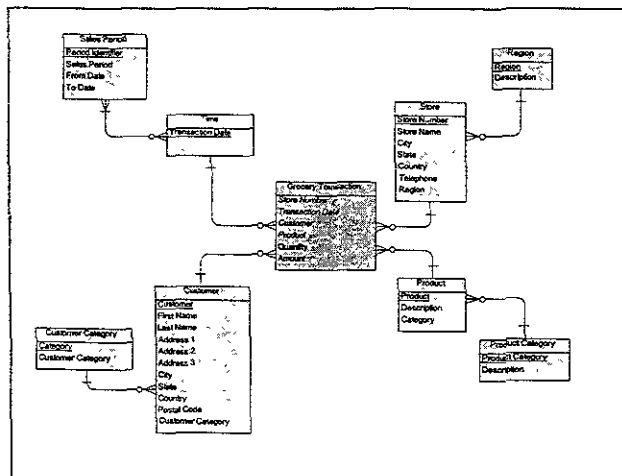


Figura 2.10 Modelo *Snowflake*.

Modelo Dimensional

El modelado dimensional es otra técnica utilizada en el modelado de *DW*. Como mencionamos en la sección anterior, una dimensión es un atributo por el cual una consulta puede ser formulada.

Una simple forma de modelado dimensional es construir un mapa de dos dimensiones, con esta técnica se van a construir matrices para capturar las consultas necesarias que utilicen ambas dimensiones.

Considere la siguiente consulta: ¿Cuántos productos fueron vendidos en una región particular para un tiempo específico? El tiempo es siempre una dimensión, de hecho podemos tomarlo fuera de este análisis, ya que siempre será considerado; por lo tanto en este ejemplo podemos construir una matriz con producto y localización geográfica en los dos ejes. Esta técnica nos permitirá determinar las intersecciones de las dimensiones que son más importantes y aquellas que no tienen sentido para nuestro negocio.

Este mapa nos ayuda a identificar las consultas del negocio más importantes y nos ayuda a saber como se utiliza la información. Una forma más rigurosa del análisis de información es crear un cubo de análisis, el cubo de análisis es más matemático en su naturaleza. Como en el tema anterior comenzaremos categorizando el área objeto en dimensiones simétricas, entonces nosotros dibujaremos un cubo N dimensional donde tendremos una dimensión por cada atributo de dimensión.

En la figura 2.11 se puede observar 3 atributos (*Product lines*, *Region*, *Time*) los cuales representan las dimensiones y una de las dimensiones que se utiliza en este tipo de modelos es el tiempo. Se pueden obtener consultas como, ¿qué producto fue el más vendido en el mes pasado, en la región del Centro?

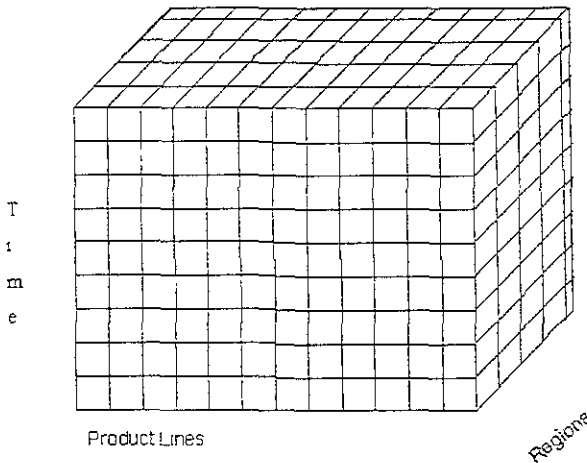


Figura 2.11 Modelo Multidimensional.

2.2.6 Metadata

Capa de administración de metadatos

La arquitectura del *DW* se basa en el concepto de definiciones de datos, o *metadatos*. Los *metadatos* invaden todas las actividades del *DW*. Las fuentes de información se caracterizan por la definición de los datos que llegan. Imprimir la fecha requiere definir *metadatos* adicionales relacionados con tal proceso. Imprimir la fuente también crea nuevos *metadatos*. La actividad de condensación requiere crear nuevas columnas para contener la información resumida. La capa de "Administración de *Metadatos*" de la arquitectura de referencia es responsable de la administración de los *metadatos* que usa el *DW*, tales como la descripción completa de los datos almacenados en el *DW*.

El *DW* y los modelos y esquemas de datos lógicos y físicos del mercado de datos, junto con el glosario técnico y empresarial, también se almacenan y administran en esta capa. El gran reto de administrar bases de datos muy grandes con su complejidad en las áreas de índices múltiples, compactación de datos, claves compuestas y las versiones de los datos se solventa y administra en esta capa.

La capa de Administración de *metadatos* tiene las siguientes responsabilidades:

- Las definiciones estándar de los datos (incluyendo las definiciones técnicas y empresariales) de los datos depositados en el *DW*.
- Los *metadatos* capturados y creados en los bloques de refinamiento y reingeniería.
- Los *metadatos* acerca de granularidad, segmentación, áreas temas, adición y condensación.
- Los *metadatos* que describen las consultas y reportes predefinidos y diseñados.
- Los *metadatos* describen índices y perfiles que mejoran el rendimiento en el acceso y la recuperación de datos.
- Los *metadatos* describen las reglas para preparar y programar el ciclo de actualización y duplicación.

La importancia de la administración de los metadatos

Como antes se mencionó, los *metadatos* se definen como datos acerca de los datos. En una base de datos, los *metadatos* son la representación de los diversos objetos que definen una base de datos. En una base de datos relacional, esta representación se constituiría en las definiciones de tablas, columnas, base de datos, visualización y otros

objetos. En un sentido más amplio (y para los fines de esta sección), usaremos el término *metadatos* para hacer referencia a todo lo que defina un objeto del *DW*—ya sea una tabla, una columna, un reporte, una consulta, una regla empresarial o una transformación dentro de un *DW*-. Esta amplia definición nos permite abarcar las definiciones de todos los objetos significativos dentro del *DW*.

La comprensión de estas definiciones es esencial para todos los aspectos del desarrollo del *DW*—desde el desarrollo de programas de extracción de las bases de datos fuente que alimenta al *DW*, hasta la transformación de datos de múltiples bases de datos, para que puedan almacenarse en un formato común dentro del *DW*-.

¿Qué son los metadatos?

Los *metadatos* impregnan todos los aspectos del *DW* y constan de los siguientes tipos de elementos:

- Ubicación y descripción de servidores, bases de datos, tablas, nombres y resúmenes del *DW*.
- Reglas para la profundización automática al detalle o al resumen y a través de jerarquías de dimensión empresarial, tales como productos, mercados y cuadros contables.
- Nombres elegidos o alias definidos por el usuario final para los encabezados y hechos de datos con nombres más técnicos.
- Reglas para cálculos personalizados definidos por el usuario final.
- Seguridad a nivel personal, de grupo de trabajo y de empresa, para visualizar, cambiar y distribuir resúmenes adaptados, cálculos y otros análisis de usuario final.
- Descripciones de fuentes originales y transformaciones.
- Definiciones lógicas de tablas y atributos del *DW*.
- Definiciones físicas de tablas y columnas, así como de sus características.
- Ubicación integrada de las tablas del *DW*.
- Antecedentes de extracción.
- Información de alias.
- Algoritmos de resumen.
- Ubicación de área tema.
- Antecedentes de relaciones.
- Propiedades/Gerencia.
- Patrones de acceso.
- Tablas de referencia y datos codificados.
- Criterios de envejecimiento y purga.
- Indicador de calidad de datos.
- Seguridad.
- Unidades de medida.

En forma adicional, los *metadatos* pudieran también contener componentes de ubicación para auxiliar en las siguientes tareas:

- Identificación de fuentes operacionales.
- Ubicación sencilla de atributo a atributo.
- Conversiones de atributos.
- Conversiones de características físicas.
- Conversiones de codificación y tabla de referencia.
- Cambios de nombre.
- Cambios de llave
- Valores predeterminados que se utilizan.
- Razón predeterminada.
- Lógica para elegir entre varias fuentes operacionales.
- Formulación algorítmica empleada.

La figura 2.11 muestra un ejemplo de un elemento de *metadatos*, en donde se define desde el punto de vista empresarial de una entidad, tal como el punto de contacto y los sinónimos de uso común.

Nombre de la entidad	Cliente
Nombres alternos	Cliente
Definición	Un cliente es una persona o empresa que ha comprado a la corporación bienes o servicios por lo menos en una ocasión
Fecha de creación	15 de enero de 1995
Fecha de última modificación	15 de abril de 1996
Llave(s)	Identificador de cliente, ubicación del cliente.
Ciclo de actualización	Extraído cada mes
Ciclo de archivado	Archivado después de seis meses.
Propietario de datos	Bill Doe
Patrones de acceso	Fecha de último acceso 30 de mayo de 1996

Figura 2.11. Los metadatos para una entidad (o registro de archivos o tabla).

La figura 2.12 muestra un ejemplo de *metadatos* de atributos. Los atributos también contienen información sobre las características físicas necesarias para conversiones. Además, contiene una definición empresarial, una lista de sinónimos, antecedentes sobre su creación y parámetros relacionados con el *DW*, tales como información de transformación y conversión.

Nombre del atributo	Nombre del cliente.
Nombre alterno	Nombre de cuenta, nombre de cliente.
Definición	Un cliente es el nombre oficial que usa la persona o empresa cliente.
Fecha de creación	30 de marzo de 1993.
Fecha de última actualización	5 de abril 1995.
Indicador de llave	N (Sin llave)
Fuente de datos	Sistema de ingreso de Información de pedidos, el atributo archivo de Cliente es Num_Cuenta.
Transformación/conversión	De 20 a 35 caracteres
Algoritmo de resumen/derivación	Ninguno.
Valor predeterminado que se usa	Ninguno.
Fuentes operacionales múltiples	No

Figura 2.12. Los metadatos para un atributo (o columna o campo)

Importancia de los metadatos

Los *metadatos* son como el mapa de carreteras hacia los datos. En forma muy parecida a la que una ficha de catálogo de biblioteca apunta tanto al contenido como a la ubicación de los libros de la biblioteca, los *metadatos* apuntan a la ubicación y al significado de información diversa dentro del *DW*.

El *DW* debe mantener un catálogo de los artículos que maneja. Los usuarios finales son como clientes, generan solicitudes de información con base en una selección que hacen de un catálogo. El proceso que llena la solicitud debe saber en dónde se ubica la información dentro del *DW*.

Por lo tanto, el *DW* debe contener un componente que satisfaga las funciones del catálogo para la información que maneja. Este catálogo debe estar organizado para suministrar las siguientes necesidades:

- Debe servir como un mapa de las ubicaciones en donde se encuentra almacenada la información en el *DW*. Este mapa es importante para quienes desean saber qué está guardado en el *DW*. Además, es importante para las aplicaciones empresariales que usan el *DW*, para guardar su información. También es valioso para el personal de desarrollo, quienes deben escribir programas que consulten el *DW* o programas que integren herramientas de consulta y reportes con el *DW*. Con frecuencia, el catálogo es el esquema de la base de datos del *DW*. Debido a que la mayoría de los *DWs* utilizan un RDBMS, el esquema contiene elementos como las definiciones de tablas,

columnas visualizaciones, índices, restricciones, consultas, procedimientos almacenados, activadores, políticas y bases de datos.

- En forma óptima, el catálogo del DW debe tener dos componentes de definición para cada elemento:
 - La definición que requiere la tecnología de base de datos (tal como el nombre de la tabla, el propietario de la tabla, el tipo de tabla, los nombres de columnas, los tipos de datos y los valores predeterminados).
 - La definición que requiere un usuario empresarial. Esta definición empresarial está enfocada hacia términos y definiciones empresariales comunes y no está sujeta a los límites impuestos por la tecnología (como el límite de longitud, la incapacidad de usar el espacio, los caracteres no alfanuméricos, etc.).
 - Debe proporcionar un anteproyecto de la forma en la que un tipo de información se deriva de otra. El resumen es un ejemplo de dicha derivación. Los datos históricos de los últimos diez años pudieran presentarse resumidos por año, mes o trimestre para apoyar las necesidades de análisis. La incorporación de una dimensión de tiempo a los datos operacionales e históricos también debe capturarse en el catálogo como definiciones de *metadatos* con dominios de tiempo. Otras dimensiones que se incorporen a los datos de entrada también deben representarse mediante los *metadatos* adecuados.
 - Debe proporcionar un anteproyecto de mecanismos que extraigan datos de las aplicaciones empresariales operacionales y los depositen en el DW. Los extractores son responsables de comprender la organización de los datos de entrada y de organizar la base de datos objetivo en la que se almacenarán los datos, de hacer conversiones de formato, proporcionar valores predeterminados o truncar y redondear información
 - De almacenar las reglas de control de acceso y seguridad para ofrecer una administración confiable. Los controles de acceso y seguridad dependen de permisos que se conectan a los *metadatos*.
 - Los *metadatos* deben registrar los cambios a través del tiempo. Con frecuencia, la organización de los datos que reúne una corporación a través de los años también cambia. Los *metadatos* de clientes recopilados de 1985 a 1990 pudiera diferir en organización a los reunidos de 1991 a 1995 debido a los cambios realizados en la base de datos y en los sistemas de aplicaciones que manejan el procesamiento de pedidos. Por ejemplo, cuando la corporación se fusionó con otra, los sistemas de información de ambas se integran durante un período de tres años, dando por resultado una organización diferente de los datos.
-

- Los *metadatos* deben registrar la fecha de su creación para capturar su *cambio histórico*.
- Se requiere almacenar la estructura y el contenido del *DW*.
- Se debe definir con claridad e identificar formalmente el sistema de registro (o fuente, por lo regular las aplicaciones patrimoniales) para el *DW*.
- Como una parte regular del *DW*, debe estar disponible la lógica de transformación e integración que transfiere los datos del ambiente operacional al ambiente del *DW*.
- Como parte de los *metadatos*, debe almacenarse la historia de actualizaciones, para permitir a los usuarios determinar su periodicidad y precisión.
- Se requiere almacenar un esquema de medidas de modo que el usuario final pueda determinar si una consulta será larga o corta antes de proponerla.

Importancia de los *metadatos* durante el desarrollo del Data Warehouse

Durante el desarrollo del *DW*, los *metadatos* se emplean en los componentes siguientes:

- Extracción de las fuentes de datos.
- Actualización de datos.
- Transformación de fuentes de datos.
- Pulido de las fuentes de datos.
- Resumen y adición de las fuentes de datos.
- Diseño de la base de datos del *DW*.
- Diseño de consulta y reportes

Si el proceso de captura de *metadatos* se realiza con cuidado durante estas actividades, los *metadatos* están *automáticamente disponibles* para los usuarios finales durante la etapa de despliegue del *DW*.

El diseño de *metadatos* para un *DW* es un giro dramático en el paradigma para los analistas de información que analizan y diseñan bases de datos para apoyar sistemas operacionales. El enfoque durante el diseño de una base de datos operacional consiste en crear modelos de datos con normas y atributos de datos atómicos. Hasta ahora, la principal preocupación era eliminar la redundancia de datos mediante la aplicación de normas. El objetivo de eliminar la redundancia de datos es evitar problemas de actualización y mantener la consistencia de los datos.

El diseño de *metadatos* para el *DW* requiere de un cambio completo de mentalidad. En este caso, el enfoque consiste en representar un amplio rango de relaciones para el analista, a menudo con mucha redundancia. Debido a que las actualizaciones no son un

aspecto principal dentro del *DW*, no se asocia ningún precio a la redundancia de datos, salvo la sobrecarga de almacenamiento. Con frecuencia se obtienen economías de procesamiento realizando uniones y almacenándolas como tablas. El cambio de mentalidad para el analista consiste en adaptarse de una escasa disponibilidad de datos redundantes para contar con una amplia gama de información.

Otro cambio en el paradigma para el analista de información es el énfasis de los sistemas operacionales sobre los *metadatos* actuales. La mayoría de las aplicaciones operacionales sólo funcionan con la organización actual de la base de datos y los datos. Los datos viejos se archivan junto con versiones anteriores de la organización de la base de datos operacionales, debido a que deben emplearse los *metadatos* para extraer datos históricos. Las versiones de *metadatos* dentro del *DW* son una consideración importante.

Un aspecto importante de los *metadatos* es la necesidad de ubicar con mapas desde las fuentes hasta el *DW* a través de un proceso de extracción, refinamiento y reingeniería. Se deben mantener estos mapas para los fines siguientes:

Verificación de la calidad de datos. Los mapas contienen información sobre los diversos cambios que han sufrido los datos antes de integrarse al *DW*. Esta "auditora" es muy importante si se quiere que sean precisas las decisiones que se hacen con base en el análisis e interpretación de los datos del *DW*.

Sincronización y actualización. Conforme se genera nueva información operacional y se debe actualizar en el *DW*, los nuevos datos deben experimentar las mismas transformaciones que los datos cargados antes en el *DW*. Por lo tanto, mantener los mapas y algoritmos para la transformación es esencial para repetir el proceso de transformación sobre los datos de actualización.

Integración. Los mapas establecen relaciones entre los datos que reflejan las reglas empresariales de interés para los usuarios finales. Sin los mapas, al usuario final se le presentan piezas aisladas de información que no sirven para sustentar decisiones.

La figura 2.13 muestra los mapas que mantienen los *metadatos* acerca de los datos durante su viaje a través del *DW*.

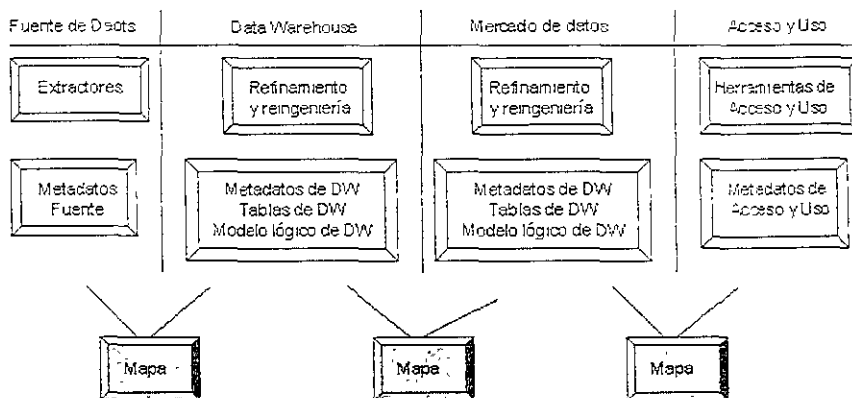


Figura 2.13 Los diferentes tipos de *metadatos* que se crean, almacenan y deben mantenerse para cada uno de los bloques del *DW*.

La siguiente sección muestra cómo se utilizan los *metadatos* para actividades de desarrollo de un *DW* de muestra.

Extracción de fuentes de datos

En la figura 2.14 se muestra la función de los *metadatos* en la arquitectura de referencia del *DW* para el bloque de Fuentes de Datos. Los *metadatos* del bloque de Fuentes de datos, tienen que ver con la definición de las bases de datos que alimentan al *DW*, así como con la definición de datos de elementos del *DW* extraídos de oficina y fuentes externas.

La función de los *metadatos* es fundamental para la tarea de integrar los datos de múltiples fuentes. Algunos de los retos que se encuentran durante la integración se describen en la siguiente lista:

Identificación de campos fuente. Los datos fuente están contenidos en una variedad de tecnologías, desde sistemas basados en archivos hasta bases de datos relacionales. Con frecuencia, los nombres de los campos de información están encriptados y es un reto para comprenderlos. El uso de un diccionario o una fuente autorizada de definiciones de los campos resulta muy valioso para identificar cuáles campos deben cargarse en el *DW*.



Figura 2.14 Los *metadatos* del bloque de Fuentes de datos.

Registro de cambios históricos en la organización de los datos. Los cambios en los sistemas de aplicación, la organización de bases de datos, o las fusiones y reorganizaciones son por lo regular responsables de estos retos. En tales casos, es necesario desarrollar una organización de datos consistente dentro del *DW* (objetivo), y realizar luego las conversiones de datos de los diversos formatos a este objetivo. Entre los cambios en la organización están modificar la longitud de los atributos, los tipos de datos, los esquemas de codificación y los campos llave.

Aplicación de valores predeterminados de manera inteligente, para los campos de datos que a propósito o en forma inadvertida no se registraron. La aplicación de valores predeterminados debe hacerse en forma consistente, de modo que el *DW* no contenga información inconsistente. Esta aplicación implica suministrar repetidamente el mismo conjunto de valores predeterminados, bajo las mismas circunstancias.

Resolución de inconsistencias en los esquemas de codificación, Con frecuencia, diferentes sistemas de aplicación utilizan distintos esquemas de codificación para representar los datos. Una aplicación pudiera usar una sintaxis M/F para representar el género sexual, mientras que otra utiliza 0/1 para la misma representación. No importa realmente qué codificación se use dentro del *DW*, siempre y cuando sea consistente y única. El proceso de extracción de datos debe considerar la conversión de esquemas a este único y consistente esquema de codificación.

Mapas de atributo a atributo. Los campos similares de diversas fuentes deben ubicarse juntos, de manera que los datos de estos campos se carguen en el mismo campo objetivo dentro del DW. Para ello, es necesaria la información de *metadatos* sobre atributos que indique cuáles campos deben ubicarse juntos.

Conversiones de atributos. A menudo, existen campos de información de entrada en diferente formato (longitud y tipo), provenientes de varios campos de datos. Se requieren *metadatos* para indicar el formato de cada uno de los campos de entrada. Se definen conversiones con base en la observación del formato de los campos de entrada y el formato del campo objetivo que deben alimentarse dentro del DW. Estas conversiones modifican los datos a fin de hacerlos compatibles para su carga en el DW. Las conversiones comunes son el truncamiento, el complemento y el redondeo.

Como un producto colateral de la recolección, transformación y desplazamiento de datos desde las bases de datos fuente hacia el DW, se capturan y almacenan los siguientes tipos de *metadatos*.

- Descripción completa de esquemas de fuente y objetivo.
- Fuentes detalladas para la ubicación de objetivos.
- Variables de conversión definidas durante las transformaciones.
- Reglas de transformación de datos.
- Opciones de recuperación de datos.
- Opciones de manejo de excepciones.
- Pasos para conversiones.
- Reportes de conversiones.

Refinamiento y reingeniería de datos. El bloque de Refinamiento y reingeniería de datos es el responsable de depurar los datos de las fuentes, agregar registros de fuentes y de fecha, transformar los datos para que coincidan con la organización de DW, y calcular con anticipación los valores de datos resumidos y derivados.

Integración y segmentación. Durante el paso de segmentación, una sola pieza de datos de entrada se divide en dos o más piezas dentro del DW. La necesidad de segmentar surge cuando los sistemas operacionales almacenan en una tabla, la cual sería mejor separar en varias en el DW—con frecuencia por razones de desempeño. Es deseable que estén separados los conceptos que contiene el objeto de datos.

Se debe emplear los *metadatos* para los datos de entrada a fin de derivar dos o más piezas de *metadatos* para el DW objetivo. Enseguida se desarrollan programas para dirigir los datos de entrada hacia varias ubicaciones destino. Otra razón para la separación es dividir de manera flexible un conjunto de datos en varias secciones, lo que permite diferenciar formas de análisis. Ejemplos de segmentación desde esta perspectiva son la separación de los datos por fecha, por línea de negocio, por geografía o por unidad organizacional. Los *metadatos* indican los atributos con un conjunto dado de datos.

La figura 2.15 recapitula los diversos pasos en el Refinamiento y la Reingeniería del DW. Estos bloques crean nuevos *metadatos* relacionados con la incorporación de registros de fecha y fuentes.

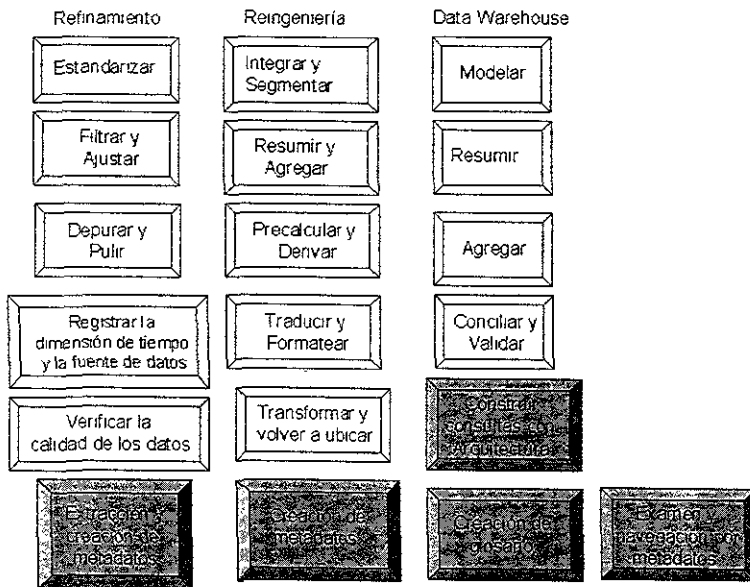


Figura 2.15 Bloques de Refinamiento y Reingeniería de un DW.

Resumen y adición. La forma más simple de resumen es la formación de estructuras acumulativas. Esto se hace simplemente sumando las cifras de varios atributos. Por ejemplo, el pedido por cliente por día es el total de todos los pedidos hechos por un cliente específico en un día determinado. Esta información se obtiene buscando todos los pedidos de ese cliente en ese día, y acumulando después en un nuevo campo la cantidad total. Por lo tanto, el proceso de resumen agrega nuevos campos a los datos que necesita para contener totales resumidos. Al mismo tiempo, la indicación de cuáles campos totalizar, cómo debe conformarse el total y en dónde debe almacenarse es una especificación de un proceso de resumen. Esto también debe almacenarse como *metadatos* dentro del DW para permitir la repetición de este proceso.

Entre los más complejos están los resúmenes rotatorios, como la acumulación de pedidos por semana, o por mes y cliente, que se derivan de los pedidos por cliente y por día. Cada paso del resumen da como resultado la incorporación de más campos, así como la necesidad de almacenar las reglas para el paso de resumen.

Cálculos previos y derivaciones. Los cálculos previos y las derivaciones son cálculos que se aplican al DW sin intervención o solicitud del usuario. Estos se calculan, almacenan y se ponen a disposición como campos de datos dentro del DW. Los cálculos previos y las derivaciones crean campos de datos adicionales. También deben manejarse como *metadatos* dentro del DW, los algoritmos que se emplean para hacer cálculos y derivar nuevos campos a partir de campos existentes. Por ejemplo, se pudiera hacer el

cálculo previo del pago anual y las deducciones a partir del pago semanal, el IVA y otros impuestos. Los cálculos previos de los salarios anuales podrían utilizarse para que los usuarios finales hicieran análisis sin realizar los cálculos cada vez que se requieren.

Transformación y reubicación. Las fuentes de los datos operacionales que proveen de datos al *DW* están organizados, por lo regular, como tablas con normas o tablas relacionales en cierta forma con normas. Por lo general, el esquema necesario para el análisis es un esquema de estrella, en donde una tabla central de hechos se une con varias tablas auxiliares. Las transformaciones y reubicación es el proceso de convertir la información de las fuentes de datos en hileras adecuadas para alimentar las tablas de hechos del *DW*. El proceso de reubicar implica ensamblar las hileras de la tabla de hechos a partir de múltiples tablas. La idea es que el precio que se paga una vez durante el ensamble de estas hileras se recupera muchas veces durante el análisis

Por lo tanto, el proceso de transformación y reubicación utiliza dos organizaciones o esquemas de bases de datos –la organización de los datos de entrada y la organización del esquema en estrella del *DW*. Con base en los mapas entre esquemas, se generan programas automáticos de transformación.

Acceso y uso

A continuación se muestra en la figura 2.16 la función de los *metadatos* en el bloque de Acceso y Uso del *DW*, que consiste en dar soporte a consultas, visualizaciones y navegación.

La función principal de los *metadatos* dentro de los bloques de Acceso y recuperación es proporcionar una trayectoria de navegación para consultas y herramientas de profundización al detalle. También se crean *metadatos* como resultado de aplicar consultas y crear depósitos temporales de datos. Las herramientas de acceso utilizan los *metadatos* que se crean para navegar por los datos.

La función principal de los *metadatos* dentro de los bloques de Análisis y reportes consiste en almacenar reportes y consultas predefinidos y apuntar hacia ellos. En tal caso, el usuario recupera con eficiencia reportes y consultas predefinidos mediante la navegación de los *metadatos* que los organizan

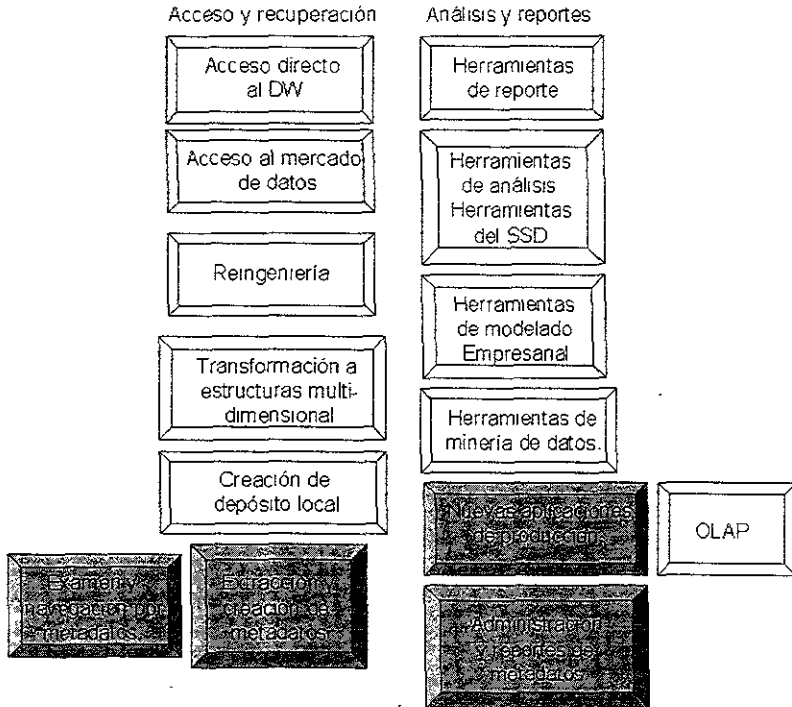


Figura 2.16 Función de los metadatos dentro del bloque de Acceso y Uso

2.2.7 Publicación y acceso a los datos

En su momento hemos mencionado varias veces ya que la base del *DW* está en dar respuesta a las preguntas de aspectos de negocios, es como alimento para el cerebro de los empleados. Nuestras respuestas vienen en forma de reportes, la culminación de todo el trabajo que se involucra en el proyecto *DW* se encuentra en ellos. Hasta los mejores diseños fracasarán si no se entregan las respuestas

Toda esta información se tiene que entregar, y la principal forma, la más conocida es a través de los reportes, en forma concisa, con claridad y oportunidad.

Exploremos las implicaciones de varios elementos de reportes que son críticos y veamos cómo conviene implementarlos en las aplicaciones.

Las herramientas con que se pueden hacer los reportes reciben una gama de nombres, con base en las diferentes arquitecturas y plataformas, el mercado se encuentra lleno con

una variedad de herramientas disponibles, las cuáles pueden ejecutarse en estaciones de trabajo, clientes o en servidores aplicativos, incluso, algunas publican la información en Internet.

Independientemente de sus nombres, arquitecturas o plataformas, todas estas herramientas tienen algo en común: sirven para obtener información de una base de datos y publicarla, llamémoslas "*herramientas de reportes*".

La mayoría de las herramientas de reportes presentan la información en una de tres formas típicas: Reportes Tradicionales, Tablas Pivote y Diagramas.

- **Los Reportes Tradicionales.** Nos son familiares a todos, típicamente contienen columnas de datos con encabezados y con uno o más niveles de subtotales. Este tipo de reportes se han mantenido casi sin cambios a lo largo de la historia de las empresas y corporaciones al igual que han sobrevivido la transición de la arquitectura de *mainframes* a la de *cliente-servidor*. Aunque ya no son fajos enormes de hojas, la experiencia dice que cualquier cuestión de negocios se puede contestar en un reporte tradicional.
- **Las tablas pivote.** Presentan las medidas en referencias cruzadas, su diferencia principal estriba en la forma en que se presentan las medidas resumidas (*summary measures*) con respecto a las dimensiones. Como los valores dimensionales se enumeran tanto en los encabezados de columnas como en los de renglones, su implementación técnica es muy diferente, pero sus conceptos básicos siguen siendo los mismos. Aunque éstas son los mecanismos favoritos de presentación de los proveedores de OLAP, y con frecuencia se les llama "el paso siguiente", permiten al usuario penetrar (*drill*) en niveles de detalle sucesivos; la experiencia puede defender con éxito los reportes tradicionales sobre este tipo de representación de datos, al añadir al *reporte atributos de referencia desde y hacia cualquier otro reporte*.
- **Los Diagramas.** Muestran la información en forma gráfica. La relación de las medidas y las dimensiones se muestra en forma gráfica (píes, barras, líneas, tridimensionales, etc.). cada diagrama se puede dibujar de muchas maneras distintas, lo cuál nos lleva a que un mismo conjunto de datos se pueda adecuar para influir de distintas formas en las personas.

Podemos decir que en realidad, el análisis de la información no depende de la forma en que se represente la información, sino en la información misma. Y todo esto nos sirve para fundamentar que aquí analicemos los principales tipos de necesidades de información en forma de reportes tradicionales

Básicamente, un reporte se compone de una serie de columnas tituladas con nombres de campos de una tabla de datos, y renglones que se van llenando con los valores de los registros asociados a dichos campos.

Nos referimos a estos reportes como simples porque toda la información requerida se puede obtener en una sola sentencia SQL sin cálculos o procesos posteriores.

Pero, si se pide una columna con porcentajes de la suma total, se puede realizar con una sola instrucción SQL que realice las búsquedas, y conforme obtiene los resultados va sacando la relación contra la sumatoria de la columna, o bien, se puede obtener resolver por métodos de programación, en este caso sencillo, la opción de hacer programación nos da ventaja al evitar hacer recálculos durante la búsqueda SQL.

Este tipo de reportes se pueden generar con cualquier herramienta de reportes del mercado, igualmente se pueden escribir muy sencillamente, sin embargo, en la práctica los reportes son bastante más complicados que eso.

Inicialmente, tenemos los subtotales, supongamos por ejemplo, en la figura 2.17 se hace un reporte en donde uno de los campos se tenga que subdividir por categorías o rangos de valor para obtener un poco más y mejor información, un poco más explícitamente, supongamos un reporte en donde los renglones se agrupan por (digamos) el nombre del proveedor, además, se quiere hacer una sumatoria parcial de las ventas por cada proveedor y de las relaciones de compras contra ganancias, por cada renglón y finalmente, se quiere tener una sumatoria total de ventas y de relaciones de ventas, por ejemplo:

Proveedor	Artículo	Compras	Ganancias	%ganancia
Legal	Sobre	10,000	1,000	10%
Legal	Frasco	5,000	800	16%
Legal	Costal	1,000	400	40%
	Subtotal:	16,000	2,200	13.75%
ORO	Sobre	10,000	1,000	10%
ORO	Frasco	5,000	800	16%
ORO	Costal	1,000	400	40%
	Subtotal:	16,000	2,200	13.75%
	Gran Total:	32,000	4,400	13.75%

Figura 2.17 Ejemplo de reporte.

En este caso, no basta una sentencia SQL sencilla para llenar el reporte pedido con sus distintos niveles de agrupación, la primera parte no representa el menor problema, con SQL se hace la suma de acuerdo al proveedor, sin embargo, los subtotales correspondientes a los porcentajes no se pueden calcular por medio de una suma, por la propiedad de asociación para la multiplicación. En este caso, el problema no es mayor, aunque no se puede hacer todo en una sola sentencia SQL, debido a que los porcentajes son componentes no aditivos; en el reporte se hacen las sumas de compras y ganancias, de tal suerte que ya sólo hay que dividirlos entre sí, esto es más sencillo por programación que por SQL.

Hay un problema con los promedios, y es que no son aditivos, y además no se puede sacar un promedio general haciendo promedio de los promedios, cuando nos

encontramos con ellos, lo que podemos hacer es un nuevo llamado a los datos base que generaron los promedios, hacer un nuevo cálculo del total de elementos y realizar el promedio por medio de sumatorias de los datos completos involucrados, lo que hablando de un *DW* se traduce en impráctico

Por fortuna existe una alternativa, ejecutar más de una sentencia SQL, primero para obtener todos los datos y realizar sumatorias, luego, previo almacenamiento del resultado, se hace una nueva búsqueda sacando las relaciones de los subtotales, y se intercala la información en los lugares adecuados.

Una objeción válida sería que por medio de programación se puede obtener toda esta información y resultados en una sola búsqueda, ¿para qué entonces realizar tres? En respuesta, consideremos una base de datos muy grande, en donde al realizar las búsquedas vamos simplificando las tablas en cada nivel subsecuente, de modo que los procesos se aceleran bastante.

¿Qué conviene más? Depende del caso específico que estemos tratando.

Otra situación en donde falla el SQL es cuando necesitamos hacer comparaciones entre los renglones de las tablas del reporte, en el ejemplo que pusimos, un caso sencillo sería: "diferencia de las ganancias de ORO en saco" contra "Legai en sobre".

Estos casos se presentan también cuando queremos hacer clasificaciones de las columnas o hacer totales corrientes. En estos casos, lo mejor es evitar todo tipo de acrobacias de SQL y dejar la tarea a la programación de la aplicación.

Las clasificaciones en orden ascendente o descendente son frecuentes en las preguntas de decisiones de negocios, algo común es que se pida un reporte de los porcentajes de ganancia en los productos, ordenados por proveedor y luego cada proveedor ordenado en forma descendente por los porcentajes de ganancia, y por si fuera poco, también quiere (porque los hay) que en el reporte aparezcan ordenados por proveedor del que da menor margen de ganancia al mayor.

Entonces, lo que tenemos que hacer es unir lo ya mencionado con la creación de tablas temporales, de tal modo que primero, por múltiples búsquedas, generamos el reporte, y lo almacenamos en una tabla temporal, y luego, por medio de una nueva sentencia hacemos un ordenamiento por valores de proveedor y porcentaje de ganancia.

Esto mismo ocurre cuando se trata de obtener totales corrientes, los cuales deben ser calculados primero, y no pueden aparecer en el reporte sino hasta que se han obtenido las dimensiones y las medidas aditivas de las bases y los subtotales se han calculado.

Posteriormente, tenemos que enfrentarnos con algunos casos en donde la información no se encuentre en una sola tabla, sino en dos o más diferentes, e incluso, en distintas bases de datos; en estos casos, no es posible ejecutar la búsqueda en una sola sentencia SQL, primero porque habrá datos que no queremos perder, pero que hay que separar, luego, hay que realizar uniones externas "*outer joins*" y posiblemente, en bases de datos heterogéneas dado que es un *DW* (muchos manejadores de BD no soportan hacer varias

uniones externas, además, los optimizadores requerirían demasiado tiempo para resolver esta situación).

Muchas veces hay que restringir las búsquedas, obteniendo las cotas de otras tablas o de filtros, de sumatorias, o de clasificaciones ascendentes o descendentes, esto provoca que en las sentencias SQL la cláusula "donde" (*having*) se vea *enriquecida* por las complicaciones antes mencionadas para cada una de estas situaciones, por lo tanto, lo que más nos conviene es hacer lo mismo que con las comparaciones de columnas: Ejecutar cada sentencia SQL por separado y luego ensamblar los resultados para generar el reporte.

Finalmente, esto nos lleva a un aspecto interesante y delicado del *DW*, los grupos y las operaciones de conjuntos: Siguiendo la temática de las tablas temporales, podemos descubrir que algunas de estas tablas generadas para cuestiones temporales se pueden repetir con cierta frecuencia, de tal suerte que almacenarlas nos generaría una mejoría en rendimiento que sobradamente justificaría el almacenar dicha tabla, y al mismo tiempo, dichas tablas pueden definir grupos o conjuntos de datos, sobre los cuáles se pueden aplicar operaciones de conjuntos, aumentando notablemente la funcionalidad al igual que el rendimiento.

Pero con cuidado, pues en un *DW* se pueden generar una gran cantidad de estas tablas, de modo que podrían consumir una gran cantidad de recursos del sistema, y por otro lado, si tenemos una población de datos que en alguna manera tenga una migración aunque sea muy lenta, estaremos generando datos erróneos.

Así, tenemos que para cada caso de pregunta de una persona de negocios habrá un tipo especial de reporte, y para cada uno de ellos habrá una o varias formas de implementación, y dependerá del programador o de las herramientas de reporte de datos la alternativa que se elija.

Así, tenemos que la elección de una herramienta de reporte de datos es un aspecto crucial en el desarrollo del *DW*, tanto como puede ser el encargado del sistema en los sistemas OLAP actuales.

En lo que se refiere a estas herramientas de reporte, las podemos clasificar por el soporte proporcionado al usuario, como se ve en la figura 2.18.

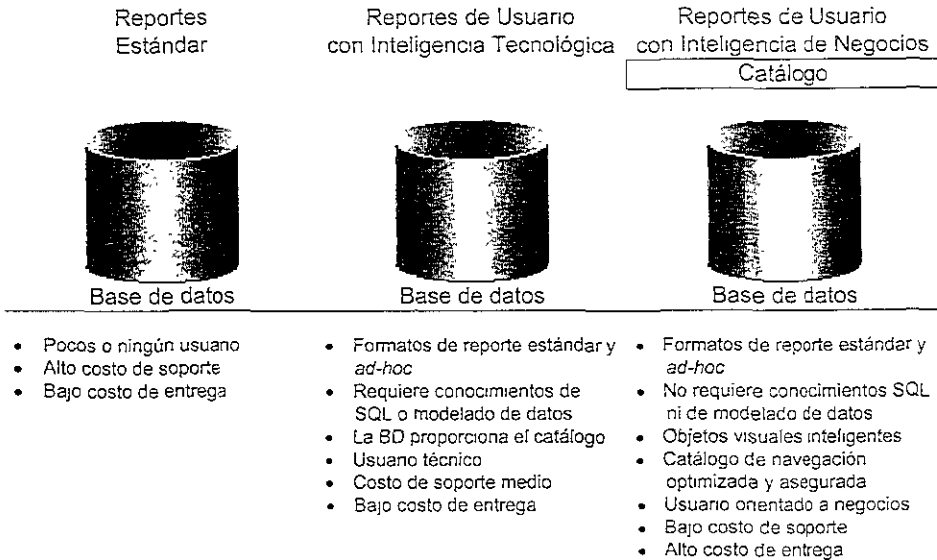


Figura 2.18 Comparación de herramientas por el tipo de soporte al usuario.

Esta figura, indica que mientras más evolucionada sea nuestra herramienta de reporte de información, será mejor la productividad de nuestro sistema *DW*, pues los costos de soporte que siempre están presentes y son altos, se reducen bastante, la información se entrega más rápido y además, el usuario obtiene justo la información que desea.

Cada una de estas opciones tiene sus beneficios en cuanto a economía, y sus propios aspectos a considerar con relación a la arquitectura que le ha de soportar o albergar.

Además, también es necesario considerar el tipo de usuarios que tenemos en nuestro *DW*, pues éstos pueden variar desde un alto ejecutivo que requiera indicaciones enormes que le lleven de un solo click a sus deseos de información, pasando por usuarios capaces de usar una PC, y finalmente pueden ser usuarios informáticos, que desean tener total control de sus "paseos" por la información, una guía corta consiste en tener presentes los siguientes puntos:

- **Patrones de conducta del usuario**
 - ¿Sabe y puede usar una PC, o las evita a toda costa?
 - ¿Necesitan accesos continuos a la información, o accesan mensual, trimestral o anualmente?
 - ¿Tiene el usuario quien haga las consultas por él, o prefiere tener la libertad de acceder la información y manipularla a su antojo?

- **Contenido y granularidad**

- ¿Necesita resúmenes de datos, o necesita profundizar mucho en detalles?
- ¿Quiere papeles para hacer un reporte, quieren reportes hechos de acuerdo a sus propios estándares, o prefieren un tipo de reporte estándar del que puedan obtener los datos y encontrar información?
- ¿Que tan profundo quieren entrar en su información, van de abajo a arriba, o de arriba abajo respecto a los niveles dimensionales de la información?

- **Equipamiento**

- ¿Tiene PC?, ¿utiliza Windows, Macintosh, Linux? ¿Sólo tiene acceso por terminal de datos?
- ¿Se puede determinar que herramienta debe usar, o ya está definida por el propio usuario?
- ¿Cuál es la configuración típica de PC para los usuarios?
- ¿Qué ejecutan regularmente en sus PCs, tienen abierto y activo un paquete de oficina, qué tantos recursos se consumen estos programas, cuántos dejan libres para las aplicaciones *DW*?

No debemos descuidar que por más completo que sea un diseño, y por más poderosas que sean las herramientas OLAP del sistema, siempre hay imprevistos que sólo se pueden resolver manipulando directamente la información.

Invariablemente, el coordinador de un *DW* se enfrenta a un inmenso trabajo para finalizar el *DW*. Este sujeto tiene que dar acceso y reportes personalizados para una gran comunidad de usuarios muy diversos con muy distintas necesidades de reportes de datos. Estas diferencias incluyen el volumen de la información, la diversidad y las estrategias de acceso asociadas a ambos factores. Los retos de dar la facilidad y los recursos a los usuarios en tiempo y forma para solventar sus necesidades de información son muchos.

La clave para finalizar exitosamente es una buena arquitectura que dé flexibilidad y reduzca los requerimientos para los reportes *ad-hoc* y los reportes estándar. Si se puede construir un *DW* fácil de entender y se les dan a los usuarios las herramientas adecuadas para acceder a los datos, se logrará un alto grado de satisfacción en el resultado.

En el siguiente capítulo se analizarán una gama de herramientas para el análisis, desarrollo e implementación de un *DW*, así como su distribución por parte de los principales fabricantes y proveedores de servicios.

Capítulo 3

Herramientas de Data Warehouse

En este capítulo se describen las herramientas existentes para lograr un buen Data Warehouse, como son herramientas de apoyo, de extracción, validación, OLAP y Minería de datos.

3.1 Herramientas de Apoyo a la construcción Data Warehouse

Una vez que se toma la decisión de intentar cosechar los beneficios de un *DW*, el departamento de informática se enfrenta al reto de la implementación. La experiencia de ser implementadores pioneros del *DW* se resume en las siguientes premisas:

- No es posible comprar un *DW*, hay que construirlo.
- El *DW* es un proceso no un lugar.

Como a estas alturas no hay un *DW* que se venda comercialmente, el área de la informática debe construirlo a partir de los componentes disponibles. La meta de las organizaciones más prudentes de tecnología de la información consiste en minimizar la construcción y maximizar la compra.

La figura 3.1 muestra que después de analizar las necesidades empresariales y construir una arquitectura como resultado, el área informática necesita examinar las estructuras de los fabricantes y los productos que existen dentro del mercado para comprar.

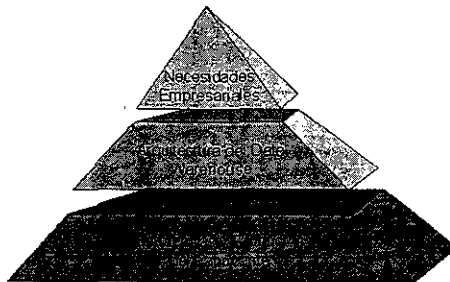


Figura 3.1. Estructuras y productos de fabricantes

El enfoque de implementación de datos del DW

Un DW se implementa a través de tres grandes enfoques, "Construya el suyo propio DW", "La Estructura del fabricante" y "El Fabricante Ancla" (ver figura 3.2). La elección de un enfoque en particular se ve afectada por muchos factores.

- Inversiones existentes en infraestructura de comunicaciones, por ejemplo, técnicas para la tecnología de la información, redes de comunicaciones y administración de redes y sistemas.
- Capacidad y rendimiento de la organización en la tecnología de la información.
- Cómputo corporativo y, políticas y estándares de tecnología.
- Inversiones existentes, elección de plataformas de cómputo y tecnología de base de datos
- Nivel de satisfacción con los proveedores actuales.
- Cultura corporativa.

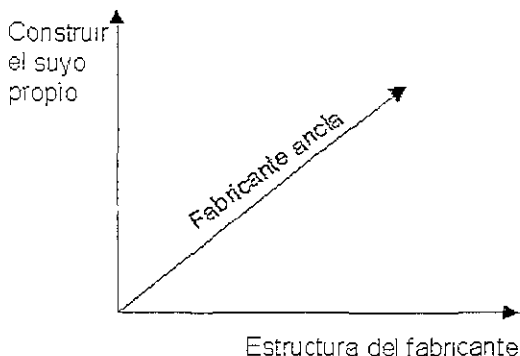


Figura 3.2. Enfoque de elección de fabricantes.

Construya el suyo propio. En este enfoque, la organización en tecnología de la información construye el DW utilizando una arquitectura personalizada. Se seleccionan los productos de fabricantes y el equipo en tecnología de la información es el responsable de la actividad de integración de sistemas. Se aplica la arquitectura de referencia para definir la arquitectura técnica y la funcionalidad de los productos requeridos. La estrategia consiste en seleccionar los mejores productos de sus clases para satisfacer los requerimientos empresariales y las restricciones de presupuesto. Este enfoque es similar a construir una casa a la medida. La elección de los productos y la integración de los sistemas son obstáculos formidables.

Estructura del Fabricante. Los principales fabricantes de bases de datos y plataformas de cómputo ofrecen estructuras de DW para influir y guiar el mercado de los DW. En la mayoría de las estructuras son similares en ámbito y sustancia, con diferencias provocadas en gran medida por la tecnología principal del fabricante. La elección de una estructura en particular limita las acciones para ciertas funciones y ofrece, en potencia,

opciones de productos de lo mejor en su clase para otras funciones del *DW*. Muchas estructuras pretenden ofrecer un gran nivel de integración y por lo tanto, como consecuencia, un desempeño mejorado y la administración del sistema.

Fabricante ancla. (Distribuidor de productos o servicios) En este enfoque de implementación, la empresa selecciona un proveedor existente, como su fabricante clave o ancla. Después, se usan los bienes o servicios del proveedor ancla para influir en la selección de productos y servicios que satisfagan el resto de las funciones que se requieren para construir el *DW* deseado.

Por lo general, los criterios importantes para utilizar éste enfoque son el ámbito y el tamaño de las inversiones existentes en técnicas personales y el uso actual de los productos del fabricante ancla. Para fines de análisis, los fabricantes ancla se dividen en cuatro grupos:

- **Fabricantes de hardware y plataformas de sistemas operativos:** IBM, DEC, HP, Microsoft, SUN, Pyramid y Sequent.
- **Fabricantes de sistemas administradores de bases de datos: (DBMS).** Oracle, Sybase, Informix, Computer Associates, Software AG, IBM, y Red Brick.
- **Fabricantes de herramientas de soporte de decisiones.** SAS Institute, Sterling Software e Information Builder.
- **Proveedores de servicios de integración de sistemas.** Andersen Consulting, Price Waterhouse, Coopers Lybrand, Ernest & Young, Computer Science Corporation, ISSC y EDS.

La figura 3.3 muestra la influencia de los proveedores de soluciones apoyadas cuando se selecciona la estructura y el fabricante ancla. Además muestra en dónde se han realizado las inversiones históricas y dónde es probable que se mantengan. Los autores comprenden que forzosamente habrá algunas discrepancias, además de que el análisis no es definitivo. La sugerencia es sentarse con los fabricantes y reunir la información necesaria para hacer la elección correcta. La figura indica los siguientes enfoques potenciales de los fabricantes y el perfil de continuidad en la inversión.

Dentro de la arquitectura de referencia del *DW* se encuentran los proveedores de servicios, de soluciones o de múltiples componentes y los proveedores de algunos componentes.

La mayoría de los proveedores de servicios de integración de sistemas pertenecen a una sociedad o alianza con uno o más proveedores de estructuras.

Componente / Fabricante	Plataforma de cómputo y/o sistema operativo	DBMS	Herramienta de desarrollo	Herramientas de análisis y soporte	Administración y manejo de sistemas	Middleware	Manejo de metadatos
IBM	♦	♦			♦	♦	
Hewlett Packard	♦				♦		
Oracle		♦	♦	♦			
Sybase		♦	♦			♦	
Informix		♦	♦				
Computer Associates		♦	♦		♦		
AT&T GIS	♦						
Platinum technology				♦	♦		♦
SAS				♦			
Pyramid	♦						
Sequent	♦						
Information Builders				♦		♦	
SUN Microsystems	♦				♦		
Microsoft	♦	♦	♦	♦			

Figura 3.3 Análisis de inversión en tecnología, para los proveedores de soluciones y de estructuras.

La arquitectura de referencia del DW consta de cuatro bloques y cuatro capas. Los proveedores de soluciones ofrecen productos para bloques o capas. En la figura 3.4, vemos una lista de los bloques y fabricantes, y capas de la arquitectura de referencia con los que se relacionan.

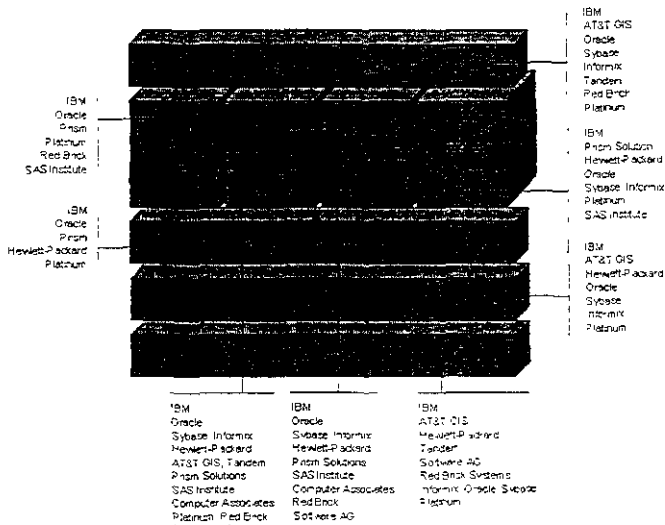


Figura 3.4 Proveedores de soluciones

En la figura anterior, donde se muestra la arquitectura de referencia del DW, mostrando cada uno de los bloques, vemos como existen fabricantes de herramientas que pueden cubrir uno o más de estos bloques. Es aquí donde un fabricante o proveedor de soluciones se relaciona con un bloque o capa en particular, no implica que lo esté con cada sub-bloque o sub-capa. Se requiere de un análisis más detallado del producto del fabricante, para comprender cuál es la funcionalidad exacta que ofrece.

Los proveedores de algunos componentes se relacionan con uno o dos de los bloques o capas de la arquitectura de referencia. En la figura 3.5 se muestra una lista de los fabricantes y los bloques y capas de la arquitectura de referencia con los que se relacionan.

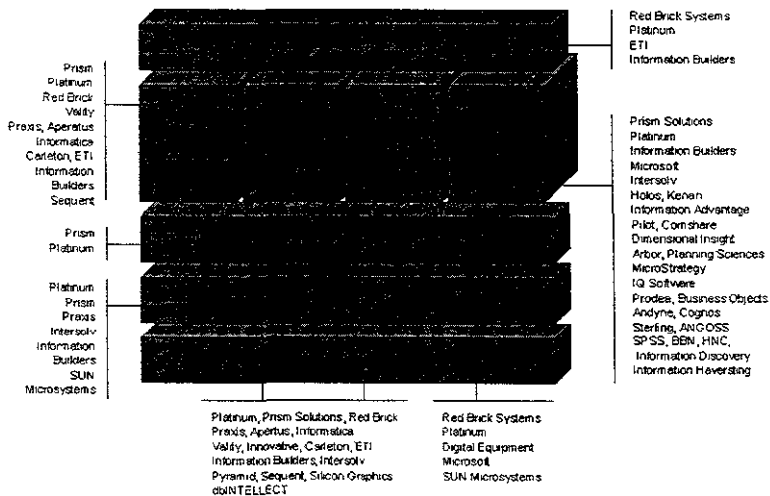


Figura 3.5 Proveedores de componentes

Aquí vemos claramente que un proveedor de componentes no está casado con un bloque de la arquitectura de referencia, sino que puede tener participación en más de un bloque. A diferencia de los proveedores de soluciones donde se encuentran los *Gigantes*, estos proveedores de componentes en algunos casos sólo participan en una capa de la arquitectura, aunque esto no quiere decir que no puedan ser también proveedores de soluciones.

3.2 Herramientas OLAP

En un DW, el usuario requiere resultados, información a la mano y con prontitud, para ello el DW se vale, entre otras herramientas, del procesamiento analítico en línea (OLAP). A diferencia del procesamiento de transacciones en línea (OLTP), en donde los datos se

reúnen y almacenan para operación y control, OLAP es una tecnología de procesamiento analítica que crea nueva información empresarial a partir de los datos existentes, por medio de un rico conjunto de transformaciones empresariales y cálculos numéricos.

OLAP es una opción de análisis y de reporte, es un componente del bloque de acceso y uso de la arquitectura del DW (figura 3.6).

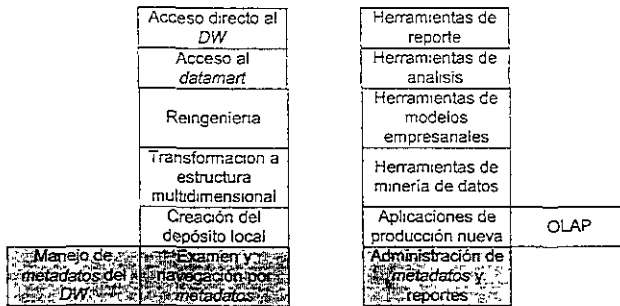


Figura 3.6 Bloque de acceso y uso de arquitectura DW.

La arquitectura OLAP, como se ve en la figura 3.7, consta de dos partes:

- **Visión OLAP.** La presentación multidimensional y lógica de los datos del DW o del *Datamart* al usuario sin importar cómo y dónde están almacenados los datos.
- **Tecnología de depósito de datos.** Las dos opciones populares son el depósito de datos multidimensional y el depósito de datos relacional.

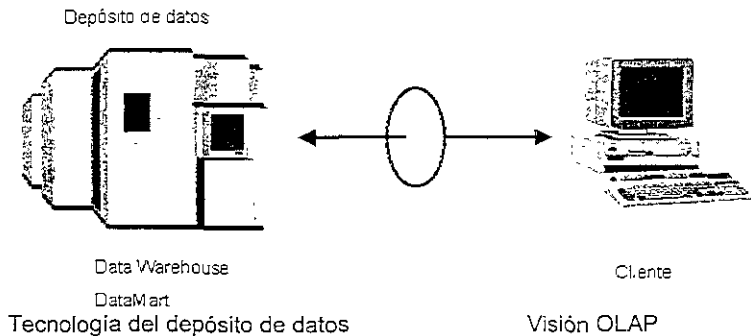


Figura 3.7 Arquitectura OLAP

Hay dos tendencias para la implementación del depósito de datos de esta arquitectura: El primer caso, el depósito de datos en el cliente, da por resultado un cliente "gordo" que

complica la carga de datos en la estación de trabajo, pero el cuello de botella es sólo un aspecto, los impactos negativos potenciales son en el desempeño y en la seguridad de los datos.

En el segundo caso hay un mercado de datos, en este esquema de la arquitectura, los datos se extraen del DW y se transforman luego en estructuras de datos multidimensionales, las cuales se almacenan en el servidor del mercado de datos.

En cualquier caso, únicamente hay dos opciones para almacenar los datos: el depósito multidimensional y el depósito relacional. En la figura 3.8 podemos ver aspectos de ambas.

	Base de datos relacional	Base de datos multidimensional
Depósito de datos, acceso y visión	<ul style="list-style-type: none"> • Relacional • Tablas de columnas e hileras • Lenguajes SQL con ampliaciones • Herramientas de terceros con soporte API 	<ul style="list-style-type: none"> • Dimensional • Arreglos <i>Hipercube</i>, Multicubo • Recopilación de matriz dispersa • Propietario de diseño de cálculo
Utilización e incorporación	<ul style="list-style-type: none"> • OLTP • Motor RDBMS • Profundización a nivel de detalle • Desempeño de consultas: rango amplio 	<ul style="list-style-type: none"> • OLAP • Motor multidimensional • Profundización a nivel de resumen/adición • Desempeño de consultas: rápido
Tamaño y actualización de base de datos	<ul style="list-style-type: none"> • Gigabytes a terabytes • El depósito de índices y el retiro de normas incrementan el tamaño • Consulta y cara paralelas • Actualización durante el uso 	<ul style="list-style-type: none"> • Gigabytes • Compresión y adición de datos dispersos • Difícil actualización durante el uso; los cambios pequeños pudieran requerir reorganización

Figura 3.8 Depósito de datos relacional y multidimensional

En el depósito de datos relacional las bases de datos relacionales almacenan los datos en tablas como registros con llaves, y el acceso a los datos mediante un lenguaje común, el SQL. Por otra parte, los depósitos de datos multidimensionales almacenan los datos de manera lógica en arreglos, con la desventaja de que no hay un modelo multidimensional común o acordado y no hay un método común o estándar para el acceso de datos.

Con un depósito de datos relacional, el DW o el *Datamart* pueden tener una dimensión muy grande. El tamaño del depósito aumenta por el uso de índices y técnicas de retiro de normas para lograr un desempeño aceptable en consultas multidimensionales.

Al decidir cómo y dónde proporcionar los servicios OLAP, se debe preguntar si es posible combinar el depósito de datos multidimensional y los servicios OLAP, o si los depósitos de datos relacionales con servidores OLAP satisfacen las necesidades multidimensionales del usuario. Cada enfoque tiene sus méritos y sus retos, sin ninguna respuesta sencilla.

Críterios de selección

Los requerimientos de bases de datos relacionales se han visto frecuentemente descritos por las 12 reglas desarrolladas por E.F.Codd, quien en 1993, junto con S.B. Codd y C.T. Salley, concibieron 12 reglas para la evaluación de herramientas de procesamiento

analítico, las reglas fueron originalmente el resultado de un estudio de cliente, por lo tanto, hay cierta duda sobre la validez de las siguientes reglas:

1. **Visión conceptual multidimensional.** Los modelos OLAP deben ser de naturaleza multidimensional, los usuarios pueden realizar operaciones de rebanar, pivotar y girar trayectorias de consolidación (resúmenes y adiciones "sumarios") dentro de un modelo.
2. **Transparencia.** OLAP debe permitir que la herramienta analítica se incorpore en cualquier parte que el usuario desee, sin efecto adverso sobre la funcionalidad de la herramienta anfitrión.
3. **Facilidad de acceso.** OLAP debe ubicar su propio esquema lógico para depósitos físicos de datos heterogéneos, y el tipo de sistema de donde provienen debe ser transparente al usuario.
4. **Desempeño consistente de reportes.** El desempeño de los reportes no debe degradarse al aumentar la cantidad de las dimensiones.
5. **Arquitectura cliente/servidor.** La incorporación de clientes debe ser con el menor esfuerzo de integración, y el servidor debe ser capaz de ubicar y consolidar datos entre bases de datos físicas y lógicas dispares para mantener la transparencia y construir un esquema conceptual común, lógico y físico.
6. **Dimensionalidad genérica.** Debe existir sólo una estructura lógica para todas las dimensiones. Debe ser posible usar cualquier función aplicable sobre cualquier dimensión en cualquier otra.
7. **Manejo de matriz de dispersión dinámica.** OLAP debe ser capaz de deducir la distribución de los datos y cómo almacenarlos de manera más eficiente, de modificar en forma dinámica los métodos de acceso y deben proporcionarse diferentes tipos de mecanismos, como cálculo directo, árboles B, y dispersión, o la mejor combinación
8. **Soporte multiusuario.** Las herramientas OLAP deben suministrar acceso concurrente, integridad y seguridad para dar soporte a usuarios que necesitan trabajar en forma concurrente.
9. **Operaciones cruzadas entre dimensiones sin restricción.** OLAP debe manejar cálculos entre dimensiones sin requerir que el usuario defina cuáles deben ser los cálculos.
10. **Manipulación intuitiva de datos.** Las dimensiones definidas en el modelo analítico deben contener toda la información que necesita el usuario para efectuar cualquier acción inherente.
11. **Reportes flexibles.** Las opciones de reporte deben proporcionar la capacidad de manipular, analizar, sintetizar y observar los datos en cualquier forma

deseada, incluyendo la creación de agrupamientos lógicos o la colocación de hileras, columnas y celdas unas junto a otras.

12. **Dimensiones y niveles de adición ilimitadas.** Cada dimensión genérica debe permitir una cantidad ilimitada de adiciones definidas por el usuario y niveles de resumen dentro de una trayectoria de consolidación determinada.

En general, podemos decir que las herramientas OLAP tienen que reunir cinco criterios: Características y funciones, Acceso a las características y funciones, Motor de servicios OLAP, Administración y visión global de la arquitectura.

Las Características y funciones que un sistema OLAP necesita para crear y presentar la información a través de las fórmulas y reglas definidas son:

- Manejar múltiples dimensiones y jerarquías dentro de una dimensión.
- Agregar, resumir, precalcular y derivar datos a lo largo de una sola dimensión o de un conjunto de dimensiones seleccionadas.
- Aplicar lógica de cálculo, fórmulas y rutinas analíticas sobre una dimensión o un conjunto seleccionado de éstas.
- Manejar el concepto de modelo analítico, un conjunto de dimensiones seleccionadas y sus elementos, la lógica de cálculo, las fórmulas y rutinas analíticas, y datos agregados, resumidos y precalculados.
- Ofrecer una rica biblioteca de funciones (financieras, comercialización, logísticas, algebraicas, estadísticas).
- Proporcionar una poderosa capacidad de cálculos y análisis comparativos, tales como clasificadores, comparaciones, porcentajes de clase, máximos, mínimos, promedios, etc.
- Realizar cálculos cruzados entre dimensiones, tales como asignaciones de costos y eliminaciones dentro de la compañía, o cálculos a nivel hilera para aplicaciones orientadas a hojas de cálculo, etc.
- Ofrecer inteligencia de tiempo, como año calendario, expansión del calendario de un período de tiempo dado, períodos actuales, calendarios, etc.
- Transformar una dimensión en otra (por ejemplo, después de una fusión).
- Navegación y análisis mediante pivoteo, tabulación cruzada, profundización y generalización de resúmenes a lo largo de una o varias dimensiones.

El desempeño debe satisfacer las necesidades analíticas de los usuarios empresariales, de modo que el proceso de análisis sea fluido y sin interrupciones.

El acceso a características y funciones se debe reflejar en la interfaz del usuario empresarial pues el acceso a los servicios OLAP se debe ofrecer con diversas opciones que confieran mayor poder a las habilidades que ya posee el usuario empresarial y al conocimiento incorporado en los modelos analíticos OLAP. Entre las opciones están:

- Las hojas de cálculo: Ya que por lo menos, los usuarios deben poder cargar los datos OLAP en sus herramientas de hojas de cálculo para análisis y reportes adicionales.

- Nos puede resultar muy útil la oferta de algunas herramientas cliente del vendedor, por ejemplo una aplicación específica para empezar pronto el procesamiento analítico. El criterio fundamental es la riqueza en funcionalidad y el ajuste a las necesidades empresariales.
- Las herramientas de otros fabricantes pueden aprovechar bien las APIs del servidor OLAP (En caso de que las API sean del proveedor, hay la posibilidad de que encajen en el servidor OLAP).
- El ambiente 4GL debe manejar todas las funciones y características del servidor OLAP.
- Las interfaces con el "estándar de facto", o ambientes de aplicación como Visual Basic y Power Builder, e interfaces como OLE, DDE y CORBA.
- Los "Navegadores de cubo" para clientes, que son herramientas de otros fabricantes que tienen una interfaz con los servicios OLAP.

Para aprovechar el conocimiento incorporado en el modelo analítico, la interfaz de acceso debe de:

- Acceder y extraer subconjuntos de datos con base en jerarquías, modelos, tiempo y otras dimensiones seleccionadas.
- Acceder múltiples niveles de jerarquía con una sola solicitud de extracción
- Estar "atento" de los datos agregados y resumidos, los segmentos y los índices, para crear la consulta adecuada.
- Estar optimizada para la base de datos relacional particular, incluyendo sus extensiones SQL, cuando se acceda al depósito de datos relacional.

El motor de servicios OLAP, en cualquiera de sus configuraciones, la de depósito dimensional o la de depósito relacional, debe satisfacer la capacidad, la viabilidad de cambio de escala y las características tecnológicas del modelo analítico y la aplicación planeados. Las características de tecnología requeridas dependen del modelo analítico y del uso contemplado. Algunas de estas características son:

- **Capacidad de lectura-escritura.** Ésta es para aplicaciones interactivas de pronósticos y presupuestación.
- **Escritura multiusuario.** Ésta es para manejar el análisis multidimensional de grupos de trabajo. La escritura multiusuario OLTP representa un mayor reto que un acceso directo de escritura de datos relacionales. En vez de sólo referirse a una hilera o tabla, una solicitud de actualización o escritura OLTP puede requerir que se vuelvan a calcular valores derivados y calculados que afectan muchas dimensiones y jerarquías dentro de las dimensiones. El ámbito del bloqueo de escritura puede ser muy amplio y el recálculo puede usar muchos recursos de cómputo, dando por resultado bloqueos prolongados y que el desempeño parezca bajo.
- **Bases de datos múltiples.** Si hay una base de datos para cada aplicación OLAP, pudiera requerirse un mecanismo para la interacción entre las bases de datos, debido a que los valores derivados de una pudieran ser las entradas para otra.

- **Rango de tipos de datos.** Estos son numéricos, períodos de tiempo, de calendario, descripciones (para exhibición y reportes), etc. Otros tipos de datos de imagen mejoran las comunicaciones del análisis complejo, con exhibiciones atractivas y reportes ejecutivos.

Las funciones necesarias de la administración para fines de preparación inicial, configuración y operación continua incluyen:

- Definición del Modelo analítico dimensional.
- Creación y mantenimiento del depósito de *metadatos*.
- Control de acceso y privilegios con base en el uso. Tenemos que concentrarnos en saber qué desean hacer los usuarios empresariales y quién puede obtener acceso al modelo analítico y sus datos.
- Carga del modelo analítico desde el *DW* o el *Datamart*.
- Afinación del desempeño a niveles aceptables para permitir un análisis que no desorganice.
- Reorganización de la base de datos para mejorar el desempeño, cambiar el modelo dimensional o actualizar los datos.
- Administración de todas las partes del sistema, incluyendo el *middleware*, ya que la arquitectura proporciona un modo ordenado de entender el ámbito de la tarea de administración de sistemas.
- Distribución de los datos al cliente para análisis adicional y local.

La arquitectura global tiene que definirse sobre la base de la perspectiva del depósito de datos para OLAP, ya sea multidimensional o relacional, lo cuál no es una decisión simple. La empresa necesita proporcionar los criterios para hacer la elección adecuada.

Por fortuna, la tendencia en la industria parece tratar de hacer una combinación de un proceso frontal de servidor OLAP con depósito multidimensional y un proceso posterior de depósito relacional con un nivel fino de detalle de datos (figura 3.9). Dicha combinación solventa las debilidades relativas al mismo tiempo que aprovecha sus puntos fuertes. Algunas empresas comienzan con un depósito relacional e incorporan depósitos multidimensionales cuando se hace necesario.

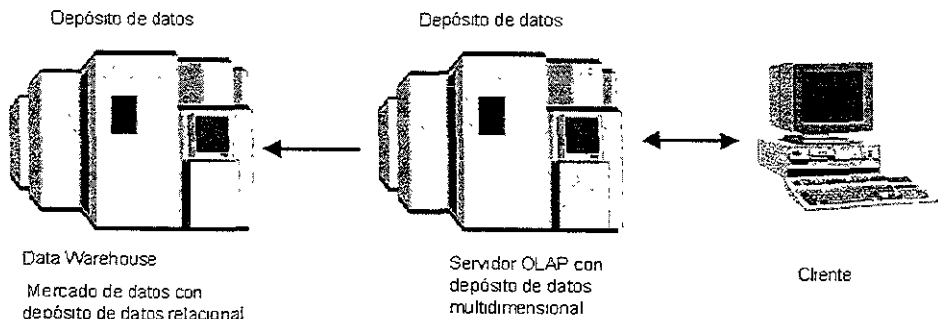


Figura 3.9 Tendencia de arquitectura OLAP.

3.3 Herramientas para Minería de Datos

La minería de datos es un arma esencial en el arsenal del soporte de decisiones del analista.

La minería de datos auxilia a los usuarios empresariales en el procesamiento de vastas reservas de datos para descubrir "relaciones insospechadas", por ejemplo, entre productos y clientes o patrones de compra de los clientes. La meta es descubrir "revelaciones estratégicas competitivas" para controlar la participación en el mercado y las utilidades.

Las herramientas de minería de datos utilizan el sub-bloque de Análisis y recuperación para tener una interfaz con el *DW* y con el mercado de datos. Muchas de las herramientas de la minería de datos también emplean el componente de depósito local del bloque de Acceso y recuperación, a fin de almacenar los datos en estructuras de datos de propietarios para análisis subsecuentes y presentaciones de resultados.

La mayoría de las herramientas de minería de datos pueden con facilidad saltarse el *DW* o el mercado de los datos y acceder de manera directa la fuente de los datos. Tradicionalmente, las herramientas de minería de datos acceden los datos de la fuente. Sin embargo, los datos del *DW* o del mercado de los datos están refinados, integrados y estandarizados. La estandarización eliminó aspectos como las convenciones de nombres múltiples, las estructuras ocultas de codificación y los campos faltantes. Los datos de los sistemas operacionales fuente son por lo general inconsistentes y están dispersos en varias aplicaciones. Además, se requieren datos históricos para descubrir patrones temporales de interés.

Tecnologías y herramientas de la minería de datos

Existe una amplia variedad de tecnologías para la minería de datos, y todavía van a aparecer más en el mercado. Estas herramientas y tecnologías de minería de datos se clasifican en tres grandes categorías:

- Análisis estadístico o de datos.
- Descubrimiento de conocimientos
- Otros, como sistemas de visualización, sistemas de información geográfica, análisis fractal y herramientas de propietario.

Análisis estadístico

Los sistemas de análisis estadístico (también conocido como análisis de datos), se usan para detectar patrones no usuales de datos. Algunas de las técnicas de modelado estadístico y matemático que se emplean son el análisis lineal y no lineal, el análisis de regresión continua y logística, el análisis de univariación y multivariación, y el análisis de series históricas.

Las herramientas de análisis estadístico se utilizan en diversas aplicaciones empresariales: incrementar la participación en el mercado y las utilidades, detectar las mejores oportunidades, aumentar la satisfacción del usuario por medio del mejoramiento de la calidad en productos y servicios (programas de la administración total de la calidad), e impulsar los márgenes a través de la modernización de la manufactura de productos y de la logística (ingeniería de proceso empresarial). Las herramientas de análisis estadístico existen desde hace tiempo y son las herramientas más desarrolladas que tienen para minería de datos. Han servido para reducir el tiempo de análisis, lo cual libera los recursos limitados para otras actividades de análisis, lo que a su vez conduce a una mejor toma de decisiones.

Características de las herramientas de análisis estadístico

Dada la complejidad de muchas de las tareas del análisis estadístico, las herramientas para el efecto deben ofrecer lo siguiente:

- **Funciones de visualización.** Estas funciones ayudan a descubrir relaciones entre grandes cantidades de datos. Por ejemplo, las funciones deben reconocer patrones en las series históricas de datos y exhibir gráficas de línea o logarítmicas, o realizar ajustes de curva para encontrar en los datos la "regla o patrón empresarial", o manipular los datos mediante el agrupamiento automático de valores de variables únicas seleccionadas, o alterando el punto de inicio y el tamaño de los histogramas.
- **Funciones exploratorias.** Estas funciones ayudan a elegir la función estadística y el modelo correctos que se ajustan a los datos. Algunas de estas funciones son tablas multidimensionales de pivoteo, ayuda orientada al análisis e identificación de valores extremos y distantes. La herramienta debe producir y presentar cuadros, gráficas y tablas para el analista empresarial, en forma dinámica y automática, como parte del proceso de exploración.
- **Funciones y operaciones estadísticas.** Estas funciones y operaciones ofrecen un rico conjunto de herramientas, tales como el análisis de regresión, tanto continuo como logístico; el análisis de series históricas, incluyendo autocorrelación; transformaciones rápidas de Fourier y pronósticos; análisis de variación múltiple; pruebas no paramétricas; y análisis de respuesta múltiple.
- **Funciones de administración de datos.** Estas funciones profundizan al detalle, examinan subconjuntos de datos, discriminan valores extremos, comparan subconjuntos, etc.
- **Funciones de grabación y reproducción.** Estas funciones graban los pasos del análisis, transfieren los registros a otro analista empresarial, y reproducen luego la tarea completa de análisis. Las funciones de grabación deben incluir los pasos del análisis, el proceso de selección de conjuntos de datos, una paleta o carrusel de cuadros y gráficas seleccionadas y cualquier otra

información que se vaya a comunicar. Esta es la clave para comunicar y compartir, tanto los resultados de la tarea de análisis estadístico, como las técnicas analíticas y el proceso aplicados.

- **Herramientas de presentación.** Estas herramientas comunican los datos complejos y el análisis de cuadros, gráficas y tablas sencillas. La herramienta debe convertir con rapidez los datos de un tipo de cuadro y, cuando se necesite, exhibirlos en un tipo de cuadro diferente. La herramienta debe también mostrar la variedad de tipos de cuadros, gráficas y tablas a los que se ajustan los datos, de modo que se seleccionen con facilidad las mejores opciones de presentación.
- **Juego de herramientas del desarrollador.** Este juego se utiliza para enlazarse con facilidad a las aplicaciones de escritorio y complementar componentes para el análisis estadístico y la elaboración de cuadros, gráficas y reportes. La disponibilidad de un lenguaje de programación orientado a objetos con una interfaz gráfica, así como el intercambio de datos mediante técnicas OLE (vinculación e incorporación de objetos), reforzará al analista empresarial para incluir el análisis estadístico en aplicaciones de escritorio para soporte de decisiones.
- **Tiempo de respuesta responsable.** Este período, que se mide en minutos o incluso en horas, es aceptable para algunas decisiones empresariales. Por supuesto, existen excepciones, como en la industria de seguros; el tiempo de respuesta en días es inaceptable ya que la relevancia del análisis declina al desactualizarse los datos, y la oportunidad se aleja.

Descubrimiento de conocimientos

En el análisis estadístico y de datos, es esencial que el analista empresarial conozca cuáles son las variables antes de iniciar el análisis. ¿Qué sucede si no conoce las variables, y si los datos son tan extensos y tienen tantas variables que no sabe por dónde empezar? Quizás no se esté del todo satisfecho con el análisis estadístico y se sospecha que los datos ocultan algo. En estas situaciones, el analista empresarial requiere de la tecnología y las herramientas para el descubrimiento de conocimientos.

El descubrimiento de conocimientos tiene sus raíces en la inteligencia artificial y el aprendizaje con máquinas. Algunas definiciones para el descubrimiento de conocimiento son las siguientes:

- El descubrimiento de conocimientos es extraer de los datos información implícita, no trivial, que no se conocía y potencialmente útil.
- El descubrimiento de conocimientos es el proceso de buscar en los datos sin establecer por adelantado una hipótesis o cuestión, e incluso así encontrar información inesperada e interesante de relaciones y patrones entre los elementos de datos o reglas empresariales importantes en todos los datos investigados y analizados.

- El descubrimiento de conocimientos significa descubrir hechos empresariales antes desconocidos en los gigabytes de datos del *DW* o del mercado de datos.

El uso de herramientas de descubrimiento de conocimientos comprende las tres fases siguientes:

- Preparación de datos.
- Descubrimiento de patrones y relaciones de interés.
- Presentación de los descubrimientos al analista empresarial para evaluación y exploración.

La primera fase, la preparación de los datos, es la alimentación estándar del *DW*. La principal diferencia es que se necesitan datos de detalle, no resumidos. La selección de cuáles datos se necesitan depende de los conocimientos en la materia que posee el analista empresarial y del conocimiento que tiene el administrador del contenido del *DW*.

En la segunda fase, se avanza usando el sistema de descubrimiento de conocimientos. Este sistema es conducido por la base de conocimientos para extraer los datos correctos del *DW*. Por lo regular, la interfaz al almacén de datos es con SQL. A continuación se aplican los algoritmos de descubrimiento de conocimientos, seleccionados por el analista, para detectar patrones y relaciones entre los datos extraídos.

En la fase final se presentan los descubrimientos al analista empresarial usando una herramienta de estación de trabajo y una herramienta de escritorio adecuada.

El analista empresarial guía y controla el sistema de descubrimiento de conocimientos desde una estación de trabajo cliente. La base de conocimientos se almacena en el sistema de descubrimiento de conocimientos o en el sistema del *DW*.

El sistema de descubrimiento de conocimientos requiere de un servidor de cómputo potente, el cual pudiera ser un procesador en paralelo. La separación del procesador de cómputo de la plataforma de almacenamiento de datos ayuda a afinar el sistema, ya que ahora el desempeño y los embotellamientos de entrada/salida se abordan por separado.

Otras técnicas y herramientas de la minería de datos

Existen otras técnicas y herramientas para minería de datos. Algunas de ellas acaban de surgir de los centros de investigación, mientras que otras tienen aplicaciones específicas.

Sistemas de visualización. Los sistemas de visualización permiten a los analistas empresariales hacer descubrimientos permitiéndoles analizar los datos de manera gráfica con muchas variables, y después ver patrones y relaciones que sería muy difícil determinar mediante algoritmos de máquina, sin importar las capacidades de cómputo del sistema.

Sistemas de información geográfica. Estos sistemas relacionan los datos del *DW* en diferentes ubicaciones físicas con representaciones geográficas. El analista empresarial

aprecia los datos en un contexto geográfico y compara territorios para un mismo producto, o diferentes productos para un mismo territorio. También es posible analizar los datos temporales en el *DW* para ver los cambios a través del tiempo para ventas, inventario de productos, etc., dentro de áreas geográficas determinadas.

Análisis fractal. La base de datos multidimensionales proporcionan una abundante información analítica y tienen un tiempo de respuesta rápido, aunque sufren de limitaciones de tamaño si se necesita almacenar todo el *DW*. El análisis fractal pretende identificar los patrones usando la ciencia del caos, y en seguida empleando fractales para almacenarlos en el *DW*. La meta consiste en ofrecer respuestas estilo OLAP para un *DW* muy grande.

3.4 Diversos Proveedores de Herramientas para Data Warehouse

IBM

La solución del *DW* de IBM se denomina "*A Data Warehouse Plus!*". El enfoque de IBM consiste en extregar un conjunto completo de productos y servicios; su meta está en ofrecer una solución integrada con base en una sola arquitectura. R Finkelstein de Performance Computing, INC. declaró "parece que IBM tiene, en general y de arriba abajo, la mejor solución integrada con base en una sólida arquitectura". La familia DB2 es el ancla de la estrategia del *DW* de IBM.

IBM tiene la ventaja de que la mayoría de los datos operacionales que se van a extraer y almacenar en el *DW* residen en sistemas IBM. De ahí que la integración apegada sea un resultado natural. El reto en este momento es que casi todos los productos de IBM son para plataformas IBM. R. Finkelstein también declaró "la única área de preocupación es la conectividad entre herramientas de proceso frontal de otros fabricantes y la familia DB2 de bases de datos relacionales". Las ofertas de IBM en esta área son bastante débiles. Actualmente, IBM tiene un programa de sociedades para reclutar más socios de productos y servicios.

IBM ofrece tres soluciones de *DW*:

Mercado de datos independientes. Se concentra en un departamento o función empresarial de la organización, se maneja con una ayuda mínima de la organización en tecnología de la información.

Mercado de datos dependiente. Es similar al mercado de datos independiente, pero la organización en tecnología de la información controla y administra la conectividad con las fuentes de datos.

Data Warehouse global. Lo implementa y administra la organización en tecnología de la información, y se apoya en una arquitectura de empresa. Esto puede implicar un *DW* centralizado o uno distribuido con mercados de datos. En la figura 3.10 se muestran las principales funciones de *DW* que maneja IBM.

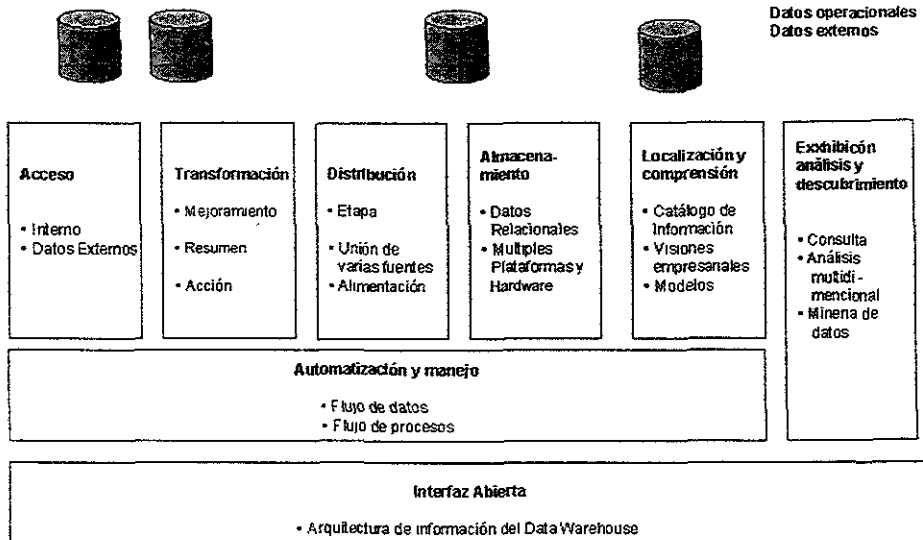


Figura 3.10 IBM: Funciones del DW.

Una solución para IBM global de DW puede basarse en DB2 para MVS, o DB2 para AIX *Parallel Edition*. La solución visual para el DW se asienta en DDB2 para OS/2 o DB2 para AIX, y se ofrece como el punto de entrada de bajo costo, IBM aborda la administración de metadatos con la familia de *DataGuide*.

En la minería de datos, IBM tiene una familia reciente de herramientas de descubrimiento de conocimientos. Las técnicas de descubrimiento de conocimientos que aplican estas herramientas son las asociaciones, los patrones secuenciales, los clasificadores y el agrupamiento. Además, IBM Research ofrece *Parallel Visual Explorer*, una poderosa técnica de análisis para visualizar espacio dimensional con coordenadas paralelas, una alternativa de representación geométrica para datos multidimensionales. El *Parallel Visual Explorer* se está utilizando en aplicaciones de minería de datos para análisis financieros, análisis comercial y manufactura.

Oracle

Oracle ofrece una solución amplia, con un enfoque en las atribuciones principales en el almacenamiento y la administración de datos, las aplicaciones de mercado vertical y las herramientas de acceso y desarrollo de datos. La solución del DW se caracteriza por dos atributos; la extensión de la línea de productos Oracle, y la cantidad de socios en su *Warehouse Technology Initiative (WTI)* (ver figura 3.11). Oracle también aprovecha su *Systems Management Tools Initiative (SMTI)* para cubrir las necesidades de administración de sistemas y vigilancia del desempeño. La fuerza de la oferta de Oracle proviene de su motor RDBMS, Oracle 7, el cual mejora constantemente para satisfacer

los requerimientos de funcionalidad del *DW*, de sus aplicaciones de mercado vertical que ofrecen el potencial de *DW* prefabricados, la amplitud de su tecnología para desarrollo y análisis de datos y la disponibilidad de productos de software de otros fabricantes.

La figura 3.11 muestra que Oracle depende de sus socios para el refinado y la reingeniería de datos, que aparecen en la figura como herramientas de transformación con pulimento.

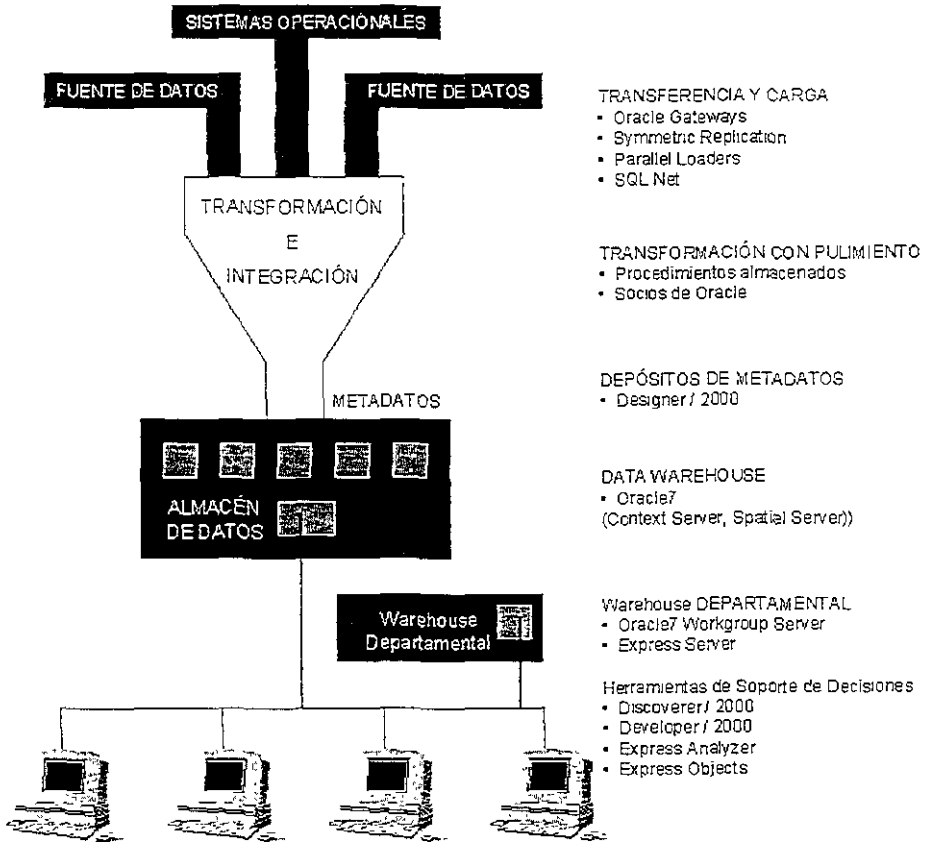


Figura 3.11 Oracle: Componentes del *DW*.

Oracle agrupa a sus socios de WTI en tres categorías: productos de diseño (tres socios), productos de construcción (15 socios) y productos de análisis (20 socios). Desde la perspectiva del *DW*, los socios de SMTI se agrupan en cinco categorías: productos de administración, para controlar bases de datos y redes (10 socios); productos de análisis, para mejorar el desempeño y la utilización de recursos (15 socios); productos de vigilancia y diagnóstico (20 socios); productos de operaciones para automatizar tareas administrativas regulares, por ejemplo respaldo y calendarización de extracciones del *DW*

(24 socios); y componentes de interoperabilidad, para permitir las comunicaciones entre sistemas heterogéneos y redes (cuatro socios). Observe que bastantes socios figuran en varias de las categorías, tanto para WTI como para SMTI.

Hewlett Packard

Hewlett Packard conduce su oferta de *DW* mediante su programa *OpenWarehouse*, que se caracteriza como una estructura para construir *DW* basada en componentes de *hardware* y *software* de primera calidad de HP y de otros fabricantes. Las anclas de lo que ofrece HP son sus plataformas UNIX y su producto *Intelligent Warehouse*, un *software* de administración del *DW*. La estructura *Open Warehouse* permite elegir un RDBMS, las herramientas de refinamiento y reingeniería, y las herramientas de acceso y uso de datos. El programa de sociedades de *Open Warehouse* se orienta específicamente a reclutar socios que ofrezcan las opciones mencionadas.

El programa *Open Warehouse* también ofrece metodología de consultoría y servicios para un rápido despliegue del *DW*. *Intelligent Warehouse* incluye una de las pocas herramientas para vigilar el uso del *DW*.

Sybase

Las estrategias corporativas de Sybase se concentran en tres mercados: el procesamiento en línea de transacciones, el *DW* y el soporte de decisiones, y "despliegue masivo" de información en toda la empresa. En su estrategia de *DW* se incluye su estructura *Warehouse WORKS*. La fortaleza de Sybase reside en su motor RDBMS (System 11), en la capacidad de conexión y acceso de su base de datos (OmniCONNECT), y en sus herramientas de desarrollo (Powerbuilder). Sybase continúa ampliando la línea y funcionalidad de sus productos por medio de adquisiciones. Por ejemplo, se está mejorando el motor para *DW* con características como la indización de bits, adquirida de *Expressway Technology*.

Al igual que sus competidores, Sybase continúa integrando un grupo de socios para su solución de *DW*.

Informix Software

La estrategia de *DW* de Informix Software apunta al crecimiento del mercado por su motor RDBMS basado en su *Dynamic Scalable Architecture*. La arquitectura de su *DW*, consta de cuatro tecnologías –base de datos relacional, *software* de administración del *DW*, herramientas de acceso a datos y plataforma de sistemas abiertos–.

Informix ha establecido sociedades para tres tecnologías con diversos fabricantes: *software* de administración del *DW*, herramientas de acceso a datos y plataforma de sistemas abiertos. Para mantener competitivo su motor de bases de datos, Informix primero adquirió *Stanford Technology* y su línea de productos OLAP, y después adquirió *Illustra Information Technologies* por su tecnología de manejo de datos no tradicionales.

La tecnología de ilustra se integrará en el motor relacional para mejorar el manejo de datos espaciales, de video, de texto y otros datos no numéricos.

AT&T GIS

La estrategia principal de AT&T GIS es solventar las necesidades empresariales donde convergen los mundos de los sistemas de soporte de decisiones y operacionales. Ofrece una solución sofisticada denominada *Enterprise Information Factory* (EIF), aprovechando la experiencia con sus sistemas *Teradata* de administración de base de datos y sus tecnologías de procesamiento en paralelo.

Desde la perspectiva de la EIF, - el *DW* es un sistema pasivo de soporte de decisiones – formula preguntas, obtiene respuestas y en seguida toma decisiones. La EIF es un sistema activo en donde las decisiones de un *DW* se convierten en acciones. La EIF es como un *DW* operacional activo. La estructura de la EIF concibe una evolución a partir de sistemas actuales de un *DW* y un ambiente operacional de comunicaciones a través de aplicaciones activas de la industria. Estas aplicaciones activas de la industria convierten las decisiones derivadas del *DW* en acciones que afectan el sistema operacional.

El mercado de AT&T GIS se concentra en industrias donde las empresas desean utilizar la tecnología del *DW* para modificar la base de competencia y racionalizar y aprovechar las relaciones de la empresa con sus clientes. Los sectores de la industria identificados incluyen la banca, los seguros y las telecomunicaciones.

SAS Institute

SAS Institute ha proporcionado una sólida administración de datos, análisis de datos y funcionalidad de reportes a lo largo de 20 años. Se espera que su solución del *DW* aproveche su capacidad en lo siguiente:

- Acceso de datos con su motor de extracción, que maneja muchos depósitos de datos operacionales, tanto relacionales como no relacionales.
- Transformación y manipulación de datos mediante su 4GL.
- Motor de almacenamiento de datos con funcionalidad multidimensional.
- Un amplio e impresionante conjunto de métodos y herramientas analíticos para el procesamiento informático, y el procesamiento analítico o multidimensional.
- Análisis estadístico para minería de datos.

Red Brick Systems

Red Brick Systems presenta su *Universal Data Warehouse Blueprint*, para auxiliar a las empresas a construir y desarrollar *DWs* desde un ámbito departamental hasta toda la empresa. Ofrece herramientas para la administración de *DWs* diagnósticos de consultas, administración de copias y de *metadatos*. Desde la perspectiva de la arquitectura de referencia, ofrece productos para el bloque del *DW* y para las capas de administración de datos y de infraestructura.

Silicon Graphics

La estrategia del *DW* de Silicon Graphics se concentra en su plataforma de procesamiento de cómputo en paralelo y en su poderosa tecnología de visualización de datos.

Pyramid Technology

La estrategia de Pyramid Technology se orienta a auxiliar a sus clientes en *DWs* de despliegue de operaciones críticas, los cuales funcionan en el ambiente operacional. Los *DWs* operacionales requieren de altos niveles de disponibilidad de datos, opciones de modificación de escala e integridad de datos.

Pyramid ofrece una solución de *DW* completa a través de su programa *Smart Warehouse*. Este programa se propone integrar la tecnología de servidor UNIX con productos asociados para extraer datos, cargar y alimentar el *DW* y acceder y analizar datos. Este es un programa de tres fases que incluye una metodología: definición del proyecto piloto y sistema de producción.

Sequent Computer Systems

La solución del *DW* de Sequent converge en su experiencia en arquitecturas de hileras múltiples. En esta arquitectura, un área del escenario denominada *Warehouse* de distribución es un área común a partir de la cual se alimentan los mercados de datos. Los datos operacionales se envían a los mercados de datos a través del *DW* de distribución. Los mercados de datos son depósitos de datos orientados a un tema, que carecen de normas debido a esquemas de estrella y de *snow flake*, para mejorar la respuesta de las consultas.

El *DW* se considera un conjunto virtual del *Warehouse* de distribución y de los mercados de datos. Según Sequent, los *Warehouse* de distribución son pequeños y no grandes y los mercados de datos son grandes y no pequeños.

El *Warehouse* de distribución sólo necesita contener los datos necesarios para abastecer a los mercados de datos; además, el *Warehouse* de distribución sólo puede contener datos con normas y un mínimo de índices – lo cual facilita los aspectos de integridad de los datos-. Por lo regular, los mercados de datos contienen de dos a tres veces los requerimientos de almacenamiento, debido a la carencia de normas y a los índices.

Además de su tecnología en paralelo con base en servidores UNIX de propósito frontal, Sequent ofrece las siguientes herramientas:

- **Decision Point for Financial.** Un *DW* “preparado para culminarse” con datos financieros corporativos, para clientes de Oracle Financial. La solución de inicio rápido incluye el modelo del *DW*, el *software* de extracción de datos

Oracle Financial, una base de datos y servidores y herramientas de consulta y reportes.

- **Decision Point Method.** Una metodología de punta a punta para construir un *DW* usando un esquema de estrella, un prototipo, y un enfoque reiterativo.

Information Builders

Information Builders, INC. (IBI), ofrece una estructura de *DW* llamada *Enterprise Data Access/Data Warehouse* que complementa su fortaleza en *middleware* EDA/SQL, y su base de datos FOCUS y 4GL. Además, la estructura incluye una metodología y una herramienta de análisis de *DWs* denominada *SiteAnalyzer*. Esta es una herramienta basada en Windows para acceder datos y vigilar y analizar la utilización. Proporciona información acerca de cuáles datos operacionales se acceden actualmente para entender las necesidades y demandas de datos del usuario, antes de diseñar y construir el *DW*. Es también una herramienta de planeación del *DW* y de recopilación de requerimientos.

Prism Software

Prism Software se encuentra exclusivamente en el mercado de *DWs*. Su solución se basa en el enfoque conceptual y la metodología de desarrollo propuestos por W. H. Inmon, uno de sus fundadores. Su estructura arquitectónica incorpora productos que funcionan activamente para ofrecer una solución completa (ver figura 3.12). La estructura comprende un enfoque conducido por *metadatos*, el cual permite captar, almacenar y administrar información de los cambios del *DW* a través del tiempo.

El administrador de *DW* de Prism extrae, transforma e integra datos de sistemas operacionales, y genera un directorio de información de *metadatos* en el *DW*. El administrador de directorio de Prism está diseñado para ayudar a los usuarios empresariales a comprender y utilizar el contenido del *DW*.

Prism también ofrece modelos de datos prefabricados denominados *Inmon Generic Data Models* – los primeros en la industria-. Estos modelos proporcionan plantillas de diseño del *DW* para diversas industrias y funciones empresariales.

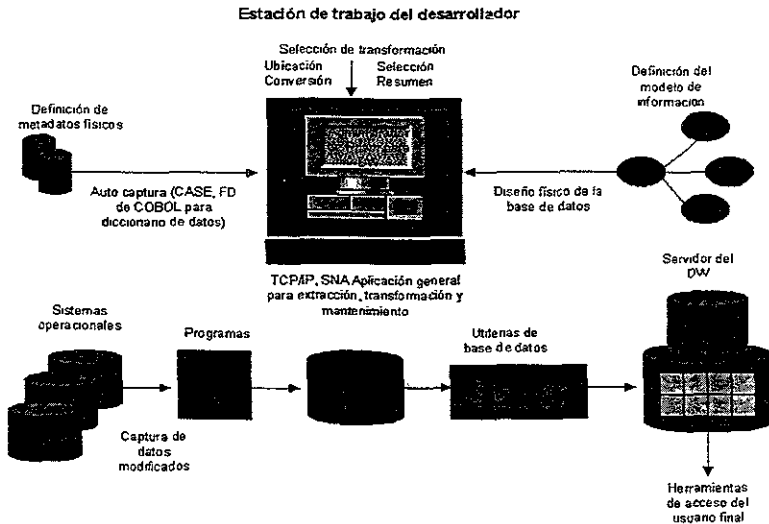


Figura 3.12 Prism: Arquitectura del Data Warehouse.

Informatica Corporation

La visión de *DW* distribuido de Informatica Corporation, consiste en mercados de datos interconectados. Su producto *OpenBridge* es una solución conducida por metadatos para extraer y almacenar datos (ver figura 3.13).

OpenBridge consta de cinco componentes alrededor de un depósito común de metadatos y del depósito de datos en el *DW*:

- **Diseñador.** Consiste en un analizador fuente, un diseñador del esquema de *DW* y un diseñador de transformación. No hay 4GL para transformación, sólo las reglas de extracción y transformación que se generan y almacenan en el depósito para que las cargue el servidor de carga.
- **Administrador del servicio.** Controla y maneja la alimentación del *DW*; carga inicial, restauración total o sólo cambios.
- **Administrador del depósito.** Crea y da mantenimiento al depósito *OpenBridge* y sus *metadatos*. También contiene un examinador de *metadatos*.
- **Servidor de carga.** El componente clave que extrae datos de las fuentes de datos operacionales seleccionadas, los transforma de acuerdo con las reglas empresariales que genere el diseñador y alimenta el *DW*.

- **Administrador de captura de cambios.** Refresca el DW con cambios en las fuentes de datos operacionales.

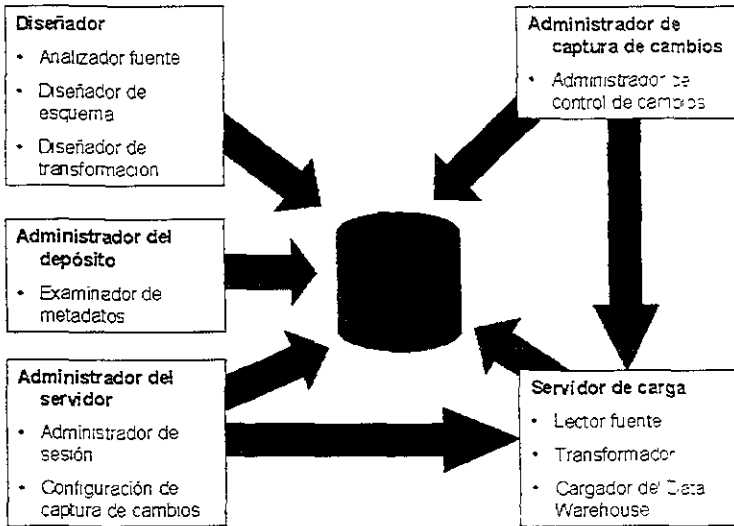


Figura 3.13 Informática: Arquitectura OpenBridge.

Vality Technology

Vality ofrece una tecnología y una metodología para investigar, estandarizar , preparar, transformar e integrar los datos a partir de varias bases de datos operacionales, datos de herencia y fuentes externas. El objetivo consiste en preparar los datos y elevar su calidad antes de ser trasladados al DW. La calidad de los datos ha sido y continúa siendo el punto débil de muchos proyectos de DW.

El producto principal de la compañía, *Integrity Data Re-engineering Tool*, solo está disponible para sistemas IBM MVS, donde de cualquier manera existen la mayoría de los datos operacionales. Una característica única del producto es su capacidad para profundizar en los datos reales operacionales o de herencia para hacer surgir *metadatos* empresariales esenciales, ausentes en los diccionarios y cuadernos de datos tradicionales. Es posible utilizar esta minería de *metadatos* para descubrir entidades ocultas, relaciones de entidad y reglas empresariales desconocidas o no documentadas.

Evolutionary Technologies International (ETI)

La estrategia de ETI consiste en automatizar y facilitar el traslado de datos entre ambientes de almacenamiento diferentes, lo cual ahorra tiempo y costos de la conversión

manual de datos. Su conjunto de herramientas *ETI EXTRACT* es un componente de una solución completa de *DW* dentro de muchas estructuras. El conjunto de herramientas ofrece lo siguiente:

- Recopilación de datos, conversión y traslado a *DWs*.
- Interfaces para herramientas especializadas de refinado y reingeniería de datos.
- Generación y ejecución automática de programas en el lenguaje apropiado de la fuente y la plataforma del *DW*, junto con las descripciones y el JCL necesarios.
- Una opción de *metadatos* para rastrear las definiciones del esquema, las correlaciones de las fuentes con los objetivos, las reglas empresariales, las reglas de transformación, y las relaciones entre bases de datos.
- Una interfaz gráfica de usuario para definir los criterios de selección de datos, las condiciones y reglas de transformación y movimiento.

El conjunto de herramientas de ETI maneja una interfaz bidireccional entre las fuentes de datos operacionales y el *DW*. Además, administra el acceso a bases de datos nativas.

Carleton

El enfoque principal de Carleton es el *DW*. Ofrece una solución completa, desde la planeación y metodología hasta la construcción del *DW*. Su producto principal, *Carleton Passport*, se propone automatizar el desarrollo y mantenimiento del *DW*. *Passport* utiliza un ambiente conducido por metadatos para construir y alimentar el *DW*. La línea de productos *Passport* ofrece lo siguiente:

- Un directorio centralizado de *metadatos* con acceso y control global. Todas las definiciones de fuentes y bases de datos objetivo se almacenan en el directorio en forma relacional.
- Acceso directo a bases de datos de herencia para extracción y transformación.
- Formateo de datos para el ambiente relacional objetivo.
- Lógica completa de transformación para el refinado y reingeniería de datos.
- Propagación de cambio *Delta* para el *DW*.
- Una plataforma para desarrollar la aplicación *Passport* con el propósito de cargarla en la plataforma del *DW*.

La línea de productos Carleton está disponible para todas las macrocomputadoras IBM y compatibles, así como para las PC y funciona en cooperación con los ambientes operativos S/370, AS/400 y OS/2.

Praxis International

Praxis ofrece el *OmniReplicator* como parte de su programa de *DW OmniWarehouse*. *OmniReplicator* se usa para manejar una duplicación bidireccional entre depósitos de datos heterogéneos. *OmniReplicator* ayuda a los usuarios del *DW* a trasladar datos de bases de datos operacionales hacia *DWs* o *datamarts*. Un producto adjunto, *OmniReplicator Administrator*, permite que el administrador de base de datos del *DW* configure los esquemas de duplicación.

Arbor Software

Arbor se orienta a ofrecer software de base de datos multidimensionales de alto desempeño, para aplicaciones empresariales complejas de planeación y análisis. Sus productos incluyen los siguientes:

- ***Essbase Analysis Server*** Un motor multidimensional disponible para plataformas UNIX, NT y OS/2.
- ***Essbase Application Tools***. Módulos de software para conversión monetaria, profundización en SQL, interfaz SQL y paquete de herramientas ampliado para hoja de cálculo.

Pilot Software

Pilot Software se concentra en proporcionar a los usuarios empresariales y profesionales en tecnología de la información las herramientas y soluciones para acceder y analizar información empresarial de misión crítica. Su paquete *LightShip Suite* es un sistema de *software* cliente/servidor para construir y utilizar soluciones basadas en tecnología OLAP. El paquete *LightShip Suite* incluye:

- ***LightShip Server***. Un servidor de datos multidimensionales basado en tecnología OLAP.
- ***LightShip Modeler***. Un Constructor de modelos para aplicaciones como modelos financieros, presupuestos, pronósticos, planes estratégicos y análisis operacionales.
- ***LightShip Professional***. Una interfaz gráfica de usuario de cliente para acceder y analizar datos.
- ***LightShip Link***. Una herramienta de acceso común a fuentes de datos relacionales.

Pilot ofrece también una herramienta *LightShip Sales and Marketing Intelligent System* (SMIS) una solución lista para usarse, para profesionales de ventas y comercialización. Además de *LightShip Suite*, SMIS contiene módulos de análisis 80/20, posicionamiento y los reportes estándar.

Dimensional Insight

Dimensional Insight ofrece un paquete integrado de productos para análisis y reportes llamado *CrossTarget*, el cual consta de dos componentes:

- **Builder.** Un motor multidimensional basado en un servidor.
- **Diver.** Una interfaz gráfica de usuario de cliente. Una característica interesante de *Diver* es su capacidad de unir varios modelos multidimensionales para crear un nuevo modelo virtual.

Information Advantage

Information Advantage (IA) se dedica exclusivamente a aplicaciones de análisis empresarial que apoyan las inversiones estratégicas de *DW*. Sus productos están diseñados en una arquitectura de tres hileras: cliente/servidor, servidor/céntrico. La línea de productos de IA, *DecisionSuite*, es una solución integrada con una arquitectura para ofrecer análisis relacional OLAP en *DWs* y *datamarts*. Incluye funciones como las siguientes:

- Entrega de información, reportes y alertas a usuarios empresariales que utilizan *software* de tecnología de agente.
- Aplicaciones operacionales tales como análisis del tipo “¿qué pasa si ...?”, cuadros empresariales e integración a sistemas de ubicación GIS.
- La capacidad de apoyar al usuario para formular reportes, analizar escenarios y agentes de instrucciones para navegar en el *DW* y producir reportes y alertas.
- Manejo de recursos y seguridad para los administradores de bases de datos y los administradores empresariales.
- Información de las colaboraciones entre usuarios empresariales.

La arquitectura de tres hileras ofrece opciones para modificar la escala y el desempeño haciendo cambios tanto en el cliente como en el servidor de base de datos. IA es miembro de diversos programas de sociedad.

Prodea Software

La estrategia de Prodea es ofrecer herramientas de soporte de decisiones integradas con aplicaciones de flujo de trabajo. La meta consiste en crear ciclos de soporte de decisiones en los cuales se produzcan acciones, no sólo reportes. Prodea ofrece dos productos adjuntos:

- **ProdeaBeacon.** Una herramienta relacional OLAP para análisis multidimensional disponible en una arquitectura de tres hileras, la cual ofrece un mejor desempeño y opciones de modificación de escala haciendo cambios tanto en el cliente como en el servidor de base de datos.
- **ProdeaSinergy.** Una herramienta de flujo de trabajo que se emplea para integrar aplicaciones de escritorio en flujos de trabajo automatizados.

Combinados, *ProdeaBeacon* automatiza la recuperación de información del *DW*, y *ProdeaSinergy* automatiza la divulgación de dicha información a los usuarios empresariales.

MicroStrategy

MicroStrategy se dedica a proporcionar productos OLAP relacionales (ROLAP) para auxiliar a los usuarios empresariales y a los profesionales en tecnología de la información a realizar análisis multidimensionales de la información almacenada en *DWs* y mercados de datos relacionales. Su arquitectura tiene un motor central ROLAP que proporciona a los usuarios empresariales una visión conceptual y multidimensional de los datos del *DW* y transforma en forma dinámica, las consultas de datos en planes de ejecución SQL. La arquitectura ofrece opciones de una, dos o tres hileras con el propósito de manejar *DWs* muy grandes. Dentro de esta arquitectura, MicroStrategy ofrece los siguientes productos:

- **DSS Agent.** Un análisis ROLAP adecuado, consulta y reportes, penetración y pivoteo, y automatización del flujo de trabajo mediante agentes de software.
- **DSS Server.** Un motor ROLAP con un controlador de consultas basado en reglas, un administrador de *DW* y un programador de trabajos.
- **DSS Executive.** Se usa para construir un Sistema de Información Ejecutiva acerca de las consultas y reportes de soporte de decisiones existentes.
- **DSS Architect.** Una herramienta de diseño para definir el modelo multidimensional para el depósito de datos relacional.

Brio Technology

La herramienta de Brio Technology *BrioQuery* ofrece un procesamiento informático visual con un análisis multidimensional integrado. Su arquitectura está diseñada para apoyar la capacidad de desempeño inherente en un *DW* diseñado con el esquema de estrella.

Tiene también un depósito para la administración central de consultas compartidas y su distribución automática. Utiliza los metadatos de la base de datos. *BrioQuery* está disponible para ambientes Macintosh y PC y se ofrece en las siguientes tres modalidades:

- **Navigator.** Los usuarios empresariales consultan sólo mediante modelos de datos predefinidos.
- **Explorer.** Los usuarios expertos consultan con modelos de datos predefinidos y crean nuevos modelos.
- **Designer.** Los profesionales en tecnología de la información pueden crear consultas predefinidas, reportes y modelos de datos.

IQ Software

IQ Software pretende principalmente producir funciones de acceso, consulta, reportes y OLAP para encauzar sin contratiempos los datos en un *DW* hacia la inteligencia empresarial. Su línea de productos *IQ/Objects* tiene una arquitectura para abordar los requerimientos de ambientes de *DW*. Capacidad para modificar la escala de los datos y de los usuarios, y administración de consultas complejas. Los productos de IQ incluyen:

- **IQ/Objects.** Consultas con base en objetos y herramientas de reportes que acceden depósitos de datos heterogéneos, así como correlacionar resultados de consultas múltiples.
- **IQ/SmartServer.** Servidor basado en una arquitectura de tres hileras para la separación inteligente de las funciones de acceso de datos y reportes entre clientes, servidores de aplicaciones y servidores dbase de datos.
- **IQ/Vision.** Una herramienta dinámica OLAP para el análisis multidimensional y de penetración de datos.

Business Objects

La estrategia de Business Objects consiste en ofrecer herramientas gráficas cliente/servidor que permitan a los usuarios manipular y analizar bases de datos relacionales.

Information Harvesting

Information Harvesting se dedica exclusivamente al área de minería de datos. Su producto *International Harvester* es una herramienta híbrida de descubrimiento de conocimientos que combina algoritmos de árbol de decisiones e inducción de reglas, estadísticas y razonamiento indistinto.

Information Discovery

Information Discovery proporciona la herramienta de descubrimiento de conocimientos iDIS (Information Discovery System), para extraer y localizar patrones ocultos, reglas empresariales y anomalías en los datos del DW. IDIS utiliza estadísticas de manera interna y aplica un enfoque de descubrimiento de reglas para generar y probar hipótesis. El ciclo de formulación y prueba automática de hipótesis continúa hasta que surgen reglas y patrones interesantes. Las reglas de descubrimiento se emplean para modelos de predicción y para la detección de anomalías. Un producto adjunto, *Neural IDIS*, combina la tecnología de red neural con el descubrimiento de reglas.

HNC Software

HNC Software se concentra en la aplicación de tecnología de red neural para el descubrimiento de conocimientos, a fin de utilizarlos para modelos predictivos. Ofreciendo los siguientes productos:

- **DataBase Mining Workstation (DMW).** Una herramienta de propósito general para crear y utilizar modelos de red neural.
- **DataBase Mining MarkSman.** Una herramienta de red neural dirigida a aplicaciones de comercialización.

SPSS

Los productos SPSS se emplean en aplicaciones de minería de datos mediante técnicas de análisis estadístico. Su herramienta de conexión neural se usa para modelos predictivos, clasificación y segmentación de datos. SPSS ofrece diversas herramientas que manejan una lista completa de técnicas de análisis estadístico.

Resumen

El mercado comercial del DW ha producido con rapidez una gran cantidad de fabricantes: desde los que tienen ingresos de varios miles de millones de dólares hasta los que sólo un millón; desde fabricantes con muchos productos y estructuras, hasta los proveedores de nicho o de un solo producto; y desde compañías bien establecidas hasta las de reciente aparición. Al mismo tiempo ningún fabricante por sí solo tiene la amplitud de productos para satisfacer totalmente las necesidades de las empresas. En este ambiente, en el área de tecnología de la información se debe tener mucho cuidado al elegir fabricantes y productos, entendiendo tanto el mercado en tecnología de DW como las inversiones y compromisos de los fabricantes con las soluciones del DW. Esto ayudará a la empresa a comprender y manejar los riesgos.

Por último, las soluciones de DW son prometedoras, aunque recientes, y hay mucho trabajo por hacer antes de que el DW desarrolle todo su potencial. Las organizaciones en tecnología de la información deben avanzar con cautela y estar listas para una intensa tarea de integración de sistemas. Incluso una autoridad en el DW como Aarón Zornes de

META Group dice que la mayoría de estos productos no funcionan juntos, y sugiere que las empresas comiencen con soluciones sencillas de soporte de decisiones.

Al avanzar con precaución con proyectos pequeños manejables, las empresas deben tener en cuenta cuatro eventos clave:

- Incrementar la madurez de la tecnología de base de datos relacional con la incorporación de ampliaciones de objetos y características del *DW*.
- Integración de herramientas de flujo de trabajo para automatizar los siempre crecientes requerimientos del *DW*.
- Determinar estándares de la industria para que los productos funcionen juntos.
- Considerar las lecciones aprendidas de los pioneros.

Conclusión

Herramientas seleccionadas:

Por lo anterior es necesario elegir una alternativa tecnológica y económicamente viable, indicando las razones de la selección.

En este caso, siendo Sybase la herramienta que nace bajo la arquitectura cliente/servidor es la que mejor satisface las necesidades requeridas para acceder a diferentes plataformas por medio del cliente y comunicarse al servidor de una forma estándar, segura e integrar la información con el formato único de una arquitectura definida, pero con la posibilidad de integrar en otras arquitecturas.

Trabajar bajo la arquitectura cliente/servidor proporciona las siguientes ventajas

- Conectar clientes a través, de una capa intermedia, a fuentes de información diversa.
- Flexibilidad y escalabilidad asociada con colocar componentes en diferentes máquinas o en dónde más necesita.
- Permite convivir con *front-ends* con transparencia.

Para construir un *DW*, Sybase cuenta con los siguientes productos:

- *Open Server*
- *Open Cliente*
- *Replication Server*
- *Cyrano* (Para tareas de monitoreo del SQL Server)

La especificación de requisitos para la producción de *software* consta de los siguientes aspectos:

- Panorama del producto.
- Ambientes de desarrollo.
- Operación.
- Mantenimiento.
- Interfaces externas.
- Flujo de datos.
- Especificaciones funcionales.
- Requisitos de operación.
- Criterios de aceptación.
- Guías de sugerencias y restricciones.

Requisitos de software y hardware:

- Aplicaciones que van en los servidores aplicativos y sus recursos de memoria son:

Aplicación	Recursos de Memoria	Reales
SQL Server	20 Mb	14 Mb
Open Server	8 Mb	2.4 Mb
Replication Server	10 Mb	7.2 Mb
Cyrano	10 Mb	2.4 Mb
Back Up Server	4 Mb	4 Mb, no constante
Agentes Monitoreo	8 Mb	2.4 Mb
Sistema Operativo	16 Mb	16 Mb
Total	74 Mb	15 Mb libres Ocupa sistema

- *Software* adicional para los servidores de Windows NT.

Nombre software	Versión
Sybase SQL Server	11.0.2
Sybase Open Server	10.0.4
Sybase Replication Server	11.0.3
Sybase Open Client	10.0.4
Servidor Aplicativo	1
Measure Ware	4.000.005
Agente Monitoreo	

Software instalado

Cada uno de los clientes debe contar con el siguiente software correctamente instalado y configurado.

- **Sistema operativo**

Requisitos
Microsoft Windows NT Versión 4
Service Pack 3
Protocolo TCP/IP
Servicio FTP, Configuración Usuario FTP

La definición de requisitos se refiere a la identificación de las funciones básicas de los componentes de programación del sistema, como se verá en el capítulo 4. El resultado de la definición de requisitos, es una especificación que describe el ambiente de procesamiento (Sybase), las funciones requeridas de los sistemas (operaciones sobre documentos), restricciones de configuración sobre los programas (tamaño, velocidad y configuración de equipo), manejo de errores, subconjuntos y prioridades de instrumentación, cambios probables y modificaciones factibles (definición de tablas dinámicamente en la metadata), así como criterios de aceptación del producto, todo lo cuál se deberá decidir tomando en cuenta los detalles revisados en el presente capítulo.

En este capítulo hemos visto que las herramientas elegidas durante la planeación del sistema son de gran relevancia, y principalmente las herramientas OLAP debido a que brindan una respuesta inmediata y se utilizan para llevar a cabo la toma de decisiones en toda una empresa. En el siguiente capítulo plantaremos nuestra problemática y un análisis del middleware.

Capítulo 4

Análisis del Middleware

En este capítulo se analizarán los componentes que constituirán el *middleware*, así como, el alcance de cada uno de ellos.

4.1 Problemática del Sistema Financiero

Un sistema financiero es el conjunto orgánico de instituciones que generan, captan, administran, orientan y dirigen el ahorro y la inversión en el contexto político-económico. Y se encuentra formado por: Entidades Normativas, Intermediarios Financieros, Grupos Financieros e Instituciones de Apoyo.

Dentro de las Entidades Normativas y Reguladoras se encuentran, por jerarquía, en primera instancia la Secretaría de Hacienda y Crédito Público; dependiendo de esta se tienen las siguientes entidades reguladoras del Sistema Financiero: Banco de México (el cual funge como asesor), Comisión Nacional Bancaria y de Valores, Comisión Nacional de Seguros y Fianzas y la Comisión Nacional del Sistema de Ahorro para el Retiro.

Una Entidad Normativa y Reguladora tiene por objeto supervisar y regular en el ámbito de su competencia, a las entidades financieras, a fin de procurar su estabilidad y correcto funcionamiento; así como, mantener y fomentar el sistema financiero en su conjunto en protección de los intereses del público.

Dentro de las entidades a supervisar se encuentran: Sociedades Controladoras de Grupos Financieros, Instituciones de Crédito, Casas de Bolsa, Especialistas Bursátiles, Bolsa de Valores, Sociedades Operadoras de Sociedades de Inversión, Sociedades de Inversión, Almacenes Generales de Depósito y Uniones de Crédito.

Por lo anterior se establece que un Sistema Financiero tiene varias Entidades Supervisadas, las cuales manejan grandes cantidades de información (Bancos, Casas de Bolsa, etc.), además, existen Entidades Normativas (SHCP, CNBV, etc.), que para realizar una de sus principales labores (toma de decisiones) requieren de una concentración de información, la cual proviene de las Entidades Supervisadas.

Esto provoca varias complicaciones en el manejo de la información del sistema en su conjunto, primeramente, la información puede darse en diferentes tiempos y estos tiempos pueden cambiar de acuerdo al origen de los datos, además algunos requerimientos de información son dinámicos, por ejemplo, durante un periodo de tiempo puede ser necesario hacer un análisis de los estados financieros de las Entidades Supervisadas, pero repentinamente puede surgir la necesidad de añadir el tipo de operaciones que éstas realizan en los bancos, y finalmente, es frecuente que las reglas que se aplican a la información que se envía cambien conforme pasa el tiempo.

Al analizar los requerimientos de un sistema de información para un Sistema Financiero, la problemática mencionada nos enfrenta con los siguientes retos:

- Definir la logística de extracción de la información contemplando los tiempos y disponibilidad de los sistemas de información de las Entidades Supervisadas.
- Estructurar la logística de extracción de datos de tal forma que no varíe en el tiempo, considerando que los requerimientos de información pueden o no ser dinámicos.
- Definir los métodos de validación de la información de modo único, considerando que las reglas de validación son diversas y variantes.

En este corto análisis de un Sistema Financiero, podemos aplicar los conceptos y arquitectura de *DW*, hablar de un "*DW* Bancario" que sería el sistema de información más adecuado para un Sistema Financiero Mexicano ya que tendría la capacidad de concentrar toda la información económica de una región o del país y proporcionarla en forma "amigable", transparente, oportuna y eficaz a los responsables de las decisiones sobre el rumbo y políticas de la economía en dicha región.

En la estructura de este *DW* (como en muchos) hay un punto muy importante que es la transferencia de información, esta transferencia se ve muy afectada por los tres puntos antes mencionados, la aplicación que se encarga de hacer dichas transferencias es el *middleware*, el cual debe extraer información de las Entidades Supervisadas, aplicar criterios que determinen la validez de esta información, y finalmente, enviarla al repositorio central de datos para su mejor uso; esta es la parte del *DW* Bancario que desarrollaremos.

4.2 Alcance del Proyecto

Es importante determinar cual será el alcance del proyecto; esto es, cuales son los procesos que realizará el *middleware* y la forma en que operará.

4.2.1 Objetivo

Realizar un *middleware* cuyo objetivo es automatizar los procesos de extracción, validación e integración de la información enviada por las Entidades Supervisadas a una Entidad Normativa, mediante una arquitectura flexible, escalable, configurable y adaptable, apoyada en la definición de reglas de extracción, validación e integración en una base de datos de *metadata*.

4.2.2 Características y restricciones

Los servicios proporcionados por el *middleware* son:

Recepción de información

La recepción de los diferentes documentos que envían las Entidades Supervisadas se realizará, en la mayoría de los casos, por medio de una aplicación a la que a partir de este momento denominaremos CLIENTE. Esta aplicación permitirá realizar la recepción de información mediante captura de datos e importación de archivos manual ó automática.

Para el caso de recepción de información mediante captura, el CLIENTE contará con una interfaz dinámica, de tal forma que se pueda recibir cualquier tipo de documento previamente definido en la *metadata* sin necesidad de modificar el código del CLIENTE. La interfaz dinámica permitirá la captura del número adecuado de campos por registro y del tipo y longitud especificados para los mismos.

Para el caso de la importación de archivos, se podrá realizar dicha importación siempre y cuando el formato de la información que contiene el archivo corresponda a los formatos definidos para dichos documentos. Esto significa que el CLIENTE no incluirá en forma explícita el código para validación de formatos o validaciones de sensibilidad y congruencia.

Para cada Entidad Supervisada se definirán los documentos que ésta debe enviar, de tal forma que por medio del CLIENTE se validará el que un usuario (perteneciente a una Entidad Supervisada) pueda capturar o importar solamente cierto tipo de documentos.

Los formatos que deben tener los documentos y las validaciones que se aplican a estos serán definidos en la *metadata*.

El *middleware* incluirá también un mecanismo que permita obtener la información de las Entidades Supervisadas sin necesidad de que ellas la envíen, ya sea a través de

importación de archivos o accedando directamente las bases de datos en donde reside la información de las Entidades Supervisadas. En este caso el *middleware* será el encargado de obtener la información en la fecha adecuada en forma automática.

Formatos de la información que se recibe

La información que recibe la Entidad Normativa por medio de captura, archivos o acceso a una base de datos, debe cumplir con un formato. Los formatos deben definirse en la *metadata*; una vez realizada la definición, el *middleware* podrá entender el formato asociado a un documento.

Por otra parte, se deben poder manejar diferentes versiones de un documento, cada una de las cuales puede tener formatos diferentes, de tal forma que se pueda procesar información atrasada con el formato que le corresponde, para ello cada versión debe tener asociado el periodo para el cual aplica. Por ejemplo: Si se desea enviar un documento tipo "A" del mes de diciembre de 1998; el cual fue modificado en enero de 1999, originando la versión 2 del mismo documento, se debe seleccionar el documento "A" versión 1 para el periodo 1998(año), 12 (mes).

Validación de la información

Los documentos que envían las Entidades Supervisadas son validados, en cuanto al contenido y formato de la información, para ello existen varias formas de validación: validaciones contra catálogos, validaciones por *stored procedure* y validaciones mediante fórmulas.

Las validaciones contra catálogos son aquellas que aseguran que el valor de una columna o varias columnas existan en una tabla definida como catálogo. Ejemplos de éstas son las siguientes:

- El Grupo Financiero indicado para un registro debe ser válido.
- La combinación Grupo Financiero/Institución debe ser una combinación permitida.
- La clave de la institución debe ser válida.

Las validaciones de formato permiten verificar que la información de una columna en un registro cumpla con ciertas reglas de formato o de dominio. Ejemplos de estas validaciones podrían ser las siguientes:

- El formato de la fecha debe ser año, mes y día (yyyymmdd)
- La clave de la Entidad Supervisada debe ser 8 dígitos (#####)

Las validaciones mediante expresiones se podrán aplicar a nivel de registro o a nivel de documento. En este caso la validación consistirá en la evaluación de una expresión aritmético lógica, cuyo resultado podrá ser verdadero (sí cumple con la regla de validación) o falso (sí no cumple con la regla de validación).

La expresión aritmético lógica podrá incluir los siguientes operadores: +, -, *, /, %, and, or, not, xor, >, >=, <, <=, =, != y <>. También se podrán utilizar paréntesis.

Los operandos tendrán que ser columnas de los registros del documento o bien constantes, esto para el caso de que la validación sea a nivel registro; para las validaciones a nivel documento, los operandos podrán ser constantes o las siguientes funciones agregadas que actuarán sobre columnas del documento: *max*, *min*, *sum*, *avg* y *count*.

En las expresiones se podrán utilizar las siguientes funciones para cadenas: *strlen*, *strcat*, *substr*; y las siguientes funciones numéricas: *abs*, *round*, etc.

Algunos ejemplos de las validaciones que pueden realizarse utilizando expresiones son las siguientes:

- (precio mercado-precio compra)*número de títulos = *inc* o *dec* por valuación
- Número de títulos <> 0
- Fecha de vencimiento préstamo > fecha de operación
- Tipo valor = "CC" ó "OA"
- Fecha liquidación <= (fecha operacion + 10 días naturales)

Un tipo especial de validaciones mediante expresiones son aquellas de tipo *if-then-else*, las cuales las podemos conceptualizar de la siguiente forma:

```

if expresión_1 then
    expresión_2
else
    expresión_3

```

donde *expresión_1*, *expresión_2* y *expresión_3* son expresiones aritmético lógicas como las descritas anteriormente, e incluso, *expresión_2* y *expresión_3* pueden ser también expresiones tipo *if-then-else*. Al igual que las expresiones aritmético lógicas, las expresiones *if-then-else* dan como resultado un valor lógico.

Ejemplos de validaciones que pueden ser representadas mediante expresiones *if-then-else* son las siguientes:

- Si tipo valor = "CM" entonces serie debe tener <= 7 caracteres
- Si posición = "CV" ó "VV" entonces fecha liquidación > fecha reporte
- Si fecha vencimiento - fecha inicio <= 3 entonces concepto = "C"

Todas las validaciones no consideradas hasta este punto serán consideradas como validaciones complejas y la forma de evaluarlas será por medio de *stored procedures*.

Las reglas de validación deben ser configurables, por ello se definen en la *metadata*. Las reglas de validación estarán asociadas a alguno o algunos de los documentos definidos para la extracción de información, de tal forma que el *middleware* leerá de la base de datos las validaciones que deberá aplicar a cada uno de los documentos recibidos.

Los procesos de validación no utilizarán ninguna fuente externa de datos que no sean los catálogos.

Esquema de seguridad

Las Entidades Supervisadas sólo podrán acceder al *middleware* por medio del CLIENTE para efecto de enviar información ó verificar el estatus de la información que hayan enviado. El acceso al CLIENTE estará determinado por el nombre de la Entidad Supervisada, el usuario y su *password*.

La actualización de las bases de datos en donde se definen catálogos, documentos, formatos, validaciones, usuarios, permisos, etc. solamente se podrá realizar desde la Entidad Normativa y se difundirá a las Entidades Supervisadas por medio de un esquema de replicación.

Para el caso de que el *middleware* deba de obtener la información vía un acceso remoto, la Entidad Supervisada deberá proporcionar el nombre del servidor, la base de datos, el *login*, *password* y el *stored procedure* que se tendrá que ejecutar para obtener la información del documento. Toda esta información de acceso se guardará en la *metadata*, para que así el *middleware* pueda automáticamente extraer la información con la periodicidad establecida.

La información que viaje entre las Entidades Supervisadas y la Entidad Normativa estará encriptada. Los algoritmos de encriptación serán los propios de las librerías *Client Library/Server Library* de Sybase.

Integración de la información

Una vez validada la información recibida, ésta se integrará en el repositorio central o *datamarts* ya existentes en la Entidad Normativa. Cada uno de los registros recibidos podrá integrarse en múltiples tablas o en una sola. La forma en como deberá integrarse la información estará determinada por el formato de la información y la institución que envía, de tal forma que envíos con el mismo formato, pero de instituciones diferentes, se podrán integrar en bases de datos diferentes.

Las reglas que determinan en donde se integrará la información recibida se definirán en la *metadata*.

En el caso de un *middleware* instalado en la Entidad Supervisada, aunque éste puede llevar a cabo las validaciones de formato y congruencia, la integración se realizará en los repositorios centrales de la Entidad Normativa. En el caso de que no exista comunicación con el *middleware* de la Entidad Normativa, el documento se marcará como pendiente por integrar.

La integración de la información recibida se realizará como transacción por documento, es decir, si parte de la información no es correcta no se integrará nada al repositorio central. Una vez que se haya integrado la información en el repositorio central se generará un acuse de recibo que indicará el detalle de como fue procesada la información y los errores

que se generaron. El acuse de recibo será generado por medio del CLIENTE y se registrará este hecho en la base de datos.

Uno de los principales objetivos del *middleware* es recibir e integrar al repositorio central la información a nivel del detalle que se recibe, esto es, no se contempla como parte de los procesos de validación, el que la información sea consolidada o transformada, así como tampoco se contempla el análisis, diseño e implementación del repositorio central en donde se integrará la información.

Control de información

Las Entidades Supervisadas envían a la Entidad Normativa información diaria, semanal, mensual, etc. El *middleware* deberá controlar que la información se reciba con la periodicidad establecida. Cuando no se reciba información en las fechas indicadas, se notificará a la Entidad Supervisada el retraso del envío. La notificación se realizará mediante el CLIENTE.

Las Entidades Supervisadas podrán utilizar el *middleware* con el fin de revisar su información antes de enviarla, para ello utilizarán el CLIENTE al que indicarán la opción de solo validación. De esta forma se aplicarán los procesos de validación de formato y congruencia a la información sin necesidad de integrarla al repositorio central.

Por medio del CLIENTE se podrán generar y/o imprimir acuses de recibo, reportes de estatus, etc. Por otra parte se podrá obtener información acerca de los procesos aplicados y el estatus de los datos capturados o importados.

Por medio del CLIENTE se podrá respaldar la información capturada, una vez que se hayan realizado las validaciones de formato, esto para evitar doble captura en el caso de que no se pasen las validaciones de congruencia.

En el caso de que en el acuse de recibo se haya indicado que la información enviada tenía errores, se podrá cargar nuevamente en la pantalla de captura, para de esta forma hacer las correcciones necesarias y enviar la información nuevamente.

No se contempla el respaldo de los archivos importados, en lugar de ello en el acuse de recibo se presentará información suficiente como para que el usuario pueda determinar la información que envió.

Se podrán realizar reenvíos de información, siempre y cuando exista una autorización para este propósito. Esto es, si una Entidad Supervisada envió un documento para el periodo "9902", ya no podrá reenviar este mismo documento para este periodo.

Administración de la metadata

Se contará con una aplicación que permitirá administrar de la *metadata*, a esta aplicación la denominaremos MODULO ADMINISTRADOR. Los requerimientos para esta aplicación se listan a continuación.

1. Permitirá la definición/modificación de documentos para la recepción de información.
2. Permitirá la definición/modificación de las reglas de validación aplicables a la información recibida.
3. Permitirá la definición/modificación de las reglas de integración de la información a las bases de datos operacionales.
4. Permitirá la definición de usuarios y permisos.

Esquema de depuración

Se debe contar con un mecanismo que permita la depuración automática, semiautomática y manual de la información que se va acumulando localmente en las Entidades Normativas. Mediante la base de datos se podrá indicar la obsolescencia para cada documento, así como la forma y horarios de realizar las tareas de depuración.

Restricciones generales

No se contempla la realización de ninguna herramienta de explotación o análisis de la información recibida o de la ya validada.

No se contempla ningún cambio en la configuración y diseño de las bases de datos y aplicaciones operacionales que actualmente existen.

No se contempla el análisis, diseño e implementación de las vistas de datos que podrán ser accesadas en algunas de las Entidades Normativas.

No se contempla el análisis, diseño e implementación de la red que formarán las Entidades Supervisadas y la Entidad Normativa.

No se contempla la implementación del esquema de replicación.

Una vez mencionados los servicios que proporcionará el *middleware*, a continuación se mostrará la figura 4.1, en donde se presenta el esquema de funcionalidad del *middleware*.

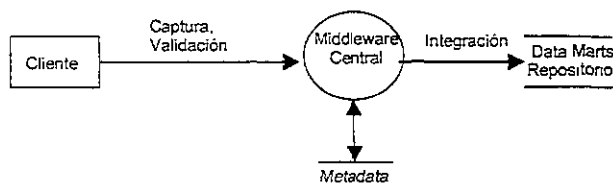


Figura 4.1 Esquema de funcionalidad del *middleware*.

En la figura anterior se muestra como una Entidad Supervisada puede enviar su documento a la Entidad Normativa vía el CLIENTE, el cual recibe la información por medio de captura o importación de un archivo para proceder a la validación del documento, y una vez validada la información ésta es integrada en el repositorio central.

4.3 Arquitectura Global

Para que el *middleware* cumpla con su objetivo se requiere de una arquitectura global , que llamaremos "Servidor Aplicativo" (SA), la cual esta formada por los componentes que acontinuacion se describen:

- **Middleware.** Es una aplicación *Open Server Multi-thread* que se encarga de realizar los servicios de validación de formato, validación de congruencia y sensibilidad, integración al repositorio central, bitácora de estatus, y notificaciones.
- **Base de datos metadata.** Es la base de datos que utiliza el *middleware* para su funcionamiento; en ella se encuentra la definición de los documentos que las Entidades Supervisadas envían a la Entidad Normativa; así como, sus formatos, reglas de validación, reglas de integración, procesos aplicados a la información, reglas de ruteo, etc.

En la *metadata* se manejará la administración de ésta, especificando los usuarios válidos, privilegios, creación de tablas temporales, etc.

- **El repositorio central.** Es la base de datos en donde se almacenan la información que han pasado por un proceso de validación y que servirá para realizar el análisis que apoye la toma de decisiones. Esta base de datos es definida por la Entidad Normativa.
- **Base de datos de catálogos.** Es la base de datos en donde residen los catálogos utilizados para los procesos de validación.
- **Replication Server.** Este componente permitirá replicar los cambios de información que se realicen en la *metadata* y en la base de datos de catálogos de la Entidad Normativa a las Entidades Supervisadas.
- **Sql Server.** Es un manejador que controla la *metadata*, la base de datos de catálogos y el repositorio central.
- **OC timer.** Es una aplicación *Open Client*, que se ejecutará repetitivamente ya que cuenta con un reloj que servirá para activar ciertos eventos (acceso a bases de datos, reenvío de información).
- **CLIENTE.** Es una aplicación gráfica encargada de realizar el proceso de recepción de información mediante la captura o importación de archivos. Esta aplicación controlará el acceso al *middleware* y será el medio por el cual un usuario de una Entidad Supervisada interactue con el sistema.

- **Módulo administrador.** Es una aplicación gráfica que permitirá el control y la administración de la *metadata* en la que se apoya el *middleware*. Por medio de esta aplicación se podrán crear los formatos de captura de información; mediante las reglas de validación se podrán dar de alta a los usuarios o *Entidades Supervisadas*, *definir permisos*, etc. Este componente existe en la Entidad Normativa.

En las siguientes secciones se muestran las diferentes configuraciones que se pueden utilizar en las Entidades Supervisadas para enviar su información a la Entidad Normativa.

4.3.1 SA Local con acceso a bases de datos operacionales

En esta configuración la Entidad Supervisada cuenta con un SA Local completo, bajo este esquema la recepción y validación de información se realiza en el SA Local. Por otra parte el SA puede extraer directamente de las bases de datos operacionales de la Entidad Supervisada la información que se requiera.

En la figura 4.2 se puede apreciar que existen dos Servidores Aplicativos (SA), uno que llamaremos SA Central y al otro SA Local, en donde el SA Central se encuentra en la Entidad Normativa y el SA Local se encuentra en la Entidad Supervisada. Para poder realizar la extracción de las bases de datos operacionales de la Entidad Supervisada el SA Local cuenta con el *OC timer*, el cual se encarga de ejecutar un proceso para la obtención de los documentos que la Entidad Supervisada envía a la Entidad Normativa; una vez obtenidos éste o estos documentos el SA Local aplica un proceso de validación a través del *middleware* y si éste es exitoso el SA Local envía el documento al SA Central, donde el *middleware* integrará dicho documento en el repositorio central. Para mantener la integridad de la información de validación se utilizará un esquema de replicación de la *metadata*.

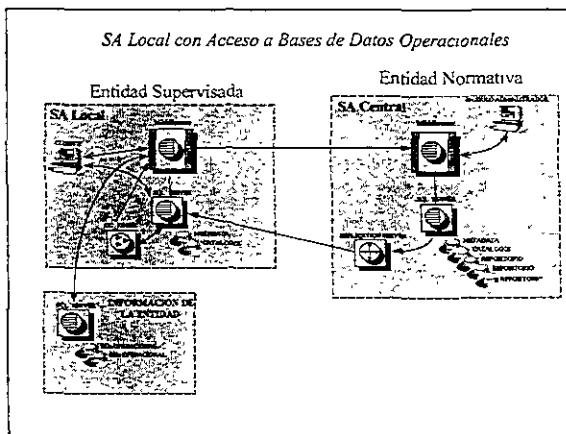


Figura 4.2 Representación de un SA accedendo bases de datos operacionales de Entidades Supervisadas.

4.3.2 SA Local sin acceso a bases de datos operacionales de Entidades Supervisadas

Esta configuración es similar a la anterior, excepto que en vez de extraer la información de la base de datos operacional de la Entidad Supervisada, ésta envía el documento vía el CLIENTE, ya sea por medio de la importación de un archivo de texto (que cumpla con el formato especificado por la Entidad Normativa), o bien, por captura manual.

En la figura 4.3 se aprecia esta configuración y al igual que la anterior, una vez que la Entidad Supervisada carga su documento se procede al proceso de validación para posteriormente realizar el envío a la Entidad Normativa e integrar el documento en el repositorio central.

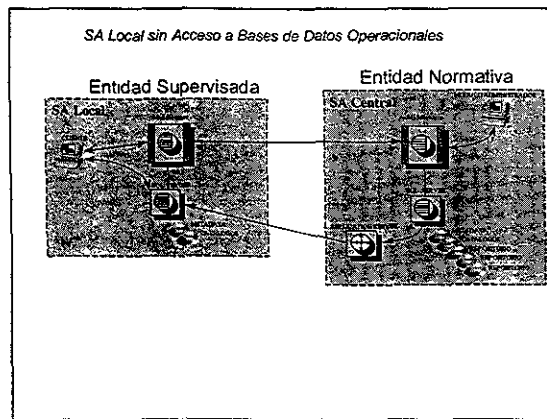


Figura 4.3 Representación de un SA sin acceder bases de datos operacionales de la Entidad Supervisada.

4.3.3 Envío de información directo a la Entidad Normativa utilizando el CLIENTE

En esta configuración la Entidad Supervisada solamente cuenta con el CLIENTE, el cual estará conectado directamente al SA Central; de esta forma la Entidad Supervisada puede enviar su documento a la Entidad Normativa vía la importación de un archivo, o bien captura manual. El *middleware* realizará el proceso de validación e integración al repositorio central.

En la figura 4.4 se puede apreciar este tipo de configuración en el cual el SA Central realiza todo el trabajo, carga, validación e integración de un documento.

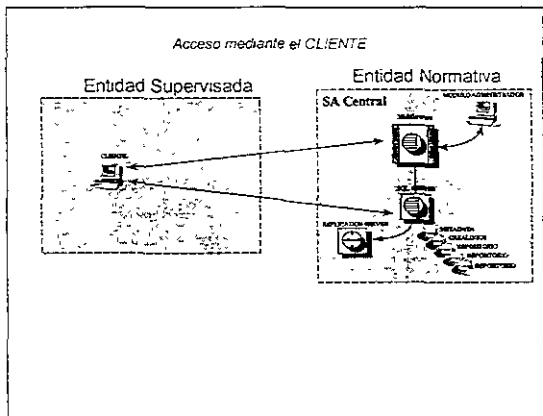


Figura 4.4 Representación de un SA con sólo el CLIENTE para envío de información.

4.3.4 Funcionalidad

Como se observó en el tema anterior, existen tres configuraciones que pueden utilizar las Entidades Supervisadas para enviar su información a la Entidad Normativa. A continuación se describen los procesos generales que realiza el SA.

- **Carga de información y validación de formato.** Las Entidades Supervisadas envían su información a la Entidad Normativa por medio del CLIENTE, el cual se encuentra conectado a un SA Local o Central. Al iniciar una sesión en el CLIENTE, la Entidad Supervisada deberá especificar, su clave de usuario y su password; esta información se validará en la metadata (utilizando ssa_validaUsuario) del SA Local o Central.

Una vez establecida una sesión, el CLIENTE obtendrá de la metadata (utilizando ssa_getDocumentos) los documentos que el usuario de la Entidad Supervisada puede enviar y la periodicidad con la que éstos deben procesarse.

El CLIENTE será capaz de cargar documentos mediante captura o importación de archivos, en cualquiera de los formatos previamente definidos en la metadata. Por medio del CLIENTE se indicará el tipo de documento que se desea enviar y el periodo (año y mes) para el cuál aplica, el CLIENTE validará el que usuario tenga permiso de transmitir, en esa fecha, el tipo de documento seleccionado.

El SA podrá, para algunas Entidades Supervisadas, extraer en forma automática y en la fecha adecuada, los documentos directamente de las bases de datos operacionales de la Entidad Supervisada; este procedimiento

es realizado por un SA Local (utilizando *rp_accesalInfo*) mediante un acceso remoto. Para obtener información mediante un acceso remoto (ya sea por petición o en forma automática), el SA obtendrá de la *metadata* (utilizando *ssa_getAccessInfo*) la siguiente información: el servidor de bases de datos a acceder, la base de datos a utilizar, *login* y *password* y el *stored procedure*, tabla o vista que permitirá obtener los documentos. Una vez obtenida esta información el SA establecerá una conexión al servidor de la Entidad Supervisada para poder obtener los documentos. En el caso de que los documentos no puedan obtenerse de esta forma, se regresará el error correspondiente a quien haya generado la petición (si se realizó una petición) o al responsable del envío (si se utilizó el mecanismo automático) y se registrará este hecho en la bitácora de la *metadata*.

Para el caso de recepción de información por captura, el CLIENTE genera una pantalla de captura con los campos que forman el documento y el formato que deberán cumplir cada uno de ellos, esta información la obtiene el CLIENTE de la *metadata* (utilizando *ssa_getFormato*). Una vez capturada la información, se podrá elegir la opción de respaldo en un archivo antes de enviar hacia el SQL Server.

El CLIENTE cargará la información (capturada o importada manual o automáticamente) en una tabla temporal. Esta tabla temporal tiene el mismo número de columnas que el documento cargado, sin embargo estas columnas son todas de tipo *varchar*. Se genera una tabla temporal por cada documento que cargue la Entidad Supervisada y se encuentra en la *metadata*. El realizar la carga en este tipo de tablas garantizará que todos los registros sean insertados aunque existan errores de formato. Después de haber realizado la carga del documento, el CLIENTE actualizará la bitácora para dicho documento (utilizando *ssa_creaBitacora* con un estatus carga correcta CO) y hará una petición al SA (utilizando *rp_procDocumento*) para que éste pueda continuar con los procesos de validación.

- **Validación de sensibilidad y congruencia.** Una vez que el SA (local o Central) recibe el aviso (*rp_procDocumento*) de que se ha cargado el documento; se determina de acuerdo al tipo de documento recibido, las validaciones de formato y las validaciones de sensibilidad y congruencia que se deberán aplicar al documento. Las reglas de validación y la secuencia en como éstas serán aplicadas se tomarán de la *metadata* (utilizando *ssa_getValidaciones*). Las validaciones más comunes son a nivel de registro, pero existen validaciones complejas que requieren de múltiples registros.

Las validaciones a nivel registro se aplicarán a todos los registros, es decir, el proceso de validación no terminará al encontrarse el primer error, esto permitirá generar un reporte de todos los errores encontrados en el documento, este reporte se regresará como acuse de recibo.

Existen validaciones simples y complejas las cuales se realizarán por medio de *stored procedures*, catálogos y fórmulas, dependiendo de la complejidad de estas.

Por medio del CLIENTE, se indica la realización de los procesos de validación e integración de la información de un documento, o bien, se puede indicar que se realice el proceso de validación exclusivamente, esto probablemente con el fin de validar la información antes de ejecutar el envío formal. En el caso de que se haga la petición de "sólo validación", el SA (Local o Central) realizará las validaciones con la diferencia que una vez terminadas, ya sea con error o sin él, no se generará una petición de integración.

Como ya habíamos mencionado un documento es cargado en tablas temporales, cuando se aplica el proceso de validación se genera una tabla histórica, la cual contiene la información ya validada.

- **Reenvíos por errores en los procesos de validación.** En el caso de que existan errores generados en el proceso de validación, estos errores se regresarán como reporte de errores; el cual podrá ser consultado utilizando el CLIENTE (*seguimiento del documento*). Por otra parte, se mantendrá guardada la información que haya sido cargada en la tabla temporal, para que el usuario, una vez que haya obtenido la respuesta de la validación fallida, pueda obtener de nuevo su documento (utilizando captura y modificación del documento) y así poderla presentar en la pantalla de captura para corregir los errores y realizar una nueva petición de validación.

Las tablas temporales que se generan cuando se realiza el proceso de carga son eliminadas por el SA (Local o Central) una vez que hayan pasado por el proceso de validación exitosamente.

- **Integración al repositorio central.** Este proceso es realizado únicamente por el SA Central, el cual se encuentra en la Entidad Normativa, ya que es ahí donde se podrá integrar la información para ser analizada por dicha Entidad.

Una vez validada la información, el SA Local (si se tiene uno en la Entidad Supervisada) enviará el HISTÓRICO generado al SA Central en la Entidad Normativa, este último se cargará al repositorio central. Para determinar la forma en cómo debe ser integrada la información se ejecutará *ssa_getMapeo* en la *metadata*. En el caso de que la integración a las bases de datos requiera de procesos más complejos, existirá la opción de integración mediante la ejecución de un procedimiento almacenado, en este caso el procedimiento almacenado contendrá el código necesario para realizar todo el proceso de integración. El nombre del procedimiento almacenado se obtendrá al momento de ejecutar *ssa_getMapeo*.

El SA Central se encargará de asegurar que la información se integre a las bases de datos de la Entidad Normativa, no importando que existan problemas de comunicación. Se registrará en la *metadata* del SA Local los documentos que estén pendientes por enviar (utilizando *ssa_envioPendiente*) para que el SA Local reintente el envío de información el número de veces que sea posible hasta lograrla.

- **Envío a múltiples SA Centrales.** Los documentos recibidos podrán ser enviados a múltiples SA Centrales, para determinar cuales son éstos, se ejecutará el procedimiento almacenado *ssa_getServidoresApp*. Una vez que se ha obtenido la información de los SA Centrales, se realiza el proceso de envío del HISTORICO y de la bitácora, controlando los posibles errores por falta de conexión a los SA Centrales.
- **Bitácora de estatus y acuse de recibo.** El *middleware* se encargará de registrar los procesos por los que pasa un documento y guardar el estatus de los mismos en la *metadata* del SA (Local y Central) utilizando *ssa_creaBitacora* y *ssa_setStatusEnvio*. Los mensajes de la bitácora (acuse de recibo o reporte de errores) serán enviados como resultado de una petición, o podrán ser consultados por medio del CLIENTE.

Para cada proceso aplicado a la información se generará un registro en la bitácora, de tal forma que se obtendrán múltiples mensajes de resultados en el acuse de recibo.

En el caso de acceso remoto a la información, el manejo del acuse de recibo podrá manejarse de dos formas: si el usuario responsable del documento enviado está conectado al *middleware*, se notificará de los procesos por los que ha pasado el envío y los resultados finales y si no existe ninguna conexión del usuario responsable, al momento que éste realice una sesión, se le notificará de todos los envíos para los cuales no recibió acuse de recibo. En cualquiera de los dos casos anteriores, se mantendrá la bitácora en la *metadata* del SA (Local y Central).

El acuse de recibo se generará una vez que se hayan aplicado los procesos de validación. En el acuse de recibo se indicará: la Entidad Supervisada, el usuario, el documento enviado, la fecha y los errores que pudiera contener el documento.

El reenvío de documentos se realiza cuando por algún motivo la integración es errónea, ya sea por que la información que la Entidad Supervisada no era la que tenía que enviar, pero sin embargo pasó el proceso de validación, o bien, por fallas de integración. En cualquiera de los dos casos el reenvío se realizará completamente y no en forma parcial. Esto implica que la Entidad Supervisada tenga que volver a cargar, validar e integrar su documento. El esquema de reenvíos también aplica para los accesos remotos, pero en caso de que se desee modificar la información, será responsabilidad de la Entidad Supervisada mantenerla en sus bases de datos, actualizarla y después indicar por medio del CLIENTE (*rp_accesaInfo*) que se necesita hacer un acceso remoto para nuevamente obtener la información.

- **Definición de *metadata*.** Por medio del MODULO ADMINISTRADOR se podrá definir en la base de datos de *metadata*:
 - Los documentos que se pueden recibir son:

Identificador del documento (Estados Financieros, Cómputos de Capitalización, etc.).

Para cada campo o dato en el documento: nombre, tipo, longitud, formato.

Periodicidad.

Reglas de validación y formatos asociados.

La secuencia en como deberán aplicarse las reglas de validación.

Procedimientos almacenados de validaciones complejas aplicados al documento.

La forma en como se integrará la información correspondiente a un documento, o el *stored procedure* que se deberá ejecutar para realizar la integración.

Los procedimientos almacenados que generarán la información en los acuses de recibo.

- Los documentos que un usuario en una institución puede transmitir.
- Los documentos que deben obtenerse en forma automática para una Entidad.
- Las Entidades Supervisadas para las cuales debe obtenerse en forma automática sus documentos, además de la información necesaria para el acceso remoto: servidor de bases de datos, base de datos, *login*, *password* y *stored procedure* a ejecutar.
- Los SA Centrales.
- Los servidores, base de datos y tablas en donde se integra la información.
- **Replicación de catálogos.** Para los procesos de validación, se utilizará información proveniente de la base de datos de catálogos de la Entidad Normativa, a esta información la denominaremos "catálogos".

La información de catálogos, así como la de la *metadata*, se mantendrá en sincronía entre la Entidad Normativa y las Entidades Supervisadas, por medio de un esquema de replicación de datos utilizando la tecnología de *Replication Server*.

La *metadata* será replicada pero no en su totalidad hacia las Entidades Supervisadas, ya que algunas tablas mantienen información asociada a la Entidad Supervisada (envío, bitácora, *estatus_envio_central*, etc.).

Una vez analizada la problemática del sistema financiero y revisados los conceptos principales procederemos en el siguiente capítulo con el diseño del *middleware* propuesto.

Capítulo 5

Diseño del Middleware

En este capítulo se verá el diseño de la funcionalidad del *middleware* y de sus procesos más relevantes como son la extracción, validación e integración de información.

5.1 Funcionalidad

Dentro de la arquitectura del DW, la función principal del *middleware* es la *extracción, validación e integración de información* desde las Entidades Supervisadas hacia la Entidad Normativa.

Internamente, el *middleware* se forma de varios *threads* y colas, nominalmente son:

Threads	Colas
• Notificador	• Del Notificador
• Conector	• Del Conector
• De Acceso	• De Acceso
• Validador	• De Validación
• Integrador	• De Integración

Cada vez que el *middleware* recibe una solicitud de envío de documento, todos estos *Threads* comienzan a interactuar entre sí por medio de las colas asignadas, el modo en que lo hacen se describirá posteriormente.

A continuación se describirá la funcionalidad de cada uno de los procesos más importantes del *middleware*.

5.1.1 Extracción de información

Analizaremos primero cómo llevar a cabo la extracción de información, en la descripción de la arquitectura global vimos que de acuerdo con la configuración que se defina para el SA, hay dos formas en que se puede obtener la información de las Entidades Supervisadas:

- Por medio del CLIENTE.
- Por medio de un acceso a base de datos operacionales.

Extracción por medio del CLIENTE

Dentro del *middleware* existe un *thread* llamado CONNECT_T y tiene asociada una cola llamada CONNECT_Q. Cuando se establece una conexión se coloca una petición en la cola de este *thread*, por medio de esta cola se obtendrán los resultados asociados a las peticiones generadas. En la figura 5.1 podemos ver la estructura funcional definida para el *middleware*, con los *threads* y *queues* internos (identificados por los números 1,2 y 3).

En el primer caso, el CLIENTE se encargará de la creación de una tabla temporal en la *metadata*, y de la inserción en la misma de la información del documento que se esté procesando (1). Esta tabla temporal se considerará parte de las bases de *metadata* y se creará una tabla temporal por cada documento que envíe la Entidad Supervisada.

Una vez que se ha cargado la información en la tabla temporal, el CLIENTE ejecutará un RPC (Remote Procedure Call, Procedimiento Remoto) *rp_procDocumento* (2) el cual procesa el documento previamente cargado y el CLIENTE quedará en espera de la respuesta. El CONNECT_T colocará una petición al *thread* de validación VALIDADOR_T utilizando la cola de mensajes VALIDA_Q que se encuentra asociada con este último en cada petición (3), para esto será necesario crear la estructura de datos PETICION, con los siguientes campos:

- Usuario
- Institución
- Identificación del documento
- Tipo de petición
- Tipo de servicio
- *Thread* asociado a la petición

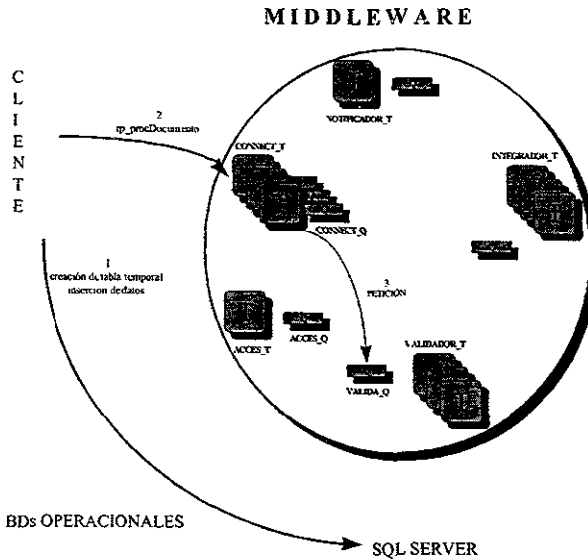


Figura 5.1 Threads y queues internos, configuración CLIENTE.

Una vez que el CONNECT_T coloca la petición pasará a un estado de "sleep", esperando en CONNECT_Q los mensajes de respuesta que se vayan generando como producto del proceso de la petición. El CONNECT_T despertará cuando reciba un mensaje en CONNECT_Q y regresará el control al CLIENTE para que éste pueda obtener los mensajes generados en la bitácora por medio de ssa_getBitacora.

Extracción por medio de acceso a bases de datos operacionales

La extracción de información mediante el acceso a las bases de datos operacionales que se encuentran en la Entidad Supervisada, se realiza mediante un proceso remoto. La Figura 5.2 muestra al *middleware* en la configuración para dicho proceso (indicado con los números del 1 al 7).

El OC timer estará configurado para generar "pulsos de reloj" cada N segundos (N es configurable) cada vez que se genere el "pulso de reloj", el OC timer ejecutará en el SQL Server el *stored procedure* ssa_getDoctosAccesoRemoto (1), el cual obtiene los documentos de una institución en la fecha correspondiente, para ello se basa en un cálculo del horario, los días de la semana y la fecha a partir de la cual se puede hacer la extracción. El *stored procedure* regresa los documentos para los cuales se puede hacer el acceso remoto. El OC timer generará una llamada al RPC rp_accesaInfo (2) para cada documento obtenido.

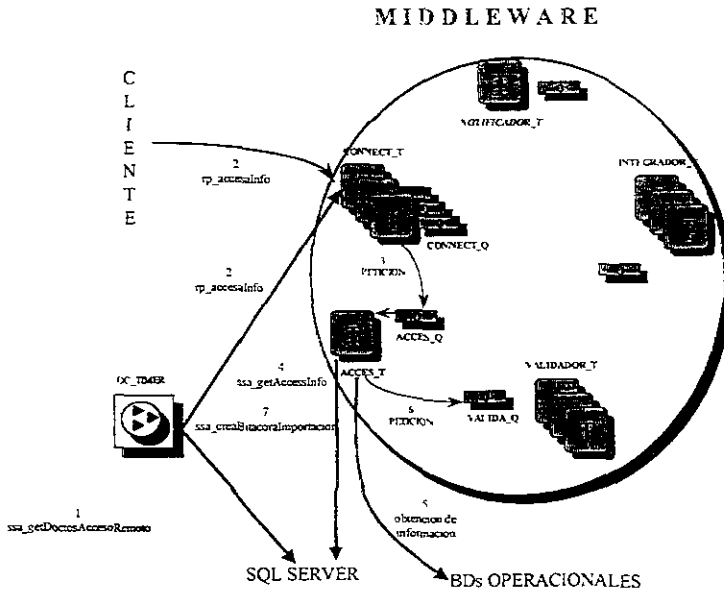


Figura 5.2 Extracción de información por acceso a las bases de datos operacionales.

Cuando el *middleware* recibe la llamada del RPC, esta petición se pasa al *thread* ACCESS_T (3), se ejecuta el *stored procedure* *ss_a_getAccesainfo* (4), el cual obtendrá de la *metadata* la información necesaria para realizar el acceso remoto en la Entidad Supervisada:

- El SQL Server de la base de datos que será accesada.
- La base de datos.
- Login y passwords.
- Tabla o tablas, vista ó *stored procedure* a ejecutar.

Una vez que se haya obtenido la información (5), se creará una tabla temporal que se asociará con el documento, y se generará una petición al VALIDADOR_T para que se apliquen los procesos normales de validación e integración (6) a la información del documento en cuestión. En el caso de que no se pueda realizar el acceso remoto a la información, el ACCESS_T generará un mensaje para la bitácora (7) utilizando *ss_a_creaBitacoraImportacion*, para que de esta forma el CLIENTE pueda informar correctamente sobre las causas por las que no se haya podido obtener la información.

5.1.2 Validación de sensibilidad y congruencia

El *middleware* contará con un número configurable de *threads* VALIDADOR_T. Estos *threads* compartirán la cola de mensajes VALIDADOR_Q de la cual obtendrán las

peticiones para realizar los servicios de validación correspondiente; las peticiones serán obtenidas por el *thread* que en ese momento esté desocupado. En la figura 5.3 se muestra este proceso (pasos 1,2,3 y 4).

Una vez que se ha obtenido la petición(1), el VALIDADOR_T deberá obtener las reglas de validación que deberá aplicar a la información previamente cargada en una tabla temporal.

El *middleware* contará con una lista doblemente ligada (CACHE_DOCTOS) de estructuras DOCTO_INFO en donde se mantendrán la información (incluyendo reglas de validación e integración) de los documentos más utilizados bajo una estrategia MRU/LRU (*MoreRecentUtilice/LesRecentUtilice*, Más Recientemente Utilizada / Menos Recientemente Utilizada). Esta lista tendrá asociada una tabla de *hash* para determinar en forma rápida si un documento se encuentra en esta lista.

Para obtener las reglas de validación el VALIDADOR_T buscará el documento en proceso en CACHE_DOCTOS(2), en caso de que no se encuentre, ejecutará *ssa_getValidaciones(2)* para obtenerlas de la *metadata*.

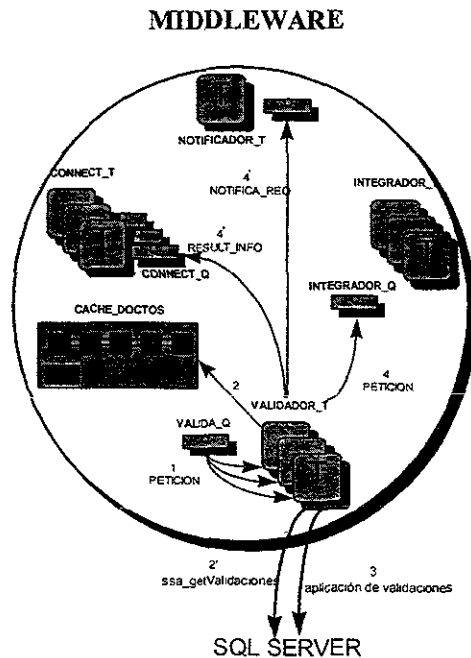


Figura 5.3. Validación de Información.

El VALIDADOR_T aplicará las reglas de validación(3) a nivel registro, (registro por registro). El proceso de validación se aplicará a todos los registros no importando que para algunos registros existan errores, de tal forma que se podrá generar un acuse de recibo con todos los errores que existan en la información.

El proceso de validación a nivel registro insertará en una tabla, a la que llamaremos histórico, la información que pasó los procesos de validación a nivel registro. Esta tabla tendrá una estructura (en cuanto a número de columnas y tipo de las mismas) igual al formato del documento en proceso.

Una vez cargada la información en histórico, el VALIDADOR_T aplicará las validaciones complejas mediante la ejecución de *stored procedures*. Si las validaciones complejas tienen éxito, la información en el histórico permanecerá, de otra forma se borrará.

Cuando la información haya sido validada en forma exitosa el VALIDADOR_T colocará una petición para integración al INTEGRADOR_T en la cola de mensajes INTEGRADOR_Q(4), siempre y cuando esté habilitada la opción de integración, en este último caso se regresará una respuesta exitosa al CONNECT_T y la información permanecerá en el HISTÓRICO para su posterior integración.

En el caso de que existan errores, el VALIDADOR_T colocará los mensajes de error en CONNECT_Q(4') para que el CONNECT_T avise al CLIENTE de los errores encontrados.

5.1.3 Integración de información

El *middleware* contará con un número configurable de *threads* INTEGRADOR_T. Estos *threads* compartirán la cola de mensajes INTEGRADOR_Q de la cual obtendrán las peticiones para realizar los servicios de integración correspondiente, las peticiones serán obtenidas por el *thread* que en ese momento esté desocupado. Dichos procesos se indican mediante los números del 1 al 5 de la figura 5.4.

La funcionalidad de los *threads* INTEGRADOR_T dependerá de que el *middleware* se encuentre en un SA Local o bien el SA Central.

Proceso de integración desde SA Local

Una vez que se ha obtenido la petición (1), como se muestra en la figura 5.4, el INTEGRADOR_T ejecutará el RPC *rp_integraPetición* (2) en el SA Central, además de que enviará la información a integrar almacenada en el HISTORICO.

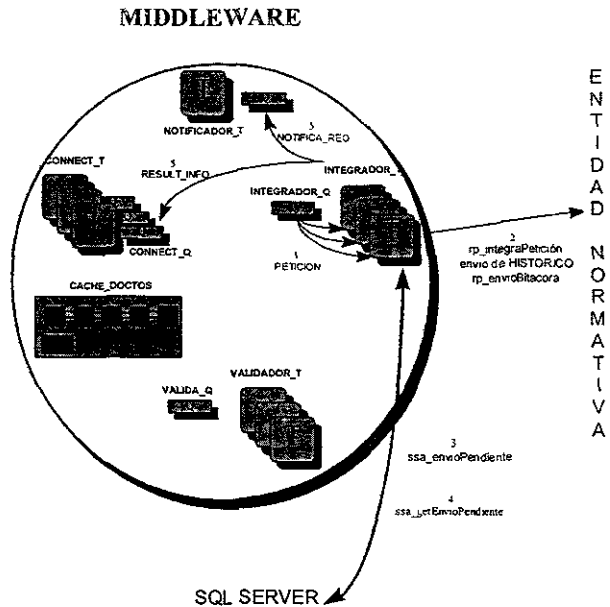


Figura 5.4 Proceso de Integración en un SA Local.

El INTEGRADOR_T del SA Local se encargará de enviar la información al SA Central que se encuentra en la Entidad Normativa. En el caso de que existan errores de comunicación que impidan el envío, el INTEGRADOR_T grabará en una bitácora en la base de datos de la *metadata* (usando *ssa_envioPendiente*) este hecho (3). A pesar de que no se pueda enviar la información a la Entidad Normativa, el INTEGRADOR_T del SA Local regresará un mensaje indicando que la información ha sido validada con éxito (estatus VO) y está en proceso de ser transferida a la Entidad Normativa para su integración al repositorio central (5).

El INTEGRADOR_T se encargará de estar revisando las conexiones hacia la Entidad Normativa, para que cuando no haya problemas de comunicación se puedan realizar los envíos que se encuentran pendientes (4), usando *ssa_getEnvioPendiente*.

El INTEGRADOR_T también recibirá peticiones para el envío de la bitácora. Para el caso de documentos sin errores, después de hacer el envío del HISTORICO, se enviará la bitácora del mismo. Para éste proceso el *middleware* ejecutará el RPC *rp_envioBitacora* (2) y enviará el contenido de la bitácora.

Este proceso de integración puede realizarse a un repositorio central o *datamarts* de una Entidad Normativa o a varios repositorios centrales o *datamarts* de otras Entidades Normativas, para determinar esto el INTEGRADOR_T ejecutará el *stored procedure ssa_getServidoresApp*.

Proceso de integración desde SA Central de la Entidad Normativa

Los procesos de integración de información que se realicen en el SA de la Entidad Normativa pueden deberse a:

- **Peticiones procesadas completamente en el SA Central.** Una vez que se ha obtenido la petición colocada por el VALIDADOR_T (1), el INTEGRADOR_T deberá obtener las reglas de integración que deberá aplicar a la información en HISTORICO, como se muestra en la figura 5.5.

Para obtener las reglas de integración el INTEGRADOR_T ejecutará *ssa_getMapeo* (2') en la base de datos *metadata*.

El INTEGRADOR_T integrará la información en el repositorio central o *datamarts* de la Entidad Normativa (3).

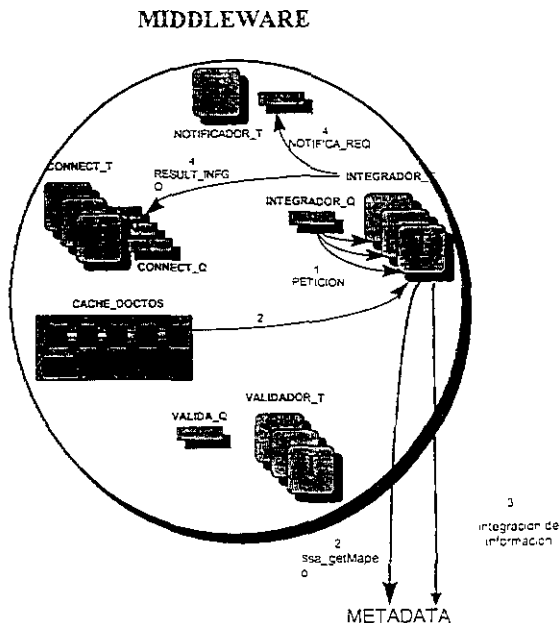


Figura 5.5 Proceso de Integración directo en el SA Central

Cuando la información haya sido integrada en forma exitosa (estatus IO) el INTEGRADOR_T regresará un mensaje de proceso exitoso(4) en CONNECT_Q. En el caso de que existan errores, el INTEGRADOR_T colocará los mensajes de error en CONNECT_Q para que el CONNECT_T avise al CLIENTE de los errores encontrados.

- **Peticiones provenientes de un SA Local.** El SA en la Entidad Normativa recibirá una llamada al RPC *rp_integraPetición* y enseguida recibirá la información del documento a integrar. Dicho proceso es ilustrado en la figura 5.6.

Una vez que el **CONNECT_T** haya cargado la información, éste realizará una petición al **INTEGRADOR_T** por medio de la cola de mensajes **INTEGRADOR_Q** (3).

Una vez que se ha recibido una petición el **INTEGRADOR_T** aplicará el mismo procedimiento de integración que el aplicado para las peticiones realizadas directamente en la Entidad Normativa (pasos 4-5).

El SA Central recibirá también de los SA Locales llamadas al RPC *rp_envioBitacora*, junto con la llamada al RPC se recibirá también la información generada en la bitácora, el *middleware* se encargará de insertar dicha información en la base de datos de *metadata* en la Entidad Normativa.

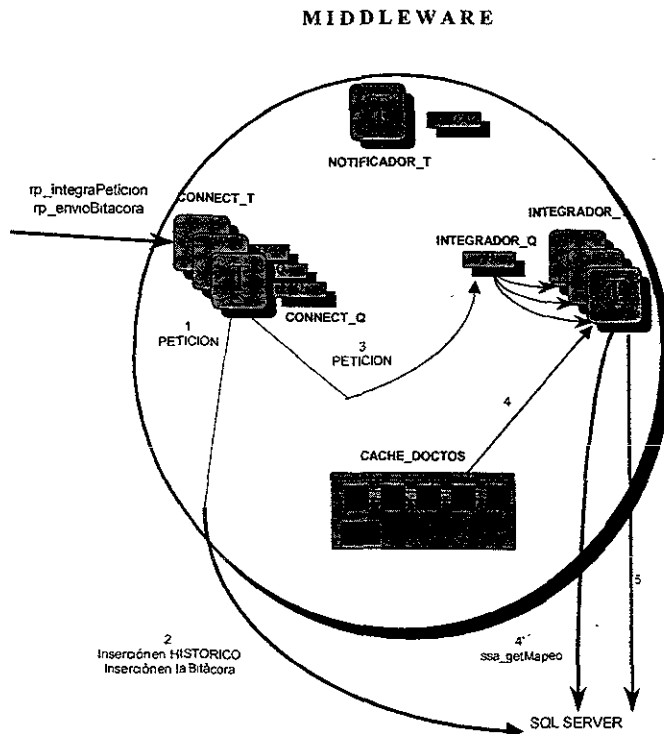


Figura 5.6 Proceso de Integración en el SA Central por peticiones de un SA Local.

Generación del acuse de recibo

Los usuarios del *middleware* obtendrán por cada documento enviado a la Entidad Normativa un acuse de recibo. El acuse de recibo, ya impreso, podrá ser generado a través del CLIENTE.

El que se obtenga un acuse de recibo, será exclusivo de las integraciones exitosas de documentos. El acuse de recibo estará formado por varios mensajes, los cuales indicarán la forma en que se procesó el documento en las diferentes etapas. Por medio del CLIENTE, se obtendrán los diferentes mensajes de los que esté formado el acuse de recibo. Por otra parte, se podrá consultar cada uno de los acuses de recibo generados para un usuario.

El CLIENTE recibirá la respuesta a la petición generada, después de esto podrá acceder a la bitácora para obtener cada uno de los mensajes del acuse de recibo. Para el caso de documentos procesados en forma exitosa, cuyo proceso se haya generado en un SA Local, pero que la integración se haya dado en la Entidad Normativa, el último mensaje del acuse de recibo y la notificación al usuario se realizará como a continuación se describe.

Una vez que se ha integrado la información en la Entidad Normativa, el *middleware* generará un mensaje a la bitácora en la base de datos de *metadata* (1).

Cuando se reciba la replicación de los mensajes de bitácora, como se ve en la figura 5.7, el *SQL Server* ejecutará un RPC(*rp_bitacoraMsg*) al SA Local (2), para que éste internamente genere una petición en el NOTIFICADOR_T(3).

Actualización de *metadata* e información de administración

El *middleware* utilizará una serie de estructuras de datos con información cargada de la base de datos *metadata*. Cuando la información en estas bases de datos sea actualizada, los cambios tendrán que reflejarse en dichas estructuras de datos, para que de esta forma pueda el *middleware* trabajar con información actualizada.

Cuando la información en las bases de datos sea actualizada, como se muestra en la figura 5.7, se utiliza el esquema de replicación para actualizar las bases de datos de *metadata* de los SA Locales de las Entidades Supervisadas.

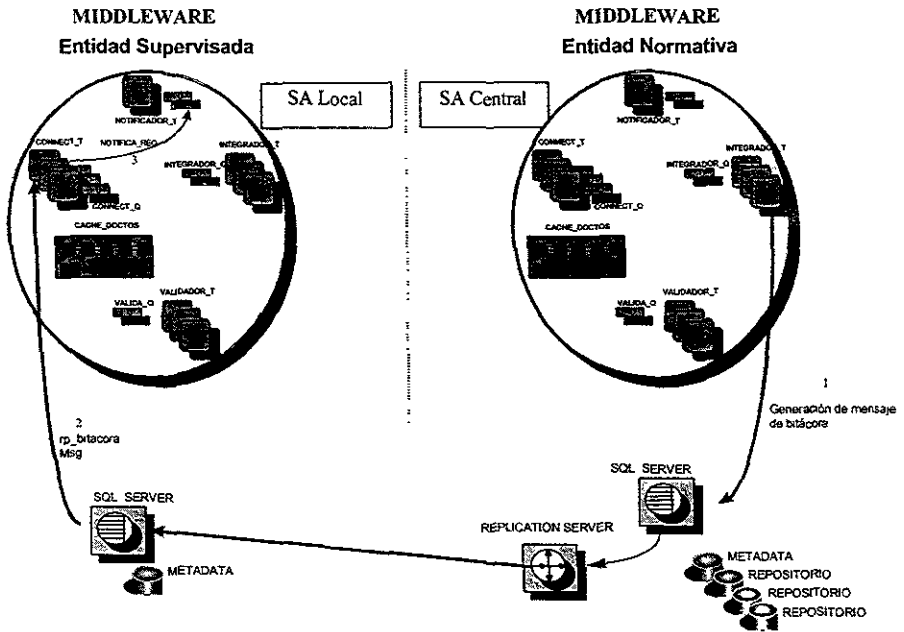


Figura 5.7 Generación del acuse de recibo final por peticiones de un SA Local.

Estatus de las peticiones

La información enviada al *middleware* y que se trata como un envío puede pasar por los estatus que a continuación se describen:

Estatus	Significado	Descripción
c	Carga	Se está cargando información a la tabla temporal. Esto implica que se pueden realizar varias cargas de la información de un documento.
co	Carga OK	La información ha sido cargada completamente a la tabla temporal, pero todavía no se indica al <i>middleware</i> para que éste la comience a procesar.
ce	Carga enviada	La información ha sido cargada a la tabla temporal y ya se ha avisado al <i>middleware</i> para que éste la procese. Este estatus es útil para el procedimiento de recuperación del <i>middleware</i> .
v	Validación	El <i>middleware</i> ha recibido una petición para procesar la información y está en el proceso de validarla.
vo	Validación OK	La información recibida se ha terminado de validar y el resultado ha sido exitoso.
vf	Validación <i>FAIL</i>	La información recibida tiene errores, sin embargo se mantiene en la tabla temporal para que pueda ser leída nuevamente.
i	Integración	La información recibida necesita ser integrada. Para el caso de SA Local, este estatus indica que la información tiene que mandarse al SA Central.
ie	Integración ENVIO	Para el caso de SA Local, este estatus indica que la información ha sido mandada al SA Central.
io	Integración OK	El proceso de integración para la información se realizó en forma exitosa.
if	Integración <i>FAIL</i>	Existieron problemas al integrar la información.
sv	Sólo validación	La integración está deshabilitada.

Estatus de la bitácora

La bitácora asociada a un envío puede pasar por los estatus que a continuación se describen.

Estatus	Significado	Descripción
C	Creación	La bitácora ha sido creada
PD	Pendiente	La bitácora del envío está pendiente de enviar al SA en la Entidad Normativa
E	Envío OK	La bitácora ha sido enviada a la Entidad Normativa

5.2 Definición de Estructuras de Datos

Una estructura de datos es un conjunto de elementos de diferentes tipos de datos. Las estructuras de datos son utilizadas para mejorar el funcionamiento de un sistema.

En el *middleware* se definen varias estructuras de datos las cuales nos ayudarán a tener un mejor manejo de la información.

CACHE

Estructura de datos en donde se mantiene una lista ligada de los documentos más frecuentemente utilizados. El reemplazo en la lista se realiza utilizando una estrategia de MRU/LRU. Además de la lista ligada existe una tabla de *hash* para poder determinar en forma rápida si un documento se encuentra en la lista.

DOCTO_LIST	documentos	Lista de documentos
HASH_TABLA	tabla	Tabla de <i>Hash</i>

COLUMN_LIST

Lista de columnas de un documento enviado por la Entidad Supervisada

CS_INT	numColumnas	Número de columnas.
FTO_COLUMNNA	*columna[MAX_COLUMNS]	Arreglo de columnas.

DOCTO_INFO

En esta estructura de datos se mantiene la información de un documento, incluyendo reglas de validación e integración.

CS_INT	idDocto	Identificador del documento.
CS_SMALLINT	version	Versión del documento.
CS_CHAR	periodicidad	Periodicidad para enviar el documento: A Anual, 6 Semestral B Bimestral, M Mensual S Semanal, D Diaria
COLUMN_LIST	columnas	Lista de columnas.
REGLAS_INTEGRACION_LIST	reglasIntegracion	Reglas de integración.
VALIDACION	validacion	Reglas <i>stored procedures</i> de validación

CS_INT	reglasCargadas	Bitmask para indicar si la estructura tiene cargadas las reglas de validación y de integración: REGLAS_VALIDACION REGLAS_INTEGRACION
CS_CHAR	tipo	Tipo de documento: DOCUMENTO_MATRICIAL o DOCUMENTO_TABULAR
struct DOCTO_INFO	*siguiente	Apuntador para el manejo de la lista.

DOCTO_LIST

La estructura DOCTO_LIST es una lista de documentos que envían las Entidades Supervisadas a la Entidad Normativa.

DOCTO_INFO	*primero	Apuntador al elemento que se encuentra en el LRU y que será el siguiente elemento en ser reemplazado.
DOCTO_INFO	*ultimo	Apuntador al elemento que se encuentra en el MRU y que es en donde se insertará el siguiente elemento.
CS_INT	numDocumentos	Número de elementos en la lista.

RENGLON

Renglón restituído por un comando al *SQL Server*.

CS_INT	numColumns	Identifica el número de columnas que componen el renglón.
CS_VOID	**rowBuffer	Datos del renglón.
CS_SMALLINT	*indicator	Indicador
CS_DATAFMT	*rowDesc	Descripción de la columna
CS_INT	colSize	Ancho de la columna

MAPEO_COLUMNNA

Estructura de datos que mantiene el mapeo de una columna del documento recibido a una columna en una tabla.

CS_INT	numColumna	Número de columna en el documento recibido
--------	------------	--

CS_CHAR	*formulaIntegracion	Fórmula para obtener la información que se tiene que integrar.
CS_INT	llave	Indica si la columna es parte de la llave para la tabla operacional.
CS_CHAR	*columna	Columna donde se integrara la información.
FORMATO_COLUMNA	*formatoColumna	Contiene información de una columna de un documento.
CS_CHAR	*nombre	Nombre de la columna.
CS_CHAR	*tipo	Tipo de la columna.
CS_INT	longitud	Lóngitud de la columna.
CS_INT	decimales	Número de decimales si el tipo es <i>numeric decimal</i> .

MONITOR_INFO

Estructura de datos que mantiene estadísticas del *middleware*.

CS_INT	conexionesActivas	Número de conexiones que se encuentran activas en el <i>middleware</i> .
CS_INT	conexionesTotales	Número de conexiones totales que se han recibido.
CS_LONG	memAlloc	Cantidad de memoria dinámica alojada.
CS_LONG	memFree	Cantidad de memoria liberada.
CS_INT	peticionesExitosas	Número de peticiones procesadas exitosamente.
CS_INT	peticionesFallidas	Número de peticiones que generaron errores.
CS_INT	peticionesPorIntegrar	Número de peticiones pendientes de integrar.
CS_INT	peticionesPorValidar	Número de peticiones pendientes de validar.
CS_INT	accesosRemotosExitosos	Número de documentos obtenidos por acceso remoto
CS_INT	accesosRemotosFallidos	Número de intentos fallidos de acceso remoto.
CS_INT	notificacionesDeRetraso	Número de notificaciones por retraso en envío de documentos.

NOTIFICA_REQ

La estructura NOTIFICA_REQ se utiliza para solicitar envío de notificaciones al NOTIFICA_T.

CS_INT	tipo	Tipo de notificación: DOCTO_NO_ENVIADO, RESULTADO_PETICION.
CS_CHAR	*usuario	Usuario al que se deberá enviar la notificación.
CS_CHAR	*institucion	Institución a la que pertenece el usuario.
CS_CHAR	*dato1	Parámetro a enviar en la notificación.
CS_CHAR	*dato2	Parámetro a enviar en la notificación.
CS_CHAR	*dato3	Parámetro a enviar en la notificación.

PETICION

Estructura de datos utilizada para almacenar información referente a las peticiones de envío de documentos generadas al *middleware*.

CS_CHAR	*usuario	Usuario que generó la petición.
CS_INT	folio	Folio de la petición.
CS_CHAR	*institucion	Institución a la que pertenece el usuario. (nombre corto).
CS_INT	cveInstitucion	Clave de la institución.
CS_INT	cveSector	Sector de la institución.
CS_INT	areald	Identificador del área en la institución.
CS_CHAR	status[3]	Estatus de la petición: C,V,VO,VF,I,IE,IO,IF.
CS_INT	tipo	<i>Bitmask:</i> PETICION_CONECTADA, PETICION_SINCRONA, PETICION_CAPTURADA, PETICION_ACCESO_REM, PETICION_IMPORTACION.
CS_CHAR	tipoReenvio	N No aplica, R insertar rengiones nuevos, T reemplazo total.
CS_CHAR	*fecha	Fecha en la que se recibe el envío.
CS_SMALLINT	ano	Año para el que aplica el envío.
CS_SMALLINT	periodo	Periodo para el que aplica el envío.
CS_CHAR	servicio	Tipo de servicio solicitado: VALIDACION, INTEGRACION.
THREAD_INFO	*thread	<i>Thread</i> por el que se recibió la petición.
DOCTO_INFO	*documento	Documento asociado a la petición.
CS_CHAR	*tablaTemporal	
CS_INT	idCentral	
CS_CHAR	*saName	Nombre del SA Central.
CS_CHAR	*saLogin	<i>Login</i> para el servidor central.
CS_CHAR	*saPassword	<i>Password</i> para el servidor central.
CS_CHAR	*sqlName	Nombre del Sql server central.

CS_CHAR	*sqlLogin	<i>Login</i> para el Sql server central.
CS_CHAR	*sqlPassword	<i>Password</i> para el Sql server central.
CS_VOID	*dato1	Campo de propósito general.
CS_VOID	*dato2	Campo de propósito general.

QUEUE_INFO

Estructura de datos con información de una cola de mensajes.

CS_CHAR	*cola	Nombre de la cola.
CS_INT	mensajesEscritos	Número de mensajes colocados en la cola.
CS_INT	mensajesLeidos	Número de mensajes leídos de la cola.
CS_VOID	*listaPeticiones	Lista de mensajes en la cola.
struct queue_info	*siguiente	Apuntador para el manejo de la lista.

QUEUE_LIST

Lista con información del estado de las colas de mensajes

QUEUE_INFO	*primero	Primer elemento en la lista.
QUEUE_INFO	*ultimo	Ultimo elemento en la lista.
CS_INT	numColas	Número de colas.

REGLA_LIST

Es una estructura que contiene las reglas de integración asociadas a un documento.

CS_INT	numReglas	Número de reglas de integración.
REGLA_INTEGRACION	*reglas	Lista de reglas de integración.

REGLA_INTEGRACION

Estructura de datos que mantiene la información de una tabla en donde se integrará la información.

CS_CHAR	*tabla	Tabla en donde se integrara la información.
CS_CHAR	*bd	Base de datos en donde reside la tabla.
CS_CHAR	*server	Servidor en donde reside la base de datos.
CS_CHAR	*login	<i>Login</i> con el que se tiene acceso al servidor.

CS_CHAR	*password	Password asociado al login de acceso al servidor.
MAPEO_COLUMNNA	region[<i>MAX_COLUMNS</i>]	Mapeo de columnas.
CS_INT	numColumnas	Número de columnas.

RESULT_INFO

Estructura de datos que contiene los mensajes de respuesta generados por las peticiones y que deben ser enviados al cliente que genera la petición.

CS_INT	tipoMsg	ERROR_MSG, FINAL_MSG, WARNING.
CS_CHAR	*msg	Mensaje de resultado.
CS_INT	numError	Número del error generado.
CS_INT	severity	Nivel de severidad del error.

SISTEMA_INFO

La estructura SISTEMA_INFO mantiene información general del *middleware*.

CS_CONTEXT	*cntx_ptr	Apuntador a la estructura CS_CONTEXT.
CS_CHAR	*login	Login de administrador del SA.
CS_CHAR	*password	Password asociado al login del administrador del SA
CS_BOOL	shutdown	Bandera que indica si se ha solicitado <i>shutdown</i> .
CS_INT	tipoSistema	Tipo de SERVIDOR APLICATIVO: SERVER_LOCAL, SERVER_CENTRAL.
CS_CHAR	*openServerName	Nombre del SA.
CS_CHAR	*institucion	Nombre de la institución donde reside el SA.
CS_CHAR	*configFile	Nombre del archivo de configuración.
CS_CHAR	*errorLog	Nombre del archivo de errores.
CS_CHAR	*sqlServerName	Nombre del SQL SERVER.
CS_INT	traceLevel;	Bitmask para el rastreo de mensajes: TRC_EVERYTHING, TRC_FUNCTION_ENTER_RETURN, TRC_FUNCTION_PARAMETERS, TRC_FUNCTION_RESULTS, TRC_THREAD_ERRORS, TRC_FATAL_ERRORS
CS_CHAR	*saEN	SERVIDOR APLICATIVO en la Entidad Normativa.
CS_CHAR	*sqlEN	SQL Server en la Entidad Normativa.
CS_CHAR	*validadorLogin	Login que utiliza el VALIDADOR_T.

CS_CHAR	*validadorPwd	Password del Login del VALIDADOR_T.
CS_CHAR	*integradorLogin	Login del INTEGRADOR_T.
CS_CHAR	*integradorPwd	Password del INTEGRADOR_T.
CS_CHAR	*accessLogin	Login del ACCESS_T.
CS_CHAR	*accessPwd	Password del ACCESS_T.
CS_CHAR	*bulkLogin	Login del BULK_T.
CS_CHAR	*bulkPwd	Password del BULK_T.
CS_CHAR	*mntLogin	Login del MNT_T.
CS_CHAR	*mntPwd	Password del MNT_T.
CS_SMALLINT	numIntegradorT	Número de INTEGRADOR_T configurados.
CS_SMALLINT	numValidadorT	Número de VALIDADOR_T configurados.
CS_INT	maxNumConnections	Número máximo de conexiones del <i>middleware</i> .
CS_SMALLINT	tamCache	Número máximo de documentos en la memoria Cache.

THREAD_INFO

En esta estructura de datos se mantiene la información de los *threads* de servicio y *threads* generados por conexiones.

SRV_PROC	threadId	Identificador del thread.
CS_SMALLINT	tipo	Tipo de <i>thread</i> : USER_THREAD, VALIDADOR_T, INTEGRADOR_T, ACCESS_T, NOTIFICADOR_T, ADMON_THREAD, MNT_T.
CS_CHAR	hostName	Nodo desde el que se genera la conexión.
CS_CHAR	institucion	Institución asociada a la conexión para el caso de CONNECT_T, para el caso de los <i>threads</i> de servicio es la institución asociada a la petición que se esta procesando.
CS_SMALLINT	estatus	Estatus del thread: WAITING_CMD, PROCESS_REQ, SLEEP_WAITING_MSG.
CS_INT	peticionesExitosas	Número de peticiones procesadas en forma exitosa.
CS_INT	peticionesFallidas	Número de peticiones fallidas.
CS_INT	peticionesAsincronas	Número de peticiones asíncronas recibidas.
SRV_OBJID	msgQId	Id de la cola de mensajes asociada al <i>thread</i> .

CS_CHAR	*doctoEnProceso	Para los <i>threads</i> de servicio, (documento procesándose).
CS_CHAR	*usuario	Usuario asociado al documento en proceso.
CS_INT	folio	Folio de la petición procesándose para los <i>threads</i> .
CS_INT	cveInstitucion	Clave de la institución que envía la petición.
CS_INT	cveSector	Clave del sector asociada a la institución.
CS_VOID	*dato1	Campo de propósito general.
CS_VOID	*dato2	Campo de propósito general.
CS_VOID	*dato3	Campo de propósito general.
struct thread_info	*siguiente	Apuntador para el manejo de la lista.

THREAD_LIST

La Estructura THREAD_LIST es una lista de *threads*

THREAD_INFO	*primero	Primer elemento en la lista
THREAD_INFO	*ultimo	Último elemento en la lista
CS_INT	numThreads;	Número de <i>threads</i> .

USER_INFO

Estructura de datos en donde se mantiene la información de un usuario válido en el *middleware*.

CS_CHAR	*user	Usuario válido en el SSA
CS_CHAR	*password	Password del usuario
CS_CHAR	*institucion;	Institución a la cual pertenece el usuario

USER_LIST

La estructura USER_LIST es una lista de usuarios válidos

USER_INFO	users[MAX_NUM_USERS]	Lista de usuarios.
CS_INT	numUsers	Número de usuarios.

VALIDACIÓN

Estructura que mantiene la información de reglas y stored procedures de validación.

VALIDACION_CAT_LIST	valCatList	Lista de validaciones contra catálogos.
VALIDACION_EXP_LIST	valExpList	Lista de expresiones de validación.
SP_VALIDACION_LIST	spsValidacion	Lista de <i>stored procedures</i> de validación.

VALIDACION_CAT

Validación contra catálogo, que incluye las columnas que se deben validar

CS_INT	valCatId	Identificador del catalogo.
CS_INT	numCol	Número de columnas involucradas en la validación.
CS_CHAR	*catalogo	Nombre del catálogo contra el que se hace la validación.
MAPEO_COLUMNA	mapeo[MAX_COLUMNS]	Validación columna contra columna.

VALIDACION_CAT_LIST

Lista de validaciones de columnas del documento contra columnas de las tablas catálogo.

CS_INT	numVal	Número de validaciones.
VALIDACION_CAT	validacion[MAX_NUM_VALIDACIONES]	Arreglo de validaciones.

VALIDACION_SP

Lista de *stored procedures* de validación.

CS_INT	numSP	Número de <i>stored procedures</i> .
CS_CHAR	*storedP[MAX_SPS]	<i>Stored Procedures</i> de validación.

VALIDACION_EXP_LIST

Lista de expresiones de validación

CS_INT	numExp	Número de expresiones de validación.
VALIDACION_EXP	*expresiones [MAX_NUM_VALIDACIONES]	Arreglo de expresiones de validación.

VALIDACION_EXP

Estructura que representa una expresión de validación, la cuál puede ser una expresión aritmética/lógica sencilla o una expresión IF-THEN-ELSE, en este último caso la expresión se convierte en un árbol.

CS_INT	exprId	Identificador de la expresión.
CS_CHAR	*expresion	Expresión aritmética/lógica.
struct valExp	*thenExp	Apuntador a la expresión <i>then</i> .

struct valExp	*elseExp	Apuntador a la expresión <i>else</i> .
CS_CHAR	*errorTexto	Mensaje de error asociado a la expresión.
CS_CHAR	*errorNum	Número del mensaje de error asociado a la expresión.
CS_CHAR	Nivel{2}	Nivel de validación: R renglón, M múltiples renglones, T tabla.

ACCESS_INFO

La estructura se utiliza para mantener la información necesaria para establecer un acceso remoto a un servidor de bases de datos y obtener la información de un documento.

CS_CHAR	*objeto	Nombre del objeto a acceder.
CS_CHAR	*tipoObjeto	Tipo del objeto: P (stored procedure) T (tabla) V (vista)
CS_CHAR	*serverName	Parámetro a enviar en la notificación.
CS_CHAR	*bd	Parámetro a enviar en la notificación.
CS_CHAR	*login	Parámetro a enviar en la notificación.
CS_CHAR	*pwd	Parámetro a enviar en la notificación.

5.3 Definición de Eventos y Callbacks

Un *Event Handler* es un proceso que se ejecuta al iniciarse un evento. Cuando un evento se dispara el *Open Server* coloca el evento y el *thread* activo en la cola actual. En ese momento, el *thread* ejecuta una rutina que procesa el evento. Esta rutina se llama *Event Handler*.

Existen *event handlers* predefinidos y los definidos por el usuario. De los predeterminados tenemos los siguientes:

AttentionHndl	<i>Handler</i> para el evento SRV_ATTENTION.
BulkHndl	<i>Handler</i> para el evento SRV_BULK.
ConnectHndl	<i>Handler</i> para el evento SRV_CONNECT.
DisconnectHndl	<i>Handler</i> para el evento SRV_DISCONNECT.
LanguageHndl	<i>Handler</i> para el evento SRV_LANGUAGE.
ShutdownHndl	<i>Handler</i> para el evento SHUTDOWN.
StartHndl	<i>Handler</i> para el evento SRV_START.
StopHndl	<i>Handler</i> para el evento SRV_STOP.

5.4 Definición de *Threads* Internos

Presentamos la lista de los *threads* internos con sus correspondientes descripciones.

ACCESS_T	Obtiene los mensajes de su cola de mensajes (ACCESS_Q). Estos mensajes son peticiones de acceso remoto, las cuales lleva a cabo.
BULK_T	<i>Thread</i> que solamente existe en el SA Central. Se encarga de recibir el HISTÓRICO y la bitácora de las peticiones procesadas localmente y cargarlos a las bases de datos en la Entidad Normativa. La recepción y carga las realiza con <i>bulk copy</i> .
INTEGRADOR_T	Recibe peticiones en su cola de mensajes (INTEGRADOR_Q) para realizar los servicios de integración. En el caso de un SA Local, este <i>thread</i> se encarga de enviar el HISTÓRICO y la bitácora asociada a una petición al SA Central. En el caso del SA Central, este <i>thread</i> se encarga de realizar la integración del HISTÓRICO a las bases de datos operacionales.
NOTIFICADOR_T	Recibe peticiones para realizar notificaciones. Puede notificar el término del proceso de una petición o bien avisar del retraso en el envío de información.
UPD_DATA_T	<i>Thread</i> en donde se centraliza la actualización de estructuras de datos que se mantienen en memoria. Recibe peticiones de cambio a las diversas estructuras de datos por medio de una cola de mensajes. Estas peticiones son provocadas en la mayoría de los casos por cambios en las bases de datos. Debido a este mecanismo el <i>middleware</i> aplica los cambios cuando la base de datos de <i>metadata</i> cambia.
VALIDADOR_T	<i>Thread</i> que se encarga de aplicar las reglas de validación asociadas a un documento.

5.5 Definición de Procedimientos Registrados

Los Procedimientos Registrados son pequeñas funciones que se definen en el *start Handler* para que puedan ser utilizadas desde el *middleware* o el *SQL Server*

A continuación se presenta la lista de procedimientos registrados incluyendo cada uno de sus parámetros y los resultados que se deben obtener al aplicar cada uno de estos.

rp_activaPetición

Procedimiento registrado para activar la integración de la información de un documento.

PARÁMETROS:

EnvFolio

Número de folio del documento a enviar.

InsClave

Clave de la Entidad Supervisada.

InsSector

Número de sector.

RESULTADO:

Nada

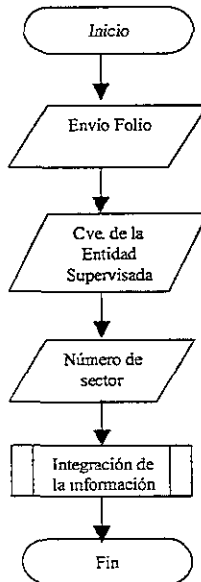


Figura 5.8 Diagrama para activar la integración de la información de un documento.

rp_accesalInfo

Procedimiento registrado para indicar al *middleware* que debe obtener, mediante acceso remoto, la información de un documento.

PARÁMETROS:

DocId	Identificador de Documento
InsClave	Identificador de la Institución
InsSector	Identificador del sector de la Institución
ServerRemoto	En el caso de peticiones generadas desde la Entidad Normativa, el <i>middleware</i> que realizará el acceso remoto.

RESULTADO:

env_folio	Número de folio de la petición generada.
-----------	--

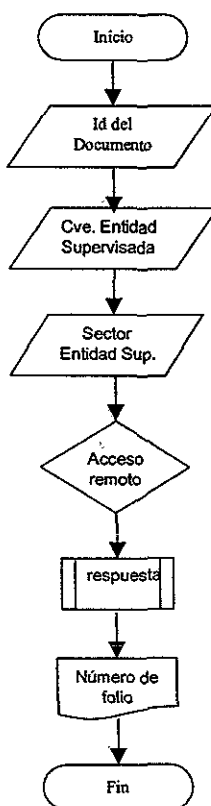


Figura 5.9 Diagrama para acceder a un documento de forma remota.

rp_bitacoraMsg

Procedimiento Registrado con el cuál el SQL Server avisa para indicar al *middleware* que se recibió mediante replicación el mensaje de que una petición ha sido integrada por el *middleware* en la Entidad Normativa.

PARÁMETROS:

envFolio Número de folio de la petición.

insClaveClave de la Entidad Supervisada

insSectoClave del Sector de la Entidad Supervisada

RESULTADO:

Ninguno

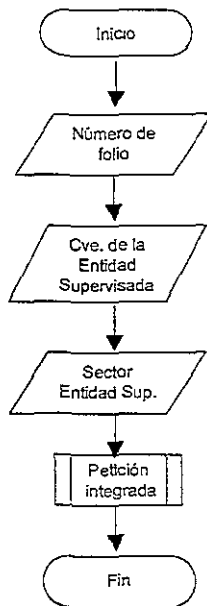


Figura 5.10 Diagrama de aviso de la Entidad Normativa a la Entidad Supervisada.

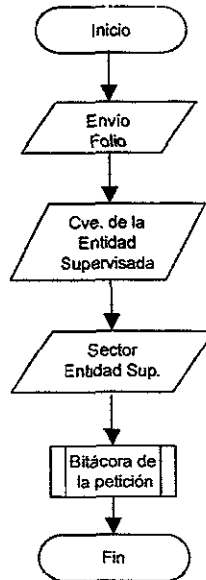


Figura 5.11 Diagrama de un mensaje de la Entidad Normativa a la Entidad Supervisada.

rp_integraPetición

Procedimiento registrado con el que un SA Local pide al SA Central la integración de una petición generada localmente y ya validada.

PARÁMETROS:

EnvFolio
InsSector
InsClave
Usuid
DocId
DocVerVersion
EnvFecha
EnvAño
EnvPeriodo
EnvTipoRespuesta

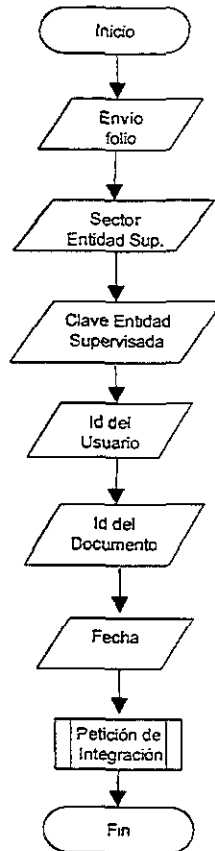


Figura 5.12 Diagrama de petición de integración a la Entidad Normativa.

rp_procDocumento

Procedimiento registrado con el que el CLIENTE pide que se procese un documento previamente cargado.

PARÁMETROS:

EnvFolio		Número de folio de la petición que se va a procesar. (numeric(9,0))
InsClave		int
InsSector		smallint
TipoServicio	char(1)	Integración, validación.

RESULTADO:

Ninguno

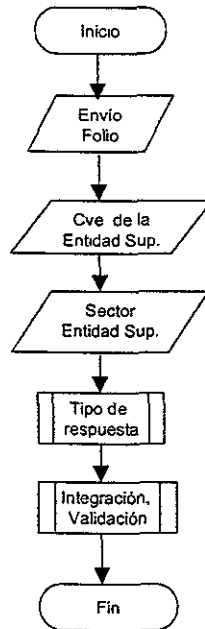


Figura 5.13 Diagrama de petición de procesamiento un documento ya cargado.

5.6 Definición de Stored Procedures

Aquí presentaremos los *stored procedures* que se ejecutan de acuerdo a cada etapa de los procesos donde intervienen.

ssa_acuseRecibo

Este procedimiento nos ayuda a obtener toda la información necesaria para la generación y emisión del acuse de recibo, cuando un documento ha sido enviado a través del CLIENTE.

PARÁMETROS

Sector
Clave
EnvFolio

ssa_bitacoraPendiente

Procedimiento que marca el envío como pendiente de mandar en la bitácora.

PARÁMETROS

EnvFolio
InsClave
InsSector
SerAppld

RESULTADOS

Ninguno

ssa_borraBitacora

Procedimiento que borra la bitácora de envío asociada a un documento, para estar limpia en espera de la llegada de otro documento.

PARÁMETROS

EnvFolio
InsClave
InsSector
BitFase

ssa_creaBitacora

Este procedimiento registra un mensaje en la bitácora por cada documento enviado por parte del *middleware*.

PARÁMETROS

EnvFolio
InsClave
InsSector
BitFase
BitMensaje
BitNumRegistro
BitNumColumna
MenNumero

RESULTADOS

Ninguno

ssa_creaBitacoraImportacion

Procedimiento que inserta un registro en la bitácora por cada uno de los procesos de importación automática de archivos y de acceso a bases de datos.

PARÁMETROS

InsSector
InsClave
Arelid

BitMensaje
DocId
DocAnio
DocPeriodo
BitTipo

ssa_creaEnvio

Este procedimiento registra en la base de datos cada envío de documentos que se realiza.

PARÁMETROS

InsSector
InsClave
Usuld
DocId
EnvAño
EnvPeriodo
EnvTipoCarga
EnvTablaTemporal
EnvStatus

RESULTADOS

env_folio

ssa_creaEnvioCentral

Este procedimiento da de alta un envío en el SQL Server del SA Central, en este caso la diferencia con dar de alta en el sistema local, es que aquí ya no se genera el número de folio, ya que se está registrando la información.

PARÁMETROS

EnvFolio
InsSector
InsClave
Usuld
DocId
DocVerVersion
EnvFecha
EnvAño
EnvPeriodo
EnvTipoRespuesta
EnvTipoCarga

ssa_creaEnvioPorAccesoBD

Procedimiento que registra un envío cuya información ha sido obtenida por el esquema de acceso remoto a bases de datos.

PARÁMETROS

InsSector
InsClave
Areld
DocId
EnvAño
EnvPeriodo
EnvTablaTemporal

ssa_envioPendiente

Procedimiento que marca el envío como pendiente en la *metadata*

PARÁMETROS

EnvFolio
InsClave
InsSector

RESULTADOS

Ninguno

ssa_formateaPeriodo

Procedimiento interno que realiza operaciones para la presentación adecuada de información del CLIENTE.

PARÁMETROS

Ninguno

RESULTADOS

Ninguno

ssa_getAccesInfo

Obtiene la información de acceso necesaria para obtener, mediante acceso remoto, la información de un documento para una Entidad Supervisada.

PARÁMETROS

InsClave
InsSector
DocId

RESULTADOS

acc_server
acc_bd
acc_login
acc_Password
acc_storeProcedure
acc_vista

acc_tabla

ssa_getAcuseRecibo

Obtiene los acuses de recibo para una Entidad, con respecto a cada documento enviado.

PARÁMETROS

EnvFolio

InsClave

InsSector

RESULTADOS

Ninguno

ssa_getBitacora

Consulta la información contenida dentro de la bitácora referente a cada envío para poder procesarlo.

PARÁMETROS

EnvFolio

InsClave

InsSector

RESULTADOS:

bit_fase

bit_mensaje

bit_numero_registro

bit_numero_columna

men_numero

ssa_getBitacoraPendiente

Permite procesar los envíos para los que estén pendientes en la bitácora, hacia el SA Central. Al mismo tiempo se desmarcan como pendientes.

PARÁMETROS

Ninguno

RESULTADOS:

env_folio

ins_clave

ins_sector

ssa_getCatalogo

Obtiene la información referente a los catálogos definidos en la base de datos.

PARÁMETROS

Ninguno

RESULTADOS

cat_id
cat_descripcion
cat_nombre

ssa_getColumnasFormato

Procedimiento que extrae el número de columnas contenidas en cada uno de los documentos.

PARÁMETROS

DocId
DocVerAnoAplicacion
DocVerPeriodoAplicacion
DocVerVersion

RESULTADOS

numero de columnas

ssa_getDocSeparadorCols

Obtiene la información para el separador de columnas dentro de un documento.

PARÁMETROS

DocVerAnioAplicacion
DocVerPeriodoAplicacion

ssa_getDoctosAccesoRemoto

Procedimiento que obtiene los documentos de una Entidad Supervisada de acuerdo a la información definida en cuanto a la periodicidad de extracción, bajo el esquema de acceso a bases de datos.

PARÁMETROS

InsNombre

ssa_getDocumentoCaptura

Procedimiento que obtiene el formato de un documento para generar la pantalla de captura dentro del CLIENTE.

PARÁMETROS

InsSector
InsClave
Usuld
DocId
Anio

Periodo

ssa_getDocumentoImportacion

Procedimiento que obtiene el formato de un documento para aplicarlo al proceso de importación de archivo.

PARÁMETROS

InsSector
InsClave
Usuld
Anio
Periodo

ssa_getDocumentoTemporal

Este procedimiento se encarga de obtener el nombre temporal que lleva asociado cada documento.

PARÁMETROS

DocId

RESULTADOS

doc_nombre_temporal

ssa_getDocumentoVersion

Permite obtener la versión a utilizar para un documento en base al año y al periodo para el cuál aplica.

PARÁMETROS

DocId
DocAñoAplicacion
DocPeriodoAplicacion
DocVerVersion (OUTPUT)

RESULTADOS

Ninguno

ssa_getDocumentos

Procedimiento almacenado que permite obtener los documentos que un usuario, en una Entidad Supervisada, está permitido a enviar.

PARÁMETROS

Usuld

RESULTADOS

doc_id
doc_descripcion

doc_periodicidad

ssa_getEnvio

Obtiene la información asociada a cada envío.

PARÁMETROS

InsSector
InsClave
EnvFolio

ssa_getEnvioLocal

Obtiene la información asociada a un envío local.

PARÁMETROS

EnvFolio
InsClave
InsSector

ssa_getEnvioPendiente

Permite obtener los envíos que estén pendientes de enviar al SA Central. Estos envíos tienen un estatus igual a IE. Al mismo tiempo se desmarcan como pendientes.

PARÁMETROS

Ninguno

RESULTADOS:

env_folio
ins_clave
ins_sector

ssa_getEnvioFolio

Procedimiento que obtiene el folio de cada envío.

PARÁMETROS

InsSector
InsClave
DocId
EnvAno
EnvPeriodo

RESULTADOS

env_folio

ssa_getEnvioInfo

Obtiene toda la información asociada a cada envío que se realiza.

PARÁMETROS

EnvFolio
InsClave
InsSector

RESULTADOS

usu_nombre
doc_id
doc_ver_version
doc_descripcion
env_ano
env_periodo
env_tabla_temporal

ssa_getExplnd

Obtiene la información de las fórmulas de validación. Solamente se obtiene para aquellas que son independientes, es decir aquellas que no son expresiones *if-then-else* o son el primer nivel del *if-then-else*.

PARÁMETROS

DocId
DocVerVersion

RESULTADOS

for_val_id
for_val_texto
then_id
else_id

ssa_getExpresion

Procedimiento que extrae las expresiones asociadas a una expresión de tipo *if-then-else*.

PARÁMETROS

ForValId

RESULTADOS

for_val_texto
then_id
else_id

ssa_getFiltro

Procedimiento que ayuda a determinar la necesidad de filtrado de información para efectuar el envío.

PARÁMETROS

EnvFolio
InsClave
InsSector

ssa_getFormato

Permite obtener el formato específico para cada uno de los campos de un documento.

PARÁMETROS

DocId
DocVerAnoAplicacion
DocVerPeriodoAplicacion
DocVerVersion

RESULTADOS

col_orden
col_nombre
col_tipo
col_longitud
col_decimales
col_formato_captura
col_longitud_despliegue

ssa_getFormatoCatalogo

Permite obtener la información referente a las columnas de los catálogos.

PARÁMETROS

CatId

RESULTADOS

cat_col_orden
cat_col_nombre
cat_col_encabezado
cat_col_tipo
cat_col_longitud
cat_col_longitud_despliegue

ssa_getImportaAutomatica

Procedimiento que ayuda a obtener todos los documentos para los cuales se debe realizar el proceso de importación automática de archivos.

PARÁMETROS

Usuld
InsSector
InsClave

ssa_getIntegracionHabilitadata

Procedimiento que determina si el documento a enviar tiene habilitada la opción de integración, en caso afirmativo se llevará a cabo dicha integración.

PARÁMETROS

DocId
InsClave
InsSector

ssa_getMapeo

Obtiene la información referente a las reglas de integración para los datos recibidos por cada documento.

PARÁMETROS

DocId
DocVerVersion

RESULTADOS

srv_nombre
srv_login
srv_Password
tab_ope_bd
tab_ope_tabla
col_orden
col_ope_columna

ssa_getNombresTemporales

Procedimiento que obtiene los nombres temporales utilizados para la identificación de las tablas temporales.

PARÁMETROS

InsClave
InsSector
DocId

ssa_getNumReglasInt

Procedimiento que obtiene el número de reglas de integración asociadas al documento, es decir a cuántas reglas será sometido el documento para lograr su integración.

PARÁMETROS

DocId

DocVerVersion
RESULTADOS
número de reglas

ssa_getNumUsers

Obtiene el número de usuarios.

PARÁMETROS
InsNombre

RESULTADOS
Número_de_usuarios

ssa_getPeticonesPorValidar

Procedimiento que obtiene la información de todos los envíos que no hayan sido validados, es decir cuyo estatus sea CO.

PARÁMETROS
Ninguno

RESULTADOS
env_folio
ins_clave
ins_sector
ins_nombre_corto
ins_clave
usu_id
doc_id
doc_ver_version
env_fecha
env_ano
env_perodo
env_tabla_temporal

ssa_getSeguimientoDoc

Obtiene la información del documento al que se le da seguimiento.

PARÁMETROS
InsSector
InsClave
EnvFolio

ssa_getServidoresApp

Obtiene las referencias de los SA Centrales a donde se deberá enviar la información de un documento.

PARÁMETROS

DocId
InsClave
InsSector

ssa_getSpValidacion

Procedimiento que obtiene los *stored procedures* utilizados para los procesos de validación de la información de un documento.

PARÁMETROS

DocId
DocVerVersion

ssa_getStatusEnvio

Este procedimiento permite obtener el estatus de un envío.

PARÁMETROS

EnvFolio
InsClave
InsSector

ssa_getUltimosAcuses

Procedimiento que obtiene la lista de los últimos acuses disponibles.

PARÁMETROS

Usuid

RESULTADOS

doc_descripcion
periodo
env_folio

ssa_getUsers

Obtiene la lista de usuarios autorizados para la operación del *middleware*.

PARÁMETROS

InsNombre
TipoSistema

RESULTADOS

usu_nombre
usu_Password
ins_nombre

ssa_getValidaCatalogo

Obtiene toda la información con respecto a las validaciones que deben realizarse contra catálogo.

PARÁMETROS

DocId
DocVerVersion

RESULTADOS

val_cat_id
cat_nombre
col_orden
cat_col_nombre

ssa_getValMensaje

Obtiene el mensaje de error asociado a cada validación realizada (fórmula, validación contra catálogos, stored procedure de validación, etc.)

PARÁMETROS

for_val_id

RESULTADOS

men_err_val_error
men_err_val_texto

ssa_nopresentarAcuseRecibo

Actualiza la información de los acuses de recibo que ya no deben presentarse en la pantalla de últimos acuses.

PARÁMETROS

InsSector
InsClave
EnvFolio

ssa_prodDocumento

Obtiene la lista de documentos que se han capturado, se han cargado en la tabla temporal o han sido verificados, pero que todavía no se envían al SA Central.

PARÁMETROS

Usuld
StatusEnv

RESULTADOS

env_folio

doc_descripcion
doc_periodicidad
env_ano
env_periodo
periodo

ssa_seguimientoDoc

Obtiene la lista de los documentos a los que se debe dar seguimiento.

PARÁMETROS

Usuld

ssa_setEnvioIntegracion

Procedimiento que establece el estatus de envío al SA Centrale para un envío.

PARÁMETROS

EnvFolio
InsClave
InsSector
EnvStatus
SerAppld

ssa_setFiltrado

Procedimiento que marca a la información de un envío, cuando debe ser filtrada antes de integrarse.

PARÁMETROS

EnvFolio
InsClave
InsSector
Filtrado

ssa_setStatusEnvio

Procedimiento que sirve para modificar el estatus en que se encuentre un envío.

PARÁMETROS

EnvFolio
InsClave
InsSector
EnvStatus

RESULTADOS

Ninguno

ssa_validaUsuario

Procedimiento utilizado por el CLIENTE para validar el que un usuario este permitido a acceder el *middleware*.

PARÁMETROS

Usuld
 UsuPassword
 InsSector
 InsClave

RESULTADOS

ins_sector
 ins_clave
 ins_nombre_temporal

5.7 Archivos

Archivo de Configuración

Este es el archivo que contiene los parámetros de configuración para SA. El nombre de este archivo se indica en la línea de comandos cuando el SA se levanta, si no se especifica un nombre, el archivo de configuración se tomará como *servername.cfg*.

Los parámetros indicados en el archivo de configuración se presentará a continuación:

Parámetro	Default	Descripción
institucion	Entidad Normativa	Nombre de la Entidad en donde se ejecuta el SA.
password	Null	Password utilizado para las conexiones al SA Central.
server_central		Valor lógico que indica si el SA es el Central.
num_validador_t	1	Número de <i>threads</i> VALIDADOR_T.
num_integrador_t	1	Número de <i>threads</i> INTEGRADOR_T.
validador_login	Validador	Login utilizado por el VALIDADOR_T para conectarse al SQL Server.
validador_pwd	Null	Password del Login utilizado por el VALIDADOR_T.
integrador_login	Integrador	Login utilizado por el INTEGRADOR_T para conectarse al SQL Server.

integrador_pwd	Null	Password del Login utilizado por el INTEGRADOR_T.
access_login	Access	Login utilizado por el ACCESS_T para conectarse al SQL Server.
access_pwd	Null	Password del Login utilizado por el ACCESS_T.
max_num_connections	25	Número máximo de conexiones simultáneas que puede aceptar el <i>middleware</i> .

El formato del archivo será:

parámetro: valor

Salvo para el parámetro SERVER_CENTRAL, si existe este parámetro significará que el SA es el de Central.

SERVIDOR_APLICATIVO.errorlog

Archivo de errores y mensajes del SA.

5.8 Parámetros en la Línea de Comandos

-S	openServerName	Nombre del Open Server.
-E	errorlog	Nombre del archivo de errores.
-C	configFile	Nombre del archivo de configuración
-D	sqlServerName	Nombre del SQL Server.
-A	SA Local	Nombre del SA Local en la Entidad Supervisada.
-T	tracelevel	Máscara de bits, que indica el nivel de rastreo que utilizará el Open Server.

5.9 Algoritmos

A continuación se presentan los algoritmos (con sus diagramas de flujo) de las funciones principales para la validación e integración de información a través del *middleware*.

5.9.1 Event handlers

Un *event handler* es una parte del código que se ejecuta cuando un evento se inicia. Cuando se dispara un evento, el *Open Server* coloca tanto al evento como

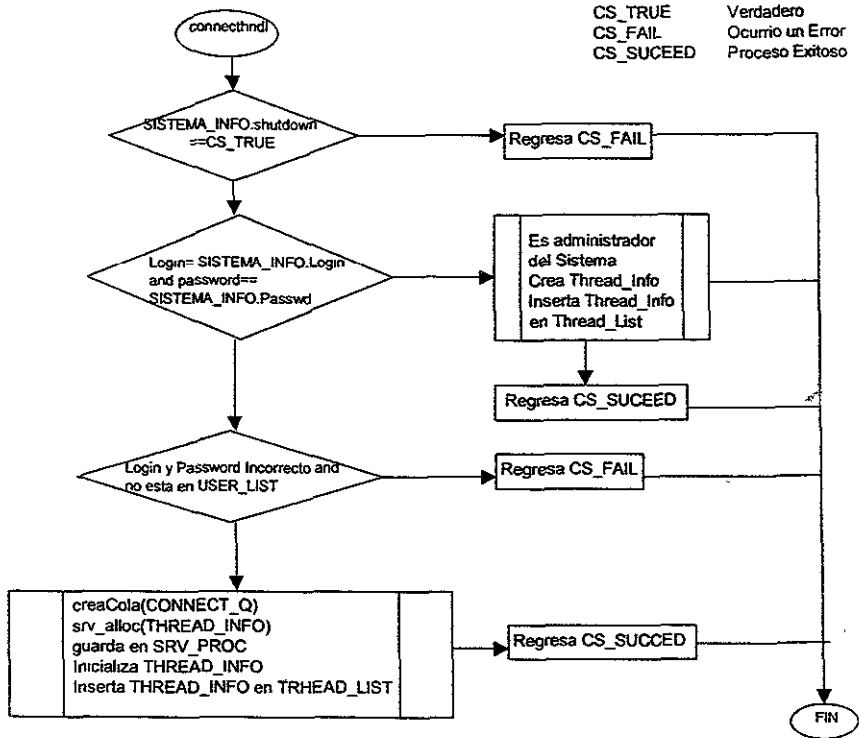
al *thread* en la cola de ejecución. Luego el *thread* ejecuta la rutina que procesa dicho evento. Dicha rutina es conocida como *event handler*.

A continuación se presentan los eventhandler diseñados para su utilización dentro del *middleware*.

connectHndl

Event Handler que es utilizado para conectarse al SA.

```
if (SISTEMA_INFO.shutdown == CS_TRUE)
    return CS_FAIL
if (Login ==SISTEMA_INFO.Login and passwd == SISTEMA_INFO passwd)
{
    /* Es el administrador del Sistema */
    crea THREAD_INFO
    THREAD_INFO ( tpo=ADMON_THREAD,
                  status = WAITING_CMD)
    inserta THREAD_INFO en THREAD_LIST
    return CS_SUCCEED
} /* if */
if (Login y passwd no son los correctos y no están en USER_LIST)
    return CS_FAIL
creaCola(CONNECT_Q)
srv_alloc(THREAD_INFO) y guardar en SRV_PROC (utilizando SRV_T_USERDATA)
Inicializa THREAD_INFO( tipo = USER_THREAD
                        status = WAITING_CMD
                        hostname, institucion, usuario,
                        msgQId de la cola creada)
Inserta THREAD_INFO en THREAD_LIST
return CS_SUCCEED
}
```

Figura 5.14 Diagrama de Flujo para la función *connectHndl***DisconnectHndl**

Event Handler que es utilizado para la desconectarse del SA.

```

{
  borra THREAD_INFO de THREAD_LIST
  srv_free(THREAD_INFO)
  if (Login == SISTEMA_INFO.Login and passwd == SISTEMA_INFO.passwd)
  {
    /* Es el administrador del Sistema */
    CS_SUCCEED
  }
  borraCola(CONNECT_Q)
  return CS_SUCCEED
}

```

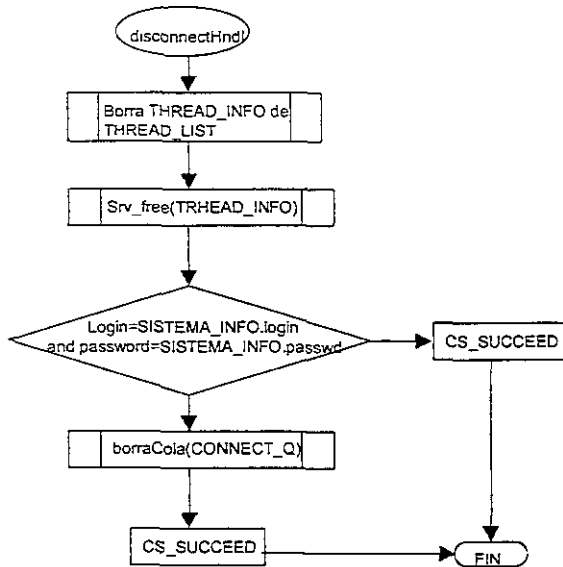


Figura 5.15 Diagrama de Flujo para la función *disconnectHndl*

LanguageHndl

Event Handler que es utilizado para el manejo del idioma a utilizar.

```

{
    Parsea los procedimientos registrados definidos y sus parámetros
    Ejecuta el Procedimiento registrado adecuado
    /* aqui no se hace srv_senddone */
    return CS_SUCCEED
}
  
```

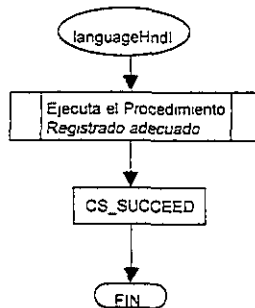


Figura 5.16 Diagrama de Flujo para la función *languageHndl*.

shutdownHndl

Event Handler que es utilizado para dar de cerrar la conexión con el SA

```

{
    SISTEMA_INFO.shutdown = CS_TRUE
    while(MONITOR_INFO.peticionesActivas > 0)
        srv_yield()
    despierta threads internos para que realicen el shutdown en forma correcta
}
  
```



```

}
    srv_event(SRV_STOP)
}

```

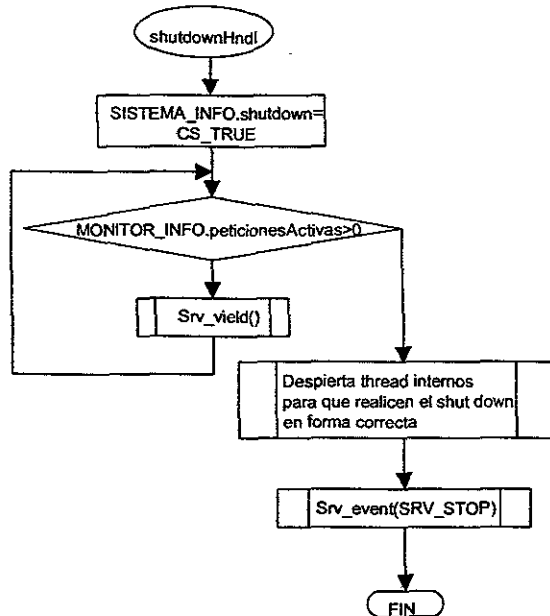


Figura 5.17 Diagrama de Flujo para la función *shutdownHndl*.

startHndl

Event Handler que es inicializa todos los procedimientos del SA.

```

{
    alloc e inicialización de CACHE
    alloc e inicialización de DOCTO_LIST
    alloc e inicialización de MONITOR_INFO
    alloc e inicialización de QUEUE_LIST
    alloc e inicialización de SISTEMA_INFO
    alloc e inicialización de THREAD_LIST
    alloc e inicialización de USER_LIST(
    Crea cada uno de los Procedimientos Registrados (hay que liberar el thread temporal)
    Levanta el ACCESS_T
    Levanta el NOTIFICADOR_T
    Levanta el UPD_DATA_T
    Levanta el número configurado de VALIDADOR_T
    Levanta el número configurado de INTEGRADOR_T
    Para cada thread levantado:
        crear THREAD_INFO
        inicializa THREAD_INFO:
            threadId
            tipo
        insertar TREAD_INFO en THREAD_LIST
        asignar THREAD_INFO en la propiedad SRV_T_USERDATA
        asignar el nombre del thread en SRV_T_USTATE
    Crear cada una de las colas del sistema
    Define el evento de SHUTDOWN
    return CS_SUCCEED
}

```

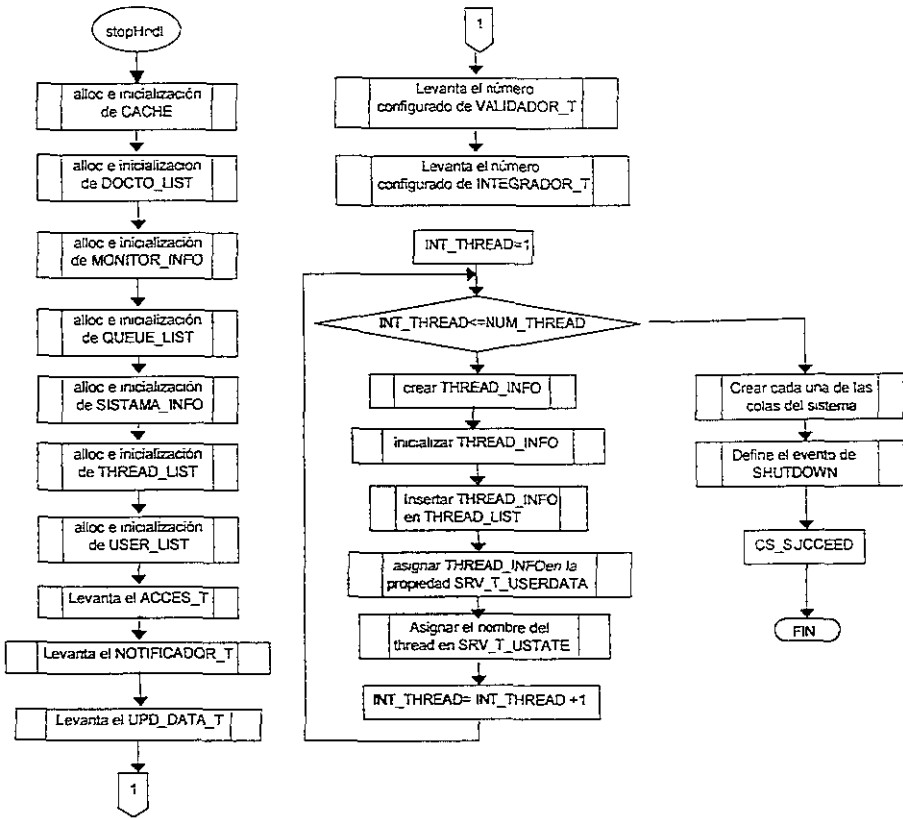


Figura 5.18 Diagrama de Flujo para la función startHndl.

stopHndl

Event Handler que detiene los procedimientos del SA.

```

{
    dealloc CACHE
    dealloc DOCTO_LIST
    dealloc MONITOR_INFO
    dealloc MONITOR_INFO
    dealloc QUEUE_LIST
    dealloc SISTEMA_INFO
    dealloc THREAD_LIST
    dealloc USER_LIST
    return CS_SUCCEED
}
    
```

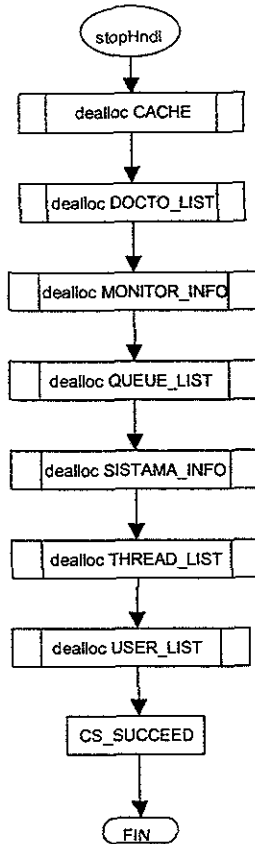


Figura 5.19 Diagrama de Flujo para la función *stopHndI*.

5.9.2 Procedimientos registrados

Los procedimientos registrados son todos aquellos que se ejecutan dentro de los procesos de extracción, validación e integración de la información a través de *middleware*.

rp_accesalInfo

Procedimiento que se utiliza para realizar la conexión al Servidor Central de la Entidad Supervisada y realizar las peticiones definidas en la metadata

```

{
    Obtener parametros del RPC
    if (SISTEMA_info.tipoSistema == SERVER_CENTRAL)
    {
        Ejecuta RPC rp_accesalInfo en el SERVIDOR APLICATIVO adecuado
        return CS_SUCCEED
    }
}
  
```

```
}
srv_alloc(DOCTO_INFO), srv_bzero(DOCTO_INFO)
inicializa DOCTO_INFO,
srv_alloc(PETICION), srv_bzero(PETICION)
inicializa PETICION (tipo = PETICION_CONECTADA)
modificar THREAD_INFO(status = PETICION_RECIBIDA)
escribir en ACCESS_Q
if (PETICION.tipo == PETICION_SINCRONA)
{
    while(no se obtenga el último mensaje de resultado)
    {
        leeCola (CONNECT_Q) →wait y sleep
        case (RESULT_INFO.tipoMsg)
        MENSAJE_INFO:
            ERROR_MSG:
                Enviar todos los mensajes en RESULT_INFO msgList al cliente
                srv_free(RESULT_INFO)
                break;
            FINAL_MSG:
                if (mensaje anterior == ERROR_MSG)
                    envia estatus de error
                                THREAD_INFO.peticionesFallidas++
                                MONITOR_INFO.peticionesFallidas++
                    else
                        envia estatus de éxito
                                THREAD_INFO.peticionesExitosas++
                                MONITOR_INFO.peticionesExitosas++
        } /* while */
    } /* if */
    else
    {
        envia status de éxito
        THREAD_INFO.peticionesAsincronas++
        MONITOR_INFO.peticionesAsincronas++
    }
}
```

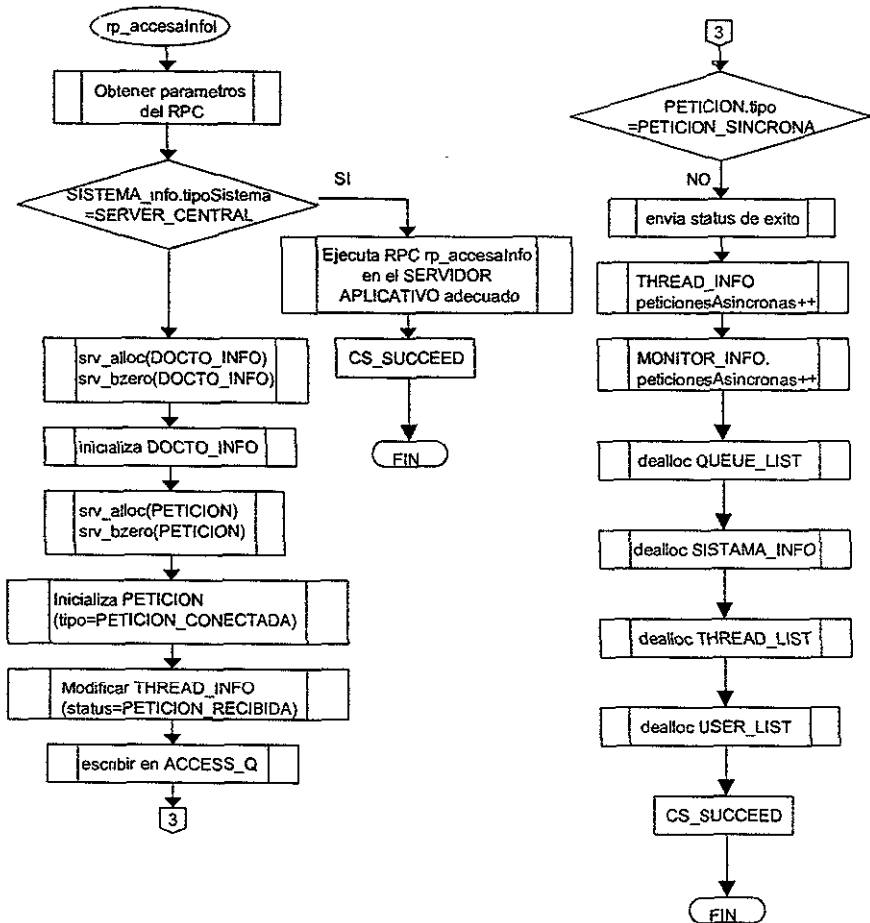


Figura 5.20 Diagrama de Flujo para el procedimiento almacenado *rp_accesaInfo*.

rp_enviaBitacora()

Procedimiento que genera y envía los reportes de la bitácora al CLIENTE.

```

{
  Obtener parámetros del RPC
  Inicializar en THREAD_INFO: folio, cveInstitucion, cveSector, bulkType(BIT_BULK)
  srv_alloc(PETICION_BLK); srv_bzero(PETICION)
  Llenar PETICION_BLK(threadId, tipoCmd=BLK_CMD, comando)
  Escribir PETICION_BLK en BULK_Q
  Esperar la respuesta en ACCESS_Q
  Dependiendo del tipo de respuesta leído en RESULT_INFO enviar el mensaje adecuado al cliente
  return CS_SUCCEED
}

```

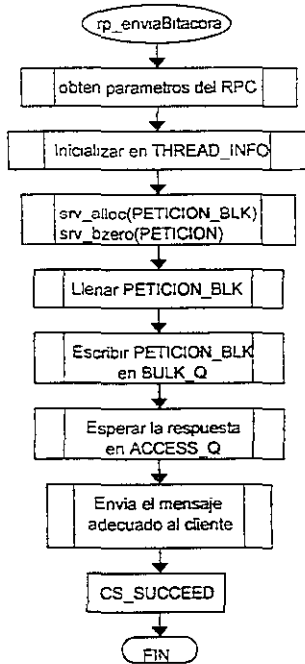


Figura 5.21 Diagrama de flujo para el procedimiento almacenado *rp_enviaBitacora*.

rp_integraPeticion()

Este procedimiento se utiliza para integrar y ejecutar las peticiones previamente definidas.

```
{  
    Obtener parámetros del RPC  
    Crear el comando de inserción en la tabla de envío que aplicará el BULK_T(utilizando todos los parametros recibidos)  
    obtener THREAD_INFO utilizando la propiedad SRV_T_USERDATA  
    Guardar en THREAD_INFO: folio, cveInstitucion, cveSector  
    srv_alloc(PETICION_BLK); srv_bzero(PETICION)  
    Llenar PETICION_BLK(threadId, tipoCmd=INS_CMD, comando)  
    Escribir PETICION_BLK en BULK_Q  
    return CS_SUCCEED  
}
```

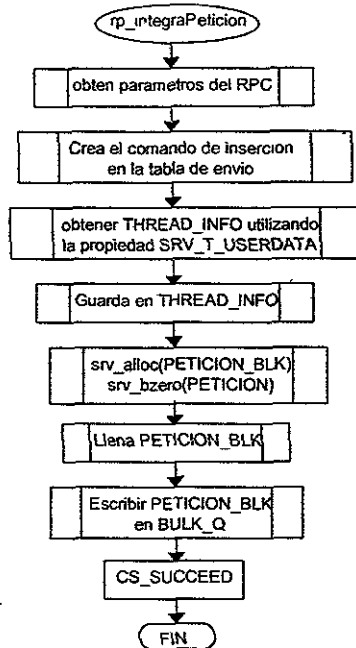


Figura 5.22 Diagrama de Flujo para el procedimiento almacenado *rp_integraPetición*.

rp_procDocumento()

Procedimiento que obtiene los parámetros del RPC para procesar los documentos.

```

{
  Obtener parámetros del RPC
  obtener THREAD_INFO de SRV_T_USERDATA
  srv_alloc(PETICION); srv_bzero(PETICION)
  inicializa PETICION:      tipo |= PETICION_CONECTADA
                          usuario, folio, cveInstitucion, cveSector
                          status=V
  tipo (de acuerdo al parámetro recibido determinar si es sincróna o no)
                          threadid
  modificar THREAD_INFO(status = PROCESS_REQ)
  escribir en VALIDA_Q
  if (PETICION.tipo == PETICION_SINCRONA)
  {
    leeCola (CONNECT_Q) --wait y sleep
    case (RESULT_INFO.tipoMsg)
      ERROR_MSG:
        Enviar el mensaje en RESULT_INFO.msg al cliente
        Envía número de error como status
        srv_free(RESULT_INFO)
        break;
      FINAL_MSG:
        envía status de éxito
        THREAD_INFO.peticionesExitosas++
        MONITOR_INFO.peticionesExitosas++
  }
} /* if */
else
{

```

```

        envia status de éxito
        THREAD_INFO.peticionesAsincronas++
        MONITOR_INFO.peticionesAsincronas++
    }
}

```

rp_updRemDoctos

Procedimiento que inicializa las peticiones y las escribe en la cola de accesos

```

{
    srv_alloc(PETICION); srv_bzero(PETICION)
    inicializa PETICION:      servicio = UPD_ACCESO_REMOTO
    escribir en ACCESS_Q
}

```

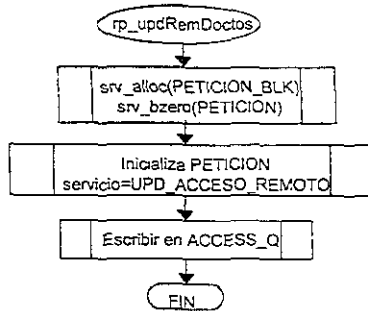


Figura 5.23 Diagrama de Flujo para el procedimiento almacenado *rp_updRemDoctos*.

5.9.3 Thread handlers

Los *Thread Handlers* son los procedimientos principales que integran y ejecutan a los procedimientos e *Event Handlers* para que en conjunto realicen las funciones de acceso, inicialización, extracción, validación e integración de los documentos.

accesHndl()

Control del acceso e inicializaci3n

```

{
    Crea ACCESS_Q
    Abre una conexi3n constante hacia el SQL Server
    Ejecuta ssa_GetRemDoctos y con ello llena la lista DOCTO_LIST en donde se mantienen los documentos que se
    accesar3n remotamente
    while(1)
    {
        if (es tiempo de obtener documentos mediante acceso remoto para DOCTO_LIST.first)
        {
            Ejecuta ssa_getAccessInfo para la instituci3n
            Establece la conexi3n remota con la informaci3n obtenida
            if (existen errores al realizar la conexi3n)
            {
            }
        }
        Se ejecuta el stored procedure, tabla o vista requiendo
        Se crea la tabla temporal en BASE DE DATOS DE ADMINISTRACION
        Se cargan los datos en la tabla TEMPORAL
    }
}

```



```

        Se ejecuta ssa_creaBitacora para generar el número de folio
        srv_alloc (DOCTO_INFO); srv_bzero(DOCTO_INFO)
inicializa DOCTO_INFO;
srv_alloc(PETICION); srv_bzero(PETICION)
inicializa PETICION; tipo = PETICION_DESCONECTADA, folio,
status=DATOS_CARGADOS
}
leeCola (ACCESS_Q) - nowait-
if (se leyó un mensaje)
{
if (PETICION.servicio == UPD_ACCESO_REMOTO)
{
Ejecuta ssaGetRemDoctos y con ello llena la lista DOCTO_LIST
}
else
{
Ejecuta getAccessInfo para la institución
Establece la conexión remota con la información obtenida
Se ejecuta el stored procedure, tabla o vista requerido
Se crea la tabla temporal en BASE DE DATOS DE ADMINISTRACION
Se cargan los datos
Se ejecuta ssa_creaBitacora para generar el número de folio
PETICION.status=DATOS_CARGADOS
Si el documento obtenido esta en DOCTO_LIST, colocarlo en la posición adecuada
}
}
srv_yield()
}
}

```

integradorHndI()

Integrador de la información.

```

{
Crea INTEGRADOR_Q (esta cola se crea solamente una vez no importando el número de threads levantados)
Abre una conexión constante hacia el SQL Server
while(1)
{
leeCola (VALIDADOR_Q) - nowait-
if (leyó mensaje)
{
determina el documento al cual pertenece PETICION
THREAD_INFO (inicializa documento, institución y usuario con los valores en
PETICION.documento, esto para el INTEGRADOR_T)
if (SISTEMA_INFO.tipoSistema == SERVER_LOCAL)
{
if (existe conexión hacia ENTIDAD SUPERVISADORA)
{
Ejecuta rp_integraPetición
Envía HISTORICO usando bulk-copy
}
else
{
Ejecuta ssa_envioPendiente en el SQL Server
}
}
else
{
Busca en CACHE PETICION.documento
if (PETICION.documento esta en CACHE)

```

```

{
    if (el documento en memoria tiene cargadas las reglas de integración
        ( reglasCargadas == REGLAS_INTEGRACION))
    {
        copia de CACHE a PETICION documento.reglasintegracion
        PETICION.documento.reglasCargadas !=REGLAS_INTEGRACION
    }
}
if (no se cargaron las reglas de integración)
{
    Ejecuta ssa_getValidaciones para PETICION.documento
    PETICION.documento.reglasCargadas !=REGLAS_VALIDACION
}
reemplaza PETICION.documento en CACHE (utilizando estrategia MRU/LRU)
Integra la información a partir del HISTORICO
}
    
```

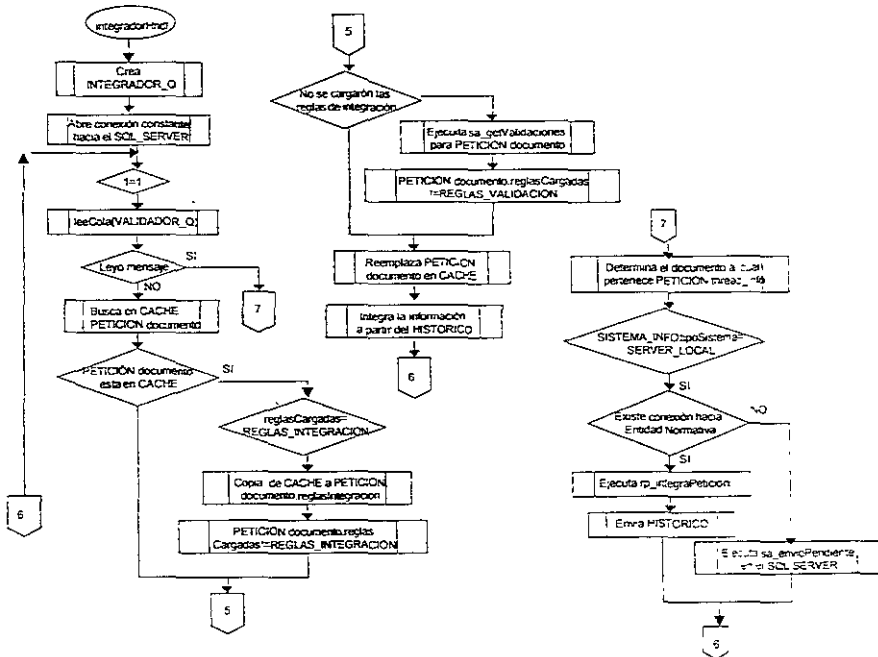


Figura 5.24 Diagrama de Flujo para el procedimiento almacenado *integradorHndI*.

validadorHndI()

Valida la información integrada previamente.

```

{
    Se Abre una conexión constante hacia el SQL Server utilizando un Login especial para el
VALIDADOR_T
    Se obtienen los OBJECT_ID de las messages queues (INTEGRADOR_Q y VALIDADOR_Q).
while(CS_TRUE)
{
    Lee de la cola VALIDADOR_Q y se queda en estado de sleep hasta que lea algo (EL haber leído
algo significa que se hizo una petición al VALIDADOR_T).
    Se borra la bitacora de VALIDACION que pudiera tener asociada el envío, ya que anteriormente se
pudieron haber realizado vanos envíos para validación solo
    
```

```

Se ejecuta ssa_getEnvioInfo para obtener información del envío, en particular se debe obtener la
información del documento asociado y la tabla temporal de donde se toman los datos.
Se ejecuta ssa_getSpValidacion, para así obtener los stored procedures de validacion y guardarlos
en PETICION.
Se obtienen las validaciones contra catálogo ( ssa_getValidaCatalogo).
Se obtienen las validaciones que son expresiones independientes ( ssa_getExpInd)
Recorrer las expresiones cargadas y para cada expresión que sea if:
    Ejecutar en forma recursiva ssa_getExpresion para obtener cada una de las expresiones del
    if, hasta que se llegue al nivel de hoja ( de acuerdo al árbol que representa al if).
Se obtiene el formato de las columnas del documento, para después poder realizar la carga de
información de la tabla temporal al HISTORICO.
    Se crea el comando que nos permitirá la inserción de la información en la tabla temporal en el
    HISTORICO. El HISTORICO tiene como nombre HIST_docId_docIdVersion.
    (Hay que recordar que el HISTORICO tiene como columnas el folio, la clave de la institución,
    el sector
    y las columnas de la tabla temporal del tipo adecuado).
    Se realiza la inserción en el HISTORICO, reportando cada error que pudiera aparecer por
registro.
If ( no existieron errores)
Para cada validación en VALIDACION_CAT_LIST, no importa el número de columnas, mapeadas:
Se forma un query( el query debe regresar los ids de los registros que no cumplen
Con la validación)
Si hay errores
Ejecutar ssa_getValMensaje para obtener el mensaje asociado a la
Fórmula, que será el utilizado para generar la bitácora
Para cada registro que no cumple con la validación:
Generar la bitácora
Para cada renglón de HISTORICO (hay que almacenar el renglón anterior):
Para cada validación en VALIDACION_EXP_LIST:
    Evaluar la expresión recomiendo las ramas IF-THEN-ELSE si existen y
    substituyendo a cada referencia de columna el valor correcto (utilizar
    Algoritmo recursivo)
    Si el resultado es FALSE
    Generar la bitácora con el mensaje adecuado.
    Ejecuta los Stored Procedures de validación en el orden que les corresponde
If (no existieron errores al ejecutar todas las validaciones)
{

if (la petición es sólo de validación, PETICION.servicio == SERV_VALIDACION)
{
    Regresa un mensaje al CONNECT_T indicando el éxito de la petición
    Borra el HISTORICO
    continue
}

En el caso de que se haya pedido servicio de validación ( no sólo de prueba), una vez que
Se ha terminado la validación y creado el HISTORICO, se debe borrar la tabla temporal
asociada.
Se escribe en la bitácora el hecho de que la validación haya resultado exitosa y se cambia el
Status del envío.
Se determina si el documento debe pasar por el proceso de integración o bien debe ser
Integrado manualmente ( para ello se ejecuta ssa_IntegracionHabilitada)
if (si el docto. tiene habilitada la opción de integracion)
    SE escribe en INTEGRA_Q_NAME
If ( la petición es SINCRONA)
    Se escribe un mensaje al CONNECT_T
}
else
{
Si existieron errores de validación por stored, se borra la información que se
había cargado al HISTORICO
If ( la petición es SINCRONA)
    Se escribe un mensaje al CONNECT_T
}

}

Se cierra la conexión hacia el SQL Server
}

```

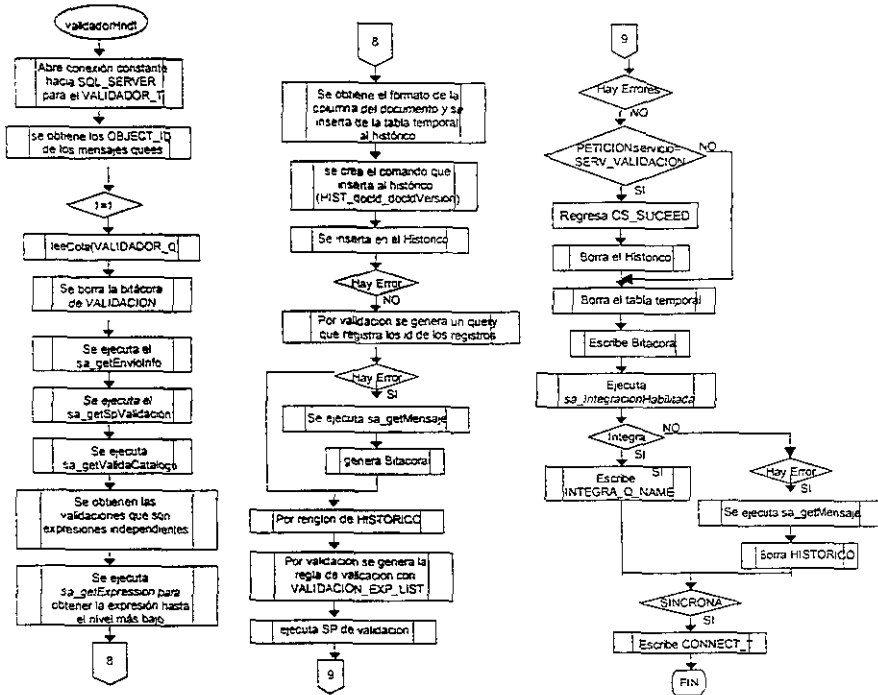


Figura 5.25 Diagrama de Flujo del procedimiento almacenado *validadorHndI*.

En este capítulo hemos revisado las estructuras de datos que se utilizarán, así como todos los procedimientos que se involucran para lograr la aplicación propuesta. De la misma forma pasaremos a continuación a mencionar el diseño que será utilizado en la base de datos de la Metadata, donde analizaremos todas y cada una de las tablas involucradas en el desarrollo.

Capítulo 6

Diseño de la Base de Datos Metadata

En este capítulo se presenta tanto el diseño conceptual como el diseño físico de la base de datos Metadata.

6.1 Modelo Conceptual

El Modelo Conceptual de Datos o Modelo Lógico de Datos es el esquema que resulta del proceso de análisis y representa los requerimientos de datos del sistema de información que se está modelando.

El Modelo conceptual de datos debe cumplir con las siguientes características:

- **Claridad y precisión.-** , Esto es, debe describir completamente los datos, la semántica de los mismos, sus relaciones, la seguridad, la integridad y los alcances del Sistema de Información que representa.
- **Constancia.-** Debe contemplar requerimientos presentes y futuros para evitar cambios frecuentes en el modelo.
- **Flexibilidad.-** Debe de poderse adaptar a nuevos requerimientos sin tener que sufrir cambios significativos. Un buen modelo conceptual no debe permitir que se afecten las vistas definidas sobre éste al hacer cambios.

- **Integrabilidad.-** Integración de aplicaciones, el modelo conceptual debe servir como base para el desarrollo de las múltiples aplicaciones que se den en el sistema de información; no solamente de aquellas que se proyecten al mismo tiempo que se desarrolla el modelo de datos, sino también para aquellas que se presenten durante toda la vida útil de la Base de Datos.
- **Independencia.-** Independiente del *software* o *hardware* utilizado para su implementación.

Modelo Entidad Relación

El Modelo Entidad-Relación está basado en la percepción de un mundo real que se compone de un conjunto de objetos básicos llamados "entidades", descritos por sus "atributos", y de "relaciones" entre esos objetos.

El modelo entidad-relación proporciona una ayuda gráfica conocida como Diagrama Entidad-Relación (DER), que permite la representación de la estructura lógica del modelo de datos.

6.1.1 Diseño conceptual de la Metadata

A continuación se presenta el modelo conceptual que se diseñó para la metadata.

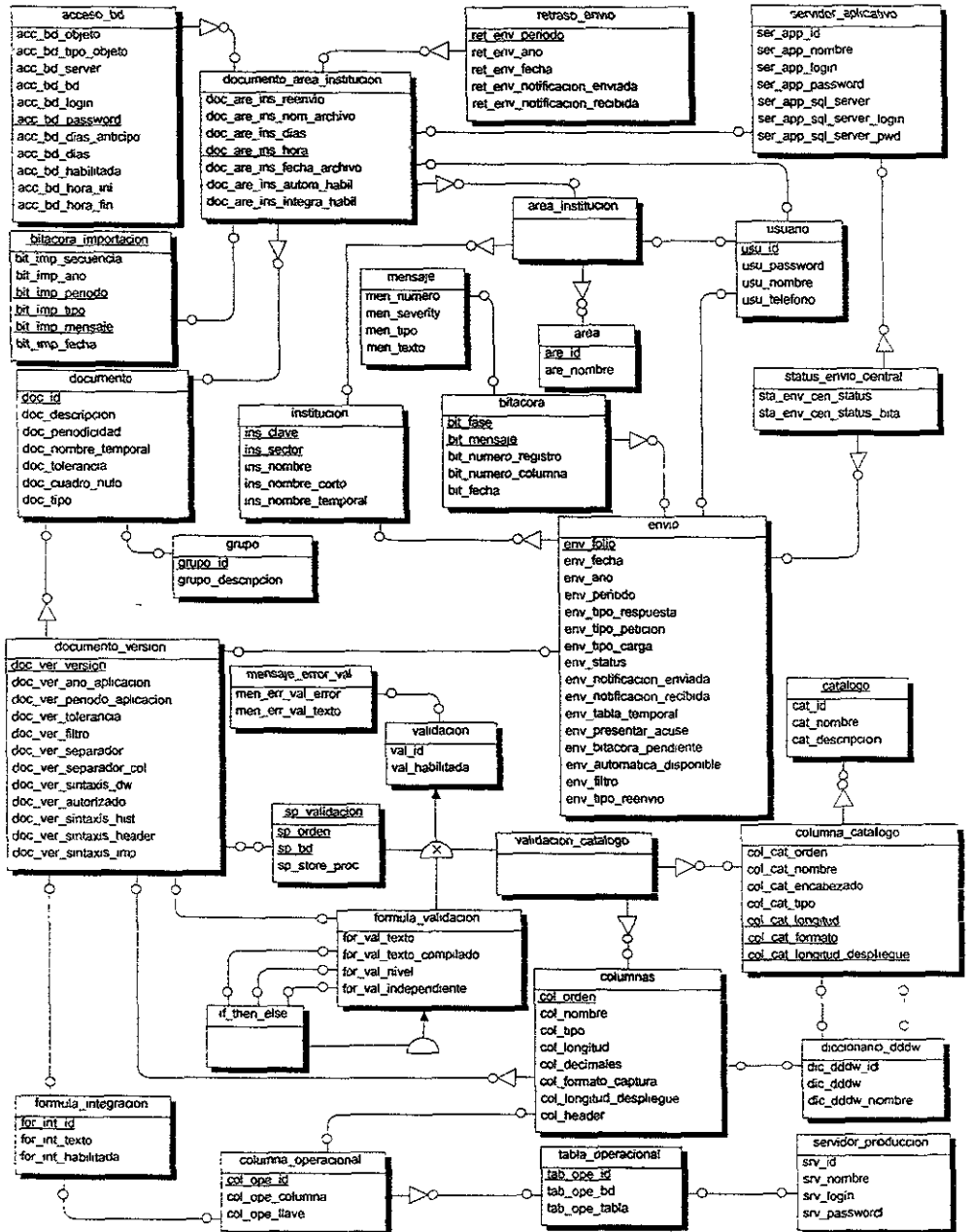


Figura 6.1 Diseño del modelo conceptual.

Ahora presentamos la descripción de las entidades del modelo conceptual de la metadata, con sus atributos y las relaciones que tiene con otras entidades.

Entidad: acceso_bd

Esta entidad mantiene la información necesaria para obtener un documento mediante el acceso remoto a una base de datos cualquiera. La información que se obtiene reside en una tabla o es resultado de un procedimiento almacenado o vista.

Lista de Atributos

Nombre	Tipo	PK	Obligatorio
acc_bd_objeto	varchar(15)	No	Si
acc_bd_tipo_objeto	char(1)	No	Si
acc_bd_server	varchar(15)	No	Si
acc_bd_bd	varchar(15)	No	No
acc_bd_login	varchar(15)	No	Si
acc_bd_password	varchar(15)	No	No
acc_bd_dias_anticipo	smallint	No	No
acc_bd_dias	char(7)	No	Si
acc_bd_habilitada	smallint	No	No
acc_bd_hora_ini	smallint	No	No
acc_bd_hora_fin	smallint	No	No

acc_bd_objeto

Nombre del objeto que permitirá la obtención de la información del documento en cuestión. Este objeto puede ser una tabla, una vista o un *stored procedure*.

acc_bd_tipo_objeto

En este atributo se indica el tipo de objeto que se va a utilizar para extraer la información por acceso a base de datos de una Entidad Supervisada. Tipo de objeto: T (Tabla), V(Vista), P(*Stored Procedure*).

acc_bd_server

Nombre del servidor de bases de datos en donde reside la información del documento.

acc_bd_bd

Nombre de la base de datos en donde reside la información del documento.

acc_bd_login

Login utilizado para acceder el servidor de bases de datos.

acc_bd_password

Password asociado al *login*.

acc_bd_dias_anticipo

Número de días antes de la fecha límite de envío de un documento, en los que se comenzará a tratar de obtener la información del documento.

acc_bd_dias

Días de la semana en los que se podrá realizar el acceso remoto, considerando que el domingo es el día número 1. Por ejemplo, para indicar que se realice lunes, miércoles y viernes, este campo tendrá el siguiente valor: "246".

Valores por Default: 1234567 (todos los días)

acc_bd_habilitada

Indica si se tiene habilitada o no la opción de acceso remoto para un documento.

Valor por default: 0 (Opción no habilitada)

acc_bd_hora_ini, acc_bd_hora_fin

Los atributos **acc_bd_hora_in** y **acc_bd_hora_fin** indican el rango de horas en el cuál se puede realizar el acceso remoto para la obtención de información.

Valor mínimo: 0

Valor máximo: 24

Lista de relaciones

Entidad	Cardinalidad	Dependiente	Relación
documento_area_institucion	1,N	Si	doc_are_ins_a_acc_bd

Entidad: area

Entidad que mantiene el catálogo de áreas de la Entidad Supervisada (Institución).

Lista de Atributos

Nombre	Tipo	PK	Obligatorio
are_id	smallint	Si	Si
are_nombre	varchar(40)	No	No

are_id

Identificador de cada área de la Entidad Supervisada.

are_nombre

Nombre del área para cada Entidad Supervisada.

Lista de relaciones

Entidad	Cardinalidad	Dependiente	Relación
area_institucion	0,1	Si	are_a_are_ins

Entidad: area_institucion

Entidad que mantiene la relación entre las áreas y la institución a la que pertenece cada una de ellas.

Lista de relaciones

Entidad	Cardinalidad	Dependiente	Relación
area	1,1	Si	are_a_are_ins
documento_area_institucion	0,1	Si	are_ins_a_doc_are_ins
usuario	0,n	No	are_ins_a_usu
institucion	1,1	Si	ins_a_are_ins

Entidad: bitacora

Entidad en la que se guardan los mensajes que conforman la bitácora para los procesos de recepción, validación e integración de la información de los documentos. Los mensajes de la bitácora son los que conforman el acuse de recibo para un documento enviado.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
bit_fase	varchar(12)	No	No
bit_mensaje	varchar(80)	No	No
bit_numero_registro	smallint	No	Si
bit_numero_columna	smallint	No	Si
bit_fecha	date time	No	No

bit_fase

Fase o proceso para el cual aplica el mensaje: CARGA, VALIDACIÓN, E INTEGRACIÓN.

Lista de valores: CARGA, VALIDACION E INTEGRACION.

bit_mensaje

Texto del mensaje.

bit_numero_registro

En el caso de que sea un mensaje de error y el error se pueda asociar a un registro, este atributo indica el número de registro de acuerdo a los registros recibidos para el documento.

Valor por default: 0.

bit_numero_columna

En el caso de que sea un mensaje de error y el error se pueda asociar a un registro, este atributo indica el número de columna que presenta el error.

Valor por default: 0.

bit_fecha

Fecha en la que se generó el mensaje.

Lista de relaciones

Entidad	Cardinalidad	Dependiente	Relación
envio	1,1	Si	env_a_bit
mensaje	0,1	No	men_a_bit

Entidad: bitacora_importacion

Entidad en la que se guardan los mensajes que indican como se realizaron los procesos de Importación automática de archivos o Acceso remoto a base de datos, para aquellos documentos cuya información se obtiene mediante dichos mecanismos.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
bit_imp_secuencia	numeric(10)	Si	Si
bit_imp_ano	smallint	No	Si
bit_imp_periodo	smallint	No	Si
bit_imp_tipo	char(1)	No	Si
bit_imp_mensaje	varchar(80)	No	No
bit_imp_fecha	date time	No	No

bit_imp_secuencia

Identificador del mensaje.

bit_imp_ano

Año asociado al periodo del documento.

bit_imp_periodo

Periodo asociado al documento, para el cual aplica el mensaje.

bit_imp_tipo

La bitácora puede corresponder a la importación de un archivo a el acceso remoto de bases de datos. Este atributo indica el tipo de bitácora: A (Importación de archivos), B (Acceso a bases de datos).

Valor por default: A.

bit_imp_mensaje

Texto del mensaje.

bit_imp_fecha

Fecha en la que se genera el mensaje.

Lista de relaciones

Entidad	Cardinalidad	Dependiente	Relación
documento_area_institucion	1,1	No	doc_are_ins_a_bit_imp

Entidad: catalogo

Entidad que mantiene información de las tablas de la base de datos de catálogos que pueden ser consultadas desde el CLIENTE.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
cat_id	integer	Si	Si
cat_nombre	varchar(20)	No	Si
cat_descripcion	varchar(50)	No	No

cat_id
Identificador del catálogo.

cat_nombre
Nombre de la tabla.

cat_descripcion
Descripción del contenido de la tabla de catálogo.

Lista de relaciones

Entidad	Cardinalidad	Dependiente	Relación
columna_catalogo	0,1	Si	cat_a_col_cat

Entidad: columna_catalogo

Columnas que forman la tabla de catálogo.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
col_cat_orden	smallint	Si	Si
col_cat_nombre	varchar(15)	No	No
col_cat_encabezado	varchar(20)	No	No
col_cat_tipo	char(10)	No	No
col_cat_longitud	smallint	No	No
col_cat_formato	varchar(20)	No	No
col_cat_longitud_despliegue	smallint	No	No

col_cat_orden
Orden en que estarán las columnas dentro del catálogo.

col_cat_nombre
Nombre de cada una de las columnas.

col_cat_encabezado
Encabezado de cada una de las columnas

col_cat_tipo

Tipo de la columna: *char, varchar, integer, numeric, decimal, float, etc.*

col_cat_longitud

Es la longitud del tipo de datos utilizado.

col_cat_formato

Formato en como se despliega la columna al ser consultada.

col_cat_longitud_despliegue

Longitud máxima que será utilizada para el despliegue de los valores de la columna.

Lista de relaciones

Entidad	Cardinalidad	Dependiente	Relación
catalogo	1,1	Si	cat_a_col_cat
diccionario_dddw	0,n	No	col_cat_a_dic_ddd
validacion_catalogo	0,1	Si	col_cat_a_val_cat
diccionario_dddw	0,n	No	dic_ddd_a_col_cat

Entidad: columna_operacional

Entidad que mantiene la información de las columnas de las tablas del repositorio central.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
col_ope_id	smallint	Si	Si
col_ope_columna	varchar(20)	No	Si
col_ope_llave	boolean	No	Si

col_ope_id

Identificador secuencial de la columna.

col_ope_columna

Nombre de la columna.

col_ope_llave

Columna que indica si la columna operacional forma parte de la que se toma como llave. Es importante identificar la llave ya que ésta sirve para identificar, en las tablas en el repositorio central, los registros de un envío y así poder realizar reenvíos con reemplazo de información.

Valor por default: 0.

Lista de relaciones

Entidad	Cardinalidad	Dependiente	Relación
columnas	0,n	No	integracion_columnas
formula_integracion	0,n	No	integracion_formulas
tabla_operacional	1,1	Si	tab_ope_a_col_ope

Entidad: columnas

Entidad que mantiene la información de las columnas por las que está formado un documento.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
col_orden	smallint	Si	Si
col_nombre	varchar(60)	No	No
col_tipo	char(10)	No	Si
col_longitud	smallint	No	Si
col_decimales	smallint	No	Si
col_formato_captura	varchar(100)	No	No
col_longitud_despliegue	smallint	No	No
col_header	booleano	No	Si

col_orden

Orden de la columna en el documento.

col_nombre

Nombre de la columna.

col_tipo

Tipo de la columna: *char, varchar, integer, numeric, decimal, float, etc.*

col_longitud

Longitud máxima de un valor en la columna.

col_decimales

Número de decimales de la columna, cuando el tipo es *numeric* o *decimal*.

col_formato_captura

Formato de captura utilizado para la columna en la pantalla de captura del CLIENTE.

col_longitud_despliegue

Longitud de despliegue en la pantalla de captura del CLIENTE.

col_header

Este campo determina si la columna forma parte del encabezado de un documento. Las columnas que forman parte del encabezado tienen el mismo valor para todos los renglones que forman el documento.

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
validacion_catalogo	0,1	Si	col_a_val_cat
diccionario_dddw	0,1	No	dic_ddd_a_col
documento_version	1,1	Si	doc_ver_a_col
columna_operacional	1,n	No	integracion_columnas

Entidad: diccionario_dddw

Especifica los DropDownDataWindow asociados a las columnas del documento.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
dic_dddw_id	integer	Si	Si
dic_dddw	text(2000)	No	No
dic_dddw_nombre	varchar(20)	No	No

dic_dddw_id

Identificador del DropDownDataWindow.

dic_dddw

Sintaxis para crear la *DataWindow* del *DropDownDataWindow*.

dic_dddw_nombre

Nombre que se asigna al DropDownDataWindow.

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
columna_catalogo	1,1	No	col_cat_a_dic_ddd
columnas	0,n	No	dic_ddd_a_col
columna_catalogo	1,1	No	dic_ddd_a_col_cat

Entidad: documento

Entidad que representa a los documentos que puede enviar una Entidad Supervisada.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
doc_id	integer	Si	Si
doc_descripcion	varchar(40)	No	No
doc_periodicidad	char(1)	No	Si
doc_nombre_temporal	char(10)	No	Si

Nombre	Tipo	PK	Obligatorio
doc_tolerancia	smallint	No	No
doc_cuadro_nulo	boolean	No	Si
doc_tipo	char(1)	No	Si

doc_id

Identificador del documento.

doc_descripción

Descripción o nombre del documento.

doc_periodicidad

Periodicidad con la que se recibe la información del documento. La periodicidad puede ser: anual (A), semestral(6), trimestral(T), bimestral(B), mensual(M), semanal(S) y diaria(D).

Valor por default: M

doc_nombre_temporal

Es el nombre a utilizar para la creación de la tabla temporal (ins_nombre_temporal+"_"+doc_nombre_temporal+"_"+ano+"_"+periodo)

doc_tolerancia

Número de días de tolerancia para la recepción del documento, antes de que se considere como retraso.

doc_cuadro_nulo

Este atributo indica si para el documento está permitido el envío de cuadros de información nula. Los cuadros de información nula, son documentos que no contienen datos.

Valor por default: 0

doc_tipo

Atributo que indica el tipo de documento, existen dos posibles: tabular(T) y matricial(M).

Valor por default: T

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
documento_area_institucion	0,1	Si	doc_a_doc_are_ins
documento_version	1,1	Si	doc_ver_a_doc
grupo	1,1	No	grupo_documento

Entidad: documento_area_institucion

Entidad que mantiene información respecto a los documentos que un área de una Entidad Supervisada está permitida a enviar.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
doc_are_ins_reenvio	boolean	No	Si
doc_are_ins_nom_archivo	varchar(70)	No	No
doc_are_ins_dias	smallint	No	No
doc_are_ins_hora	time	No	No
doc_are_ins_fecha_archivo	varchar(50)	No	No
doc_are_ins_autom_habil	boolean	No	Si
doc_are_ins_integra_habil	boolean	No	Si

doc_are_ins_reenvio

Indica si para el documento son válidos los reenvíos: 0 (no se permite reenvío), 1 (se permite el reenvío del documento).

Valor por default: 0

doc_are_ins_nom_archivo

Nombre del archivo a importar de manera automática por el CLIENTE, para obtener la información del documento.

doc_are_ins_dias

Numero de días antes de la fecha límite de envío del documento, en que se debe empezar a buscar el archivo a importar.

doc_are_ins_hora

Hora a la que debe empezar a buscar el archivo a importar, en los días en los que se puede realizar la importación antes del fin del periodo.

Valor más alto: 24

Valor más bajo: 0

doc_are_ins_fecha_archivo

Fecha del último archivo que se importó.

doc_are_ins_autom_habil

Atributo que indica si se tiene habilitada la opción de importación automática de archivos o la de acceso remoto a BD para un documento. Un documento no puede ser definido para que la información se reciba por importación de archivos o acceso remoto a BD, solamente es válida una a la vez. Los valores son: 0 (no se tiene habilitada la opción), 1 (se tiene habilitada la opción).

Valor por default: 0

doc_are_ins_integra_habil

Indica si el documento se va a enviar al SA Central para su integración, el objetivo de no enviar al SA central es mantener la información localmente para que las instituciones puedan accederla al momento que lo deseen y si es necesario enviar en forma manual la información. Los posibles valores son: 0 (no se hace envío al SA Central), 1 (se realiza envío al SA Central).

Valor por default: 1

Lista de referencias

Entidad	Cardinalidad	Dependiente	Relación
area_institucion	1,1	Si	are_ins_a_doc_are_ins
documento	1,1	Si	doc_a_doc_are_ins
acceso_bd	0,1	Si	doc_are_ins_a_acc_bd
bitacora_importacion	0,n	No	doc_are_ins_a_bit_imp
retraso_envio	0,1	Si	doc_are_ins_a_ret_env
servidor_aplicativo	0,n	No	envio_a_central
usuario	1,1	No	usu_a_doc_are_ins

Entidad: documento_version

Entidad que mantiene las diferentes versiones de un documento.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
doc_ver_version	smallint	Si	Si
doc_ver_ano_aplicación	smallint	No	No
doc_ver_periodo_aplicación	smallint	No	No
doc_ver_tolerancia	smallint	No	No
doc_ver_filtro	varchar(20)	No	No
doc_ver_separador	char(5)	No	No
doc_ver_separador_col	char(5)	No	No
doc_ver_sintaxis_dw	text(1000)	No	No
doc_ver_authorized	boolean	No	Si
doc_ver_sintaxis_hist	text(2000)	No	No
doc_ver_sintaxis_header	text(2000)	No	No
doc_ver_sintaxis_imp	text(2000)	No	No

doc_ver_version

Número de versión del documento.

doc_ver_ano_aplicacion

Año a partir del cuál aplica la versión del documento.

doc_ver_periodo_aplicacion

Período del año a partir del cuál aplica la versión del documento.

doc_ver_tolerancia

Número de días de tolerancia para la recepción del documento, antes de que se considere como retraso.

doc_ver_filtro

Nombre del *stored procedure* que filtra la información del documento antes de ser enviada al SA Central.

doc_ver_separador

Separador entre las secciones que forman un documento.

doc_ver_separador_col

Separador de las columnas del documento, para cuando la información se recibe por archivos.

doc_ver_sintaxis_dw

Sintaxis para crear el *DataWindow* con el cual se va a capturar la información.

doc_ver_autorizado

Especifica si el documento ha sido autorizado, con lo cual se pone a disposición de las instituciones.

Valor por default: 1

doc_ver_sintaxis_hist

Sintaxis del *DataWindow* para acceder la información histórica del documento.

doc_ver_sintaxis_header

Sintaxis para el *DataWindow* de los valores constantes dentro del documento (*header*).

doc_ver_sintaxis_imp

Atributo en donde se guarda la sintaxis del *DataWindow* para la importación del documento.

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
columnas	1,1	Si	doc_ver_a_col
documento	1,1	Si	doc_ver_a_doc
envio	0,n	No	doc_ver_a_env
formula_integracion	0,n	No	doc_ver_a_for_int
sp_validacion	0,n	No	doc_ver_a_sp_val
formula_validacion	0,n	No	for_val_a_doc_ver

Entidad: envio

Es la entidad principal del *middleware* y en ella se mantiene la información de los envíos. Se define un envío como el conjunto de información recibida para un documento en un periodo. Cuando se recibe información vía captura, Importación de archivos o Acceso a base de datos, se asocia un envío a dicha información.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
env_folio	numeric(9)	Si	Si
env_fecha	datetime	No	Si
env_ano	smallint	No	Si
env_periodo	smallint	No	Si
env_tipo_respuesta	char(1)	No	Si
env_tipo_peticion	char(1)	No	Si
env_tipo_carga	char(1)	No	Si
env_status	char(2)	No	Si
env_notificacion_enviada	boolean	No	Si
env_notificacion_recibida	boolean	No	Si
env_tabla_temporal	varchar(32)	No	No
env_presentar_acuse	boolean	No	Si
env_bitacora_pendiente	boolean	No	Si
env_automatica_disponible	boolean	No	Si
env_filtro	boolean	No	Si
env_tipo_reenvio	char(1)	No	Si

env_folio

Folio del envío, es un número secuencial.

env_fecha

Fecha en la que se generó el envío.

env_ano

Año del período del envío.

env_periodo

Periodo del envío. El periodo es un número que dependiendo de la periodicidad del documento asociado, representa el número de semestre, trimestre, bimestre, mes, semana o día del año. Ejemplo: un período 2 representaría el segundo semestre del año (para periodicidad semestral), la segunda semana del año (para periodicidad semanal), etc.

env_tipo_respuesta

Cuando el *middleware* recibe información mediante alguno de los esquemas de recepción, deberá enviar una respuesta al usuario que generó el envío o al responsable del mismo.

Valor por default: S

env_tipo_peticion

El tipo de servicio solicitado al SA para el envío generado: solamente validación(V), servicio completo de validación e integración(I), la información debe ser filtrada antes de ser enviada al SA central(F).

Valor por default: I

env_tipo_carga

Este atributo indica la forma en como se recibió la información del envío: captura(C), importación manual de archivo(M), importación automática de archivo(A) y acceso a base de datos remota(B).

Valor por default: C

env_status

Estatus del envío.

Valor por default: C

Dominio del atributo: Carga enviada(CE), carga ok(CO), carga(C), validación(V), validación ok(VO), validación fallida(VF), sólo validación(SV), integración(I), integración enviada(IE), integración ok(IO), integración fallida(IF).

env_notificacion_enviada

Este atributo indica si la notificación ya ha sido enviada.

Valor por default: 0

env_notificacion_recibida

Este atributo indica si la notificación enviada ya fue recibida por el usuario responsable.

Valor por default: 0

env_tabla_temporal

Nombre de la tabla temporal en donde se carga la información del envío. El nombre de la tabla temporal se forma de la siguiente manera: ins_nombre_temporal+"_"+doc_nombre_temporal+"_"+env_ano+"_"+env_periodo

env_presentar_acuse

Cuando este atributo esta activado, el acuse de recibo de este envío es presentado en la Lista de ultimos acuses del CLIENTE.

Valor por default: 1

env_bitacora_pendiente

Este atributo indica si los mensajes de bitácora del envío no se han enviado al SA Central.

Valor por default: 0

env_automatica_disponible

Cuando un documento ha sido configurado para que se obtenga mediante importación automática de archivos, con este atributo se puede desactivar la importación si por alguna razón la información se obtuvo mediante otro esquema.

Valor por default: 0

env_filtro

Este atributo indica si la información del envío debe ser filtrado por un *stored procedure* antes de ser enviada al SA Central.

Valor por default: 0

env_tipo_reenvio

Este atributo indica, para los reenvíos el tipo de éste: reemplazo total o envío de nuevos renglones. Los posibles valores son: reenvío total(T), nuevos renglones(R), no aplica(N).

Valor por default: N

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
documento_version	1,1	No	doc_ver_a_env
bitácora	0,1	Si	env_a_bit
status_envio_central	0,1	Si	env_a_sta_env_cen
institucion	1,1	Si	ins_a_env
usuario	1,1	No	usu_a_env

Entidad: formula_integracion

En esta entidad se representan las formulas de integración aplicadas a un documento.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
for_int_id	numeric(9)	Si	Si
for_int_texto	varchar(255)	No	No
for_int_habilitada	boolean	No	Si

for_int_id

Identificador de la fórmula de integración.

for_int_texto

Fórmula que se aplica para la integración.

for_int_habilitada

Este atributo muestra si la fórmula de integración está habilitada o no. habilitada(1), deshabilitada(0).

Valor por default: 0

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
documento_version	1,1	No	doc_ver_a_for_int
columna_operacional	1,n	No	integracion_formulas

Entidad: formula_validacion

En esta entidad se representan las fórmulas de validación aplicadas a un documento. Se

incluyen fórmulas de validación que puedan ser representadas con una expresión aritmético-lógica en donde los operandos sean columnas del documento y validaciones de tipo *if-then-else*. Se consideran fórmulas que se validan a nivel registro o a nivel documento.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
for_val_texto	varchar(255)	No	Si
for_val_texto_compilado	varchar(255)	No	No
for_val_nivel	char(1)	No	Si
for_val_independiente	boolean	No	Si

for_val_texto

En este atributo se almacena la expresión aritmético-lógica asociada a la fórmula. Para el caso de validaciones tipo:

```
if expresion_1 then
  expresion_2
else
  expresion_3
```

este atributo almacena el texto de expresion_1.

for_val_texto_compilado

Este atributo almacena la fórmula ya compilada.

for_val_nivel

Este atributo indica si la validación es a nivel registro(R), a nivel de todos los registros del envío(E).

Valor por default: R

for_val_independiente

Este atributo indica si la fórmula es independiente, lo cuál implica que se debe de validar no importando que esté presente en expresiones *if-then-else*, en caso contrario, la fórmula solamente se debe evaluar como parte de *if-then-else*. Para el caso de expresiones *if-then-else*, si la expresión tiene habilitado este atributo, esto indica que es un *if* de primer nivel (no forma parte de un *if-then-else* externo).

Valor por default: 1

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
if_then_else	0,n	No	else
documento_version	1,1	No	for_val_a_doc_ver
if_then_else	0,n	No	if
if_then_else	0,n	No	then

Entidad: grupo

Los documentos que se reciben por medio del *middleware* están clasificados en grupos, como contabilidad, administración, operativos, etc.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
grupo_id	numeric(4)	Si	Si
grupo_descripcion	varchar(50)	No	No

grupo_id

Identificador del grupo.

grupo_descripcion

Descripción del grupo.

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
documento	0,n	No	grupo_documento

Entidad: if_then_else

Subentidad que representa a las fórmulas de validación de tipo *if-then-else*. Este tipo de validaciones pueden ser representadas como:

```
if expresion_1 then
    expresion_2
else
    expresion_3
```

donde *expresion_1*, *expresion_2* y *expresion_3* son expresiones aritmético-lógicas (representadas en la entidad *formula_validacion*), incluso *expresion_2* y *expresion_3* pueden ser validaciones de tipo *if-then-else* (también representadas en la entidad *formula_validacion*). Lo importante de esta entidad es que mantiene tres relaciones con la entidad *formula_validacion*, las cuales representan a *expresion_1*, *expresion_2* y *expresion_3*.

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
formula_validacion	0,1	No	else
formula_validacion	1,1	No	if
formula_validacion	0,1	No	then

Entidad: institucion

Esta entidad mantiene el catálogo de las Entidades Supervisadas que pueden enviar información al *middleware*.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
ins_sector	smallint	Si	Si
ins_clave	integer	Si	Si
ins_nombre	varchar(50)	No	No
ins_nombre_corto	varchar(20)	No	No
ins_nombre_temporal	char(10)	No	Si

ins_sector

Clave del sector de la Entidad Supervisada. Es un número asignado por el área de sistemas en las bases de datos operacionales de la Entidad Normativa.

ins_clave

Es un número secuencial asignado por el área de sistemas que identifica a una Entidad Supervisada en el sector al que corresponde.

ins_nombre

Nombre completo de la Entidad Supervisada.

ins_nombre_corto

Nombre corto de la Entidad Supervisada.

ins_nombre_temporal

Es el nombre a utilizar para la creación de la tabla temporal (ins_nombre_temporal+"_"+doc_nombre_temporal+"_"+ano+"_"+periodo)

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
area_institucion	0,1	Si	ins_a_are_ins
envio	0,1	Si	ins_a_env

Entidad: mensaje

Entidad que representa el catálogo de mensajes, errores o *warnings* asociados a los procesos de recepción, validación e integración de documentos.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
men_numero	int	Si	Si
men_severity	smallint	No	Si
men_tipo	char(1)	No	Si
men_texto	varchar(40)	No	No

men_numero

Número de mensaje.

men_severity

Severidad asociada al mensaje. La severidad indica la naturaleza del mensaje.

men_tipo

Tipo de mensaje: Error (E), *Warning* (W), Informativo(I).

Valor por default: E

men_texto

Texto asociado al mensaje.

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
bitácora	0,n	No	men_a_bit

Entidad: mensaje_error_val

En esta entidad se mantiene información de los mensajes de error asociados a las validaciones de documentos.

Lista de atributos

Nombre	Tipo	PK	Obligatorio
men_err_val_error	char(7)	Si	Si
men_err_val_texto	varchar(100)	No	No

men_err_val_error

Identificador del error.

men_err_val_texto

Texto o descripción del error.

Lista de referencia

Entidad	Cardinalidad	Dependiente	Relación
validación	0,n	No	men_err_val_a_val

Entidad: retraso_envio

Entidad que mantiene información respecto a los documentos que no se recibieron a tiempo.

Lista de Atributos

Nombre	Tipo	PK	Obligatorio
et_env_ano	smallint	Si	Si
et_env_periodo	smallint	Si	Si
et_env_fecha	Date	No	No
et_env_notificacion_enviada	booleano	No	Si
et_env_notificacion_recibida	booleano	No	Si

ret_env_eno

Año asociado al periodo del documento.

ret_env_periodo

Periodo en el que se dio el retraso en la recepción del documento.

ret_env_fecha

Fecha en la que se detecto el retraso.

ret_env_notificacion_enviada

Atributo que indica si se envió notificación de retraso al usuario responsable del envío del documento.

ret_env_recibida

Con este atributo se indica si el responsable de enviar el documento recibió la notificación de retraso.

Lista de Referencias

Entidad	Cardinalidad	Dependiente	Relación
documento_area_institucion	1,1	Si	doc_are_ins_a_ret_env

Entidad: servidor_aplicativo

Esta entidad mantiene la información de los Servidores Aplicativos que serán tratados como centrales.

Lista de Atributos

Nombre	Tipo	PK	Obligatorio
ser_app_id	numerico(8)	Si	Si
ser_app_nombre	varchar(15)	No	Si
ser_app_login	varchar(15)	No	Si
ser_app_password	varchar(15)	No	Si
ser_app_sql_server	varchar(15)	No	Si
ser_app_sql_server_login	varchar(15)	No	Si
ser_app_sql_server_pwd	varchar(15)	No	Si

ser_app_id

Identificador del Servidor Aplicativo.

ser_app_nombre

Nombre del Servidor Aplicativo.

ser_app_login

Login utilizado para acceder el SA.

ser_app_password

Password asociado al login utilizado para acceder el SA.

ser_app_server

Nombre del *SQL Server* asociado al SA y que controla la base de datos de *metadata*.

ser_app_server_login

Login utilizado para acceder el *SQL Server*.

ser_app_server_pwd

Password asociado al login del *SQL Server*.

Lista de Referencias

Entidad	Cardinalidad	Dependiente.	Relaciones
documento_area_institucion	0,n	No	Envio_a_central
status_envio_central	0,1	Si	ser_apl_a_sta_env_cen

Entidad: servidor_produccion

Repositorio Central en donde se puede integrar información proveniente de los documentos recibidos.

Lista de Atributos

Nombre	Tipo	PK	Obligatorio
srv_id	smallint	Si	Si
srv_nombre	varchar(15)	No	Si
srv_login	varchar(15)	No	Si
srv_password	varchar(15)	No	Si

srv_id

Identificador del repositorio central.

srv_nombres

Nombre del repositorio central.

srv_login

Login utilizado para acceder el servidor del repositorio central.

srv_password

Password asociado al *login*.

Lista de Referencias

Entidad	Cardinalidad	Dependiente	Relaciones
tabla_operacional	0,n	No	ser_pro_a_tab_ope

Entidad: sp_validacion

Entidad que mantiene la información de los *stored procedures* que realizan los procesos de validación complejas en la información de los documentos que se reciben.

Lista de Atributos

Nombre	Tipo	PK	Obligatorio
sp_orden	smallint	No	Si
sp_bd	varchar(15)	No	Si
sp_store_proc	varchar(15)	No	Si

sp_orden

Número de secuencia en la que se debe aplicar la validación del *stored procedure*.

sp_db

Base de Datos en donde reside el *stored procedure*.

sp_store_proc

Nombre del *stored procedure*.

Lista de Referencias

Entidad	Cardinalidad	Dependiente	Relaciones
documento_version	1,1	No	doc_ver_a_sp_val

Entidad: status_envio_central

Esta entidad mantiene información referente al estatus de un envío. Principalmente se mantiene el estatus de envío al SA Central.

Lista de Atributos

Nombre	Tipo	PK	Obligatorio
sta_env_cen_status	char(2)	No	Si
sta_env_cen_status_bita	char(2)	No	Si

sta_env_cen_status

Estatus del envío. Integración pendiente(IP), integración enviada al SA Central(IE).

sta_env_cen_status_bita

Estatus de la bitácora asociada al envío. Bitácora Pendiente(BP) y Bitácora enviada al SA Central(BE).

Lista de Referencias

Entidad	Cardinalidad	Dependiente	Relaciones
envio	1,1	Si	env_a_sta_env_cen
servidor_aplicativo	1,1	Si	ser_apl_a_sta_env_cen

Entidad: tabla_operacional

Entidad que mantiene información de tablas en los repositorios centrales en donde es factible integrar información proveniente de los documentos recibidos.

Lista de Atributos

Nombre	Tipo	PK	Obligatorio
tab_ope_id	integer	Si	Si
tab_ope_bd	varchar(15)	No	Si
tab_ope_tabla	varchar(25)	No	Si

tab_ope_id

Identificador de la tabla. Este identificador es asignado secuencialmente.

tab_ope_db

Nombre de la base de datos en donde reside la tabla.

tab_ope_tabla

Nombre de la tabla.

Lista de Referencias

Entidad	Cardinalidad	Dependent e	Relaciones
servidor_produccion	1,1	No	ser_pro_a_tab_ope
columna_operacional	1,1	Si	tab_ope_a_col_ope

Entidad: Usuario

Entidad que representa a los usuarios de cada una de las Entidades Supervisadas. Se considera que los usuarios son los que utilizan el CLIENTE para el envío de documentos y para obtener información respecto a como fueron procesados los documentos enviados. La única forma de utilizar el CLIENTE es con un usuario válido para la institución en donde reside dicha aplicación.

Lista de Atributos

Nombre	Tipo	PK	Obligatorio
usu_id	char(8)	Si	Si
usu_password	varchar(10)	No	No
usu_nombre	varchar(40)	No	No
usu_telefono	varchar(15)	No	No

usu_id

Identificador del usuario asignado por la Entidad Normativa. Este identificador es el *login* que utilizan los usuarios del SA.

usu_password

Password que utilizará el usuario para acceder el SA.

usu_nombre

Nombre completo del usuario.

usu_telefono

Teléfono del usuario.

Lista de Referencias

Entidad	Cardinalidad	Dependiente	Relaciones
area_institucion	1,1	No	are_ins_a_usu
documento_area_institucion	0,n	No	usu_a_doc_are_ins
envio	0,n	No	usu_a_env

Entidad: validacion

Entidad que mantiene información de las reglas de validación aplicadas a los documentos. Esta entidad es una superentidad que agrupa a las entidades: sp_validación, formula_validacion y validacion_catalogo.

Lista de Atributos

Nombre	Tipo	PK	Obligatorio
val_id	numerico(9)	Si	Si
val_habilitada	booleano	No	Si

val_id

Identificador de la validación.

val_habilitada

Atributo booleano que indica si la validación está habilitada(1) o deshabilitada(0).

Valor de default: 0

Lista de Referencias

Entidad	Cardinalidad	Dependiente	Relaciones
mensaje_error_val	0,1	No	men_err_val_a_val

Entidad: validacion_catalogo

Entidad que mantiene las validaciones contra catálogo. Las validaciones contra catálogo son a nivel columna y pueden incluir una o varias columnas, es decir se puede validar una columna del documento contra una columna de un catálogo o bien dos columnas del documento contra dos columnas de un catálogo (tomando en cuenta la combinación de los valores que tienen las columnas del documento).

Lista de Referencias

Entidad	Cardinalidad	Dependiente	Relaciones
columnas	1,1	Si	col_a_val_cat
columna_catalogo	1,1	Si	col_cat_a_val_cat

Relaciones de las tablas

Ahora veremos una descripción técnica sobre las relaciones que existen entre las tablas de la metadata, en la descripción se identifica primero a la relación, el identificador se compone de las tres primeras letras del nombre de las tablas relacionadas, y la conjunción "a"; al mismo tiempo éste le da nombre a la relación. En la descripción de las relaciones incluimos el nombre de la misma, el nombre de las entidades involucradas, indicamos su cardinalidad; y los caracteres de obligatoriedad y dominancia en cada sentido de la relación.

Relación are_a_are_ins

Nombre: are_a_are_ins
Entidad 1: area
Entidad 2: area_institucion
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación are_ins_a_doc_are_ins

Nombre: are_ins_a_doc_are_ins
Código: are_ins_a_doc_are_ins
Entidad 1: area_institucion
Entidad 2: documento_area_institucion
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación are_ins_a_usu

Nombre: are_ins_a_usu
Código: are_ins_a_usu
Entidad 1: area_institucion

Entidad 2: usuario
 Cardinalidad: uno a muchos
 Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación cat_a_col_cat

Nombre: cat_a_col_cat
 Código: cat_a_col_cat
 Entidad 1: catalogo
 Entidad 2: columna_catalogo
 Cardinalidad: uno a muchos
 Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación col_a_val_cat

Nombre: col_a_val_cat
 Código: col_a_val_cat
 Entidad 1: columnas
 Entidad 2: validacion_catalogo
 Cardinalidad: uno a muchos
 Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación col_cat_a_dic_ddd

Nombre: col_cat_a_dic_ddd
 Código: col_cat_a_dic_ddd
 Entidad 1: columna_catalogo
 Entidad 2: diccionario_dddw
 Cardinalidad: uno a muchos
 Entidad 2 dependiente de Entidad 1: No

Entidad 1 → Entidad 2:		Entidad 2 → Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación col_cat_a_val_cat

Nombre: col_cat_a_val_cat
Código: col_cat_a_val_cat
Entidad 1: columna_catalogo
Entidad 2: validacion_catalogo
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 → Entidad 2:		Entidad 2 → Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación dic_ddd_a_col

Nombre: dic_ddd_a_col
Código: dic_ddd_a_col
Entidad 1: diccionario_dddw
Entidad 2: columnas
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 → Entidad 2:		Entidad 2 → Entidad 1:	
Obligada:	No	Obligada:	No
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	0, 1

Relación dic_ddd_a_col_cat

Nombre: dic_ddd_a_col_cat
Código: dic_ddd_a_col_cat
Entidad 1: columna_catalogo
Entidad 2: diccionario_dddw
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 → Entidad 2:		Entidad 2 → Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Entidad 2: usuario
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación cat_a_col_cat

Nombre: cat_a_col_cat
Código: cat_a_col_cat
Entidad 1: catalogo
Entidad 2: columna_catalogo
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación col_a_val_cat

Nombre: col_a_val_cat
Código: col_a_val_cat
Entidad 1: columnas
Entidad 2: validacion_catalogo
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación col_cat_a_dic_ddd

Nombre: col_cat_a_dic_ddd
Código: col_cat_a_dic_ddd
Entidad 1: columna_catalogo
Entidad 2: diccionario_dddw
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 → Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación col_cat_a_val_cat

Nombre: col_cat_a_val_cat
Código: col_cat_a_val_cat
Entidad 1: columna_catalogo
Entidad 2: validacion_catalogo
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación dic_ddd_a_col

Nombre: dic_ddd_a_col
Código: dic_ddd_a_col
Entidad 1: diccionario_dddw
Entidad 2: columnas
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	No
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	0, 1

Relación dic_ddd_a_col_cat

Nombre: dic_ddd_a_col_cat
Código: dic_ddd_a_col_cat
Entidad 1: columna_catalogo
Entidad 2: diccionario_dddw
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación **doc_a_doc_are_ins**

Nombre: doc_a_doc_are_ins
Código: doc_a_doc_are_ins
Entidad 1: documento
Entidad 2: documento_area_institucion
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación **doc_are_ins_a_acc_bd**

Nombre: doc_are_ins_a_acc_bd
Código: doc_are_ins_a_acc_bd
Entidad 1: documento_area_institucion
Entidad 2: acceso_bd
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	Si	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación **doc_are_ins_a_bit_imp**

Nombre: doc_are_ins_a_bit_imp
Código: doc_are_ins_a_bit_imp
Entidad 1: documento_area_institucion
Entidad 2: bitacora_importacion
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación **doc_are_ins_a_ret_env**

Nombre: doc_are_ins_a_ret_env
Código: doc_are_ins_a_ret_env
Entidad 1: documento_area_institucion

Entidad 2: retraso_envio
 Cardinalidad: uno a muchos
 Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación doc_ver_a_col

Nombre: doc_ver_a_col
 Código: doc_ver_a_col
 Entidad 1: documento_version
 Entidad 2: columnas
 Cardinalidad: uno a muchos
 Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	Sí	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	1, 1	Min, Max:	1, 1

Relación doc_ver_a_doc

Nombre: doc_ver_a_doc
 Código: doc_ver_a_doc
 Entidad 1: documento
 Entidad 2: documento_version
 Cardinalidad: uno a muchos
 Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	Sí	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	1, 1	Min, Max:	1, 1

Relación doc_ver_a_env

Nombre: doc_ver_a_env
 Código: doc_ver_a_env
 Entidad 1: documento_version
 Entidad 2: envio
 Cardinalidad: uno a muchos
 Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación **doc_ver_a_for_int**

Nombre: doc_ver_a_for_int
Código: doc_ver_a_for_int
Entidad 1: documento_version
Entidad 2: formula_integracion
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación **doc_ver_a_sp_val**

Nombre: doc_ver_a_sp_val
Código: doc_ver_a_sp_val
Entidad 1: documento_version
Entidad 2: sp_validacion
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación **else**

Nombre: else
Código: else
Entidad 1: formula_validacion
Entidad 2: if_then_else
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	No
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	0, 1

Descripción

En esta relación se mantiene la asociación de la fórmula de tipo if-then-else con expresion_3 considerando que la fórmula if_then_else sigue el siguiente formato:

```
if expresion_1 then
  expresion_2
else
  expresion_3
```

y que expresion_3 es una instancia de formula_validacion.

Relación env_a_bit

Nombre: env_a_bit
Código: env_a_bit
Entidad 1: envio
Entidad 2: bitacora
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación env_a_sta_env_cen

Nombre: env_a_sta_env_cen
Código: env_a_sta_env_cen
Entidad 1: envio
Entidad 2: status_envio_central
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Si

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación envio_a_central

Nombre: envio_a_central
Código: envio_a_central
Entidad 1: servidor_aplicativo
Entidad 2: documento_area_institucion
Cardinalidad: muchos a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	No
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	0, n

Relación for_val_a_doc_ver

Nombre: for_val_a_doc_ver
Código: for_val_a_doc_ver
Entidad 1: documento_version
Entidad 2: formula_validacion
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación grupo_documento

Nombre: grupo_documento
Código: grupo_documento
Entidad 1: grupo
Entidad 2: documento
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación if

Nombre: if
Código: if
Entidad 1: formula_validacion
Entidad 2: if_then_else
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Descripción

En esta relación se mantiene la asociación de la fórmula de tipo *if-then-else* con expresion_1 considerando que la fórmula *if_then_else* sigue el siguiente formato:

```
if expresion_1 then
  expresion_2
else
  expresion_3
```

y que expresion_1 es una instancia de formula_validacion.

Relación ins_a_are_ins

Nombre: ins_a_are_ins
Código: ins_a_are_ins
Entidad 1: institucion
Entidad 2: area_institucion
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 -> Entidad 2:	Entidad 2 --> Entidad 1:
Obligada: No	Obligada: Sí
Dominante: No	Dominante: No
Min, Max: 0, 1	Min, Max: 1, 1

Relación ins_a_env

Nombre: ins_a_env
Código: ins_a_env
Entidad 1: institucion
Entidad 2: envio
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 -> Entidad 2:	Entidad 2 --> Entidad 1:
Obligada: No	Obligada: Sí
Dominante: No	Dominante: No
Min, Max: 0, 1	Min, Max: 1, 1

Relación integracion_columnas

Nombre: integracion_columnas
Código: integracion_columnas
Entidad 1: columnas
Entidad 2: columna_operacional
Cardinalidad: muchos a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	Sí	Obligada:	No
Dominante:	No	Dominante:	No
Min, Max:	1, n	Min, Max:	0, n

Relación **integracion_formulas**

Nombre: integracion_formulas
Código: integracion_formulas
Entidad 1: columna_operacional
Entidad 2: formula_integracion
Cardinalidad: muchos a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, n

Relación **men_a_bit**

Nombre: men_a_bit
Código: men_a_bit
Entidad 1: mensaje
Entidad 2: bitacora
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	No
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	0, 1

Relación **men_err_val_a_val**

Nombre: men_err_val_a_val
Código: men_err_val_a_val
Entidad 1: mensaje_error_val
Entidad 2: validacion
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	No
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	0, 1

Relación ser_apl_a_sta_env_cen

Nombre: ser_apl_a_sta_env_cen
Código: ser_apl_a_sta_env_cen
Entidad 1: servidor_aplicativo
Entidad 2: status_envio_central
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, 1	Min, Max:	1, 1

Relación ser_pro_a_tab_ope

Nombre: ser_pro_a_tab_ope
Código: ser_pro_a_tab_ope
Entidad 1: servidor_produccion
Entidad 2: tabla_operacional
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación tab_ope_a_col_ope

Nombre: tab_ope_a_col_ope
Código: tab_ope_a_col_ope
Entidad 1: tabla_operacional
Entidad 2: columna_operacional
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: Sí

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	Sí	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	1, 1	Min, Max:	1, 1

Relación then

Nombre: then
Código: then
Entidad 1: formula_validacion
Entidad 2: if_then_else

Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	No
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	0, 1

Descripción

En esta relación se mantiene la asociación de la fórmula de tipo *if-then-else* con expresion_2 considerando que la fórmula *if_then_else* sigue el siguiente formato:

```
if expresion_1 then
  expresion_2
else
  expresion_3
```

y que expresion_2 es una instancia de formula_validacion.

Relación usu_a_doc_are_ins

Nombre: usu_a_doc_are_ins
Código: usu_a_doc_are_ins
Entidad 1: usuario
Entidad 2: documento_area_institucion
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

Relación usu_a_env

Nombre: usu_a_env
Código: usu_a_env
Entidad 1: usuario
Entidad 2: envio
Cardinalidad: uno a muchos
Entidad 2 dependiente de Entidad 1: No

Entidad 1 --> Entidad 2:		Entidad 2 --> Entidad 1:	
Obligada:	No	Obligada:	Sí
Dominante:	No	Dominante:	No
Min, Max:	0, n	Min, Max:	1, 1

6.2 Modelo Físico

Implementar una Base de Datos es transformar el Modelo Lógico de la misma en una serie de estructuras físicas que la representen. El Diseño Físico es el proceso de determinar las estructuras de datos y mecanismos a ser utilizados para la implementación de la Base de Datos.

El Diseño Físico de la Base de Datos es particular a un DBMS.

6.2.1 Diseño Físico de la metadata

En la figura 6.1 presentamos el diseño físico de la metadata que se generó a partir del diseño conceptual, para un DBMS de Sybase System 11.0.

A continuación se muestran las siguientes tablas que se generaron del diseño conceptual de la *metadatos*.

Tabla: acceso_bd

Generada de la entidad acceso_bd.

Columnas	Tipo	PK	M
ins_sector	smallint	Si	Si
ins_clave	int	Si	Si
area_id	smallint	Si	Si
doc_id	int	Si	Si
acc_bd_objeto	varchar(15)	No	Si
acc_bd_tipo_objeto	char(1)	No	Si
acc_bd_server	varchar(15)	No	Si
acc_bd_bd	varchar(15)	No	No
acc_bd_login	varchar(15)	No	Si
acc_bd_password	varchar(15)	No	No
acc_bd_dias_anticipo	smallint	No	No
acc_bd_dias	char(7)	No	Si
acc_bd_habilitada	smallint	No	No
acc_bd_hora_ini	smallint	No	No
acc_bd_hora_fin	smallint	No	No

Indice	PK	FK	Columnas
acceso_bd_pk	Si	Si	ins_sector, ins_clave area_id, doc_id

Lista de relaciones

Llave Primaria	Referencia	Llave Foránea
ins_sector ins_clave area_id doc_id documento_area_institucion	documento_area_institucion	ins_sector ins_clave area_id doc_id acceso_bd

Tabla: area

Generada por la entidad area

Columnas	Tipo	PK	M
area_id	smallint	Si	Si
area_nombre	varchar(40)	No	No

Indice	PK	FK	Columnas
area_pk	Si	No	area_id

Lista de relaciones

Relacionada por	Llave Primaria	Llave Foránea
area_institucion	area_id area	area_id area_institucion

Tabla: area_institucion

Generada por la entidad area_institucion

Columnas	Tipo	PK	M
ins_sector	smallint	Si	Si
ins_clave	int	Si	Si
are_id	smallint	Si	Si

Indice	PK	FK	Columnas
area_institucion_pk	Si	No	ins_sector ins_clave area_id
are_a_are_ins2_fk	No	Si	area_id
ins_a_are_ins2_fk	No	Si	ins_sector ins_clave

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
area_id area ins_sector ins_clave institucion	area institucion	area_id area_institucion ins_sector ins_clave area_institucion

Relacionada por	Llave Primaria	Llave Foránea
documento_area_institucion usuario	ins_sector ins_clave area_id area_institucion ins_sector ins_clave area_id area_institucion	ins_sector ins_clave area_id documento_area_institucion ins_sector ins_clave area_id usuario

Tabla: bitacora

Generada por la entidad bitacora

Columnas	Tipo	PK	M
ins_sector	smallint	Si	Si
ins_clave	int	Si	Si
env_folio	numeric(9)	Si	Si
men_numero	int	No	No
bit_fase	varchar(12)	No	No
bit_mensaje	varchar(80)	No	No
bit_numero_registro	smallint	No	Si
bit_numero_columna	smallint	No	Si
bit_fecha	datetime	No	No

Indice	PK	FK	Columnas
bitacora_pk	Si	Si	ins_sector ins_clave env_folio
men_a_bit_fk	No	Si	men_numero

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
ins_sector ins_clave env_folio envio	envio	ins_sector ins_clave env_folio bitacora
men_numero mensaje	mensaje	men_numero bitacora

Tabla: bitacora_importacion

Generada por la entidad bitacora_importacion

Columnas	Tipo	PK	M
bit_imp_secuencia	numeric(10)	Si	Si
ins_sector	smallint	No	Si
ins_clave	int	No	Si
are_id	smallint	No	Si
doc_id	int	No	Si
bit_imp_ano	smallint	No	Si
bit_imp_periodo	smallint	No	Si
bit_imp_tipo	char(1)	No	Si
bit_imp_mensaje	varchar(80)	No	No
bit_imp_fecha	datetime	No	No

Indice	PK	FK	Columnas
bitacora_importacion_pk	Si	No	bit_imp_secuencia
doc_are_ins_a_bit_imp_fk	No	Si	ins_sector ins_clave area_id doc_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
ins_sector ins_clave area_id doc_id	documento_area_institucion	ins_sector ins_clave area_id doc_id bitacora_importacion
documento_area_institucion		

Tabla: catalogo

Generada por la entidad catalogo

Columnas	Tipo	PK	M
cat_id	int	Si	Si
cat_nombre	varchar(20)	No	Si
cat_descripcion	varchar(50)	No	No

Indice	PK	FK	Columnas
catalogo_pk	Si	No	cat_id

Lista de relaciones

Relacionada por	Llave Primaria	Llave Foránea
columna_catalogo	cat_id catalogo	cat_id columna_catalogo

Tabla: columna_catalogo

Generada por la entidad columna_catalogo

Columnas	Tipo	P	M
cat_id	Int	Si	Si
col_cat_orden	smallint	Si	Si
col_cat_nombre	varchar(15)	No	No
col_cat_encabezado	varchar(20)	No	No
col_cat_tipo	char(10)	No	No
col_cat_longitud	smallint	No	No
col_cat_formato	varchar(20)	No	No
col_cat_longitud_despliegue	smallint	No	No

Indice	PK	FK	Columnas
columna_catalogo_pk	Si	No	cat_id col_cat_orden
cat_a_col_cat2_fk	No	Si	cat_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
cat_id catalogo	catalogo	cat_id columna_catalogo

Relacionada por	Llave Primaria	Llave Foránea
diccionario_dddw	cat_id col_cat_orden columna_catalogo	col_cat_id col_col_cat_orden diccionario_dddw
validacion_catalogo	cat_id col_cat_orden columna_catalogo	cat_id col_cat_orden validacion_catalogo

Tabla: columna_operacional

Generada por la entidad columna_operacional

Columnas	Tipo	PK	M
tab_ope_id	int	Si	Si
col_ope_id	smallint	Si	Si
col_ope_columna	varchar(20)	No	Si
col_ope_llave	bit	No	Si

Indice	PK	FK	Columnas
columna_operacional_pk	Si	No	tab_ope_id, col_ope_id
tab_ope_a_col_ope2_fk	No	Si	tab_ope_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
tab_ope_id tabla_operacional	tabla_operacional	tab_ope_id columna_operacional

Relacionada por	Llave Primaria	Llave Foránea
integracion_columnas	tab_ope_id col_ope_id columna_operacional	tab_ope_id col_ope_id integracion_columnas
integracion_formulas	tab_ope_id col_ope_id columna_operacional	tab_ope_id col_ope_id integracion_formulas

Tabla: columnas

Generada por la entidad columnas

Columnas	Tipo	PK	M
doc_id	int	Si	Si
doc_ver_version	smallint	Si	Si
col_orden	smallint	Si	Si
dic_dddw_id	int	No	No
col_nombre	varchar(60)	No	No
col_tipo	char(10)	No	Si
col_longitud	smallint	No	Si
col_decimales	smallint	No	Si
col_formato_captura	varchar(100)	No	No
col_longitud_despliegue	smallint	No	No
col_header	bit	No	Si

Indice	PK	FK	Columnas
columnas_pk	Si	No	doc_id doc_ver_version col_orden
doc_ver_a_col2_fk	No	Si	doc_id doc_ver_version
dic_ddd_a_col_fk	No	Si	dic_dddw_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
dic_dddw_id diccionario_dddw	diccionario_dddw	dic_dddw_id columnas
doc_id doc_ver_version	documento_version	doc_id doc_ver_version
documento_version		columnas

Relacionada por	Llave Primaria	Llave Foránea
validacion_catalogo	doc_id doc_ver_version col_orden columnas	doc_id doc_ver_version col_orden validacion_catalogo
integracion_columnas	doc_id doc_ver_version col_orden columnas	doc_id doc_ver_version col_orden integracion_columnas

Tabla: diccionario_dddw

Generada por la entidad diccionario_dddw

Columnas	Tipo	PK	M
dic_dddw_id	int	Si	Si
col_cat_id	int	No	Si
col_col_cat_orden	smallint	No	Si
dic_dddw	text	No	No
dic_dddw_nombre	varchar(20)	No	No

Indice	PK	FK	Columnas
diccionario_dddw_pk	Si	No	dic_dddw_id
col_cat_a_dic_ddd_fk	No	Si	col_cat_id col_col_cat_orden

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
cat_id col_cat_orden columna_catalogo	columna_catalogo	col_cat_id col_col_cat_orden diccionario_dddw

Relacionada por	Llave Primaria	Llave Foránea
columnas	dic_dddw_id diccionario_dddw	dic_dddw_id columnas

Tabla: documento

Generada por la entidad documento

Columnas	Tipo	PK	M
doc_id	int	Si	Si
doc_descripcion	varchar(40)	No	No
doc_periodicidad	char(1)	No	Si
doc_nombre_temporal	char(10)	No	Si
doc_tolerancia	smallint	No	No
doc_cuadro_nulo	bit	No	Si
doc_tipo	char(1)	No	Si
grupo_id	numeric(4)	No	Si

Indice	PK	FK	Columnas
documento_pk	Si	No	doc_id
grupo_documento_fk	No	Si	grupo_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
grupo_id grupo	grupo	grupo_id documento

Relacionada por	Llave Primaria	Llave Foránea
documento_area_institucion	doc_id documento	doc_id documento_area_institucion
documento_version	doc_id documento	doc_id documento_version

Tabla: documento_area_institucion

Generada por la entidad documento_area_institucion

Columnas	Tipo	PK	M
ins_sector	smallint	Si	Si
ins_clave	int	Si	Si
are_id	smallint	Si	Si
doc_id	int	Si	Si
usu_id	char(8)	No	Si
doc_are_ins_reenvio	bit	No	Si
doc_are_ins_nom_archivo	varchar(70)	No	No
doc_are_ins_dias	smallint	No	No
doc_are_ins_hora	datetime	No	No
doc_are_ins_fecha_archivo	varchar(50)	No	No
doc_are_ins_autom_habil	bit	No	Si
doc_are_ins_integra_habil	bit	No	Si

Indice	PK	FK	Columnas
documento_area_institucion_pk	Si	No	ins_sector ins_clave area_id doc_id
are_ins_a_doc_are_ins2_fk	No	Si	ins_sector ins_clave area_id
doc_a_doc_are_ins2_fk	No	Si	doc_id
usu_a_doc_are_ins_fk	No	Si	usu_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
ins_sector ins_clave area_id area_institucion	area_institucion	ins_sector ins_clave area_id documento_area_institucion
doc_id documento	documento	doc_id documento_area_institucion
usu_id usuario	usuario	usu_id documento_area_institucion

Relacionada por	Llave Primaria	Llave Foránea
acceso_bd	ins_sector ins_clave area_id doc_id documento_area_institucion	ins_sector ins_clave area_id doc_id acceso_bd
bitacora_importacion	ins_sector ins_clave area_id doc_id documento_area_institucion	ins_sector ins_clave area_id doc_id bitacora_importacion
retraso_envio	ins_sector ins_clave area_id doc_id documento_area_institucion	ins_sector ins_clave area_id doc_id retraso_envio
envio_a_central	ins_sector ins_clave area_id doc_id documento_area_institucion	ins_sector ins_clave area_id doc_id envio_a_central

Tabla: documento_version

Generada por la entidad documento_version

Columnas	Tipo	PK	M
doc_id	int	Si	Si
doc_ver_version	smallint	Si	Si
doc_ver_ano_aplicacion	smallint	No	No
doc_ver_periodo_aplicacion	smallint	No	No
doc_ver_tolerancia	smallint	No	No
doc_ver_filtro	varchar(20)	No	No
doc_ver_separador	char(5)	No	No
doc_ver_separador_col	char(5)	No	No
doc_ver_sintaxis_dw	text	No	No
doc_ver_authorized	bit	No	Si
doc_ver_sintaxis_hist	text	No	No
doc_ver_sintaxis_header	text	No	No
doc_ver_sintaxis_imp	text	No	No

Indice	PK	FK	Columnas
documento_version_pk	Si	No	doc_id doc_ver_version
doc_ver_a_doc2_fk	No	Si	doc_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
doc_id documento	documento	doc_id documento_version

Relacionada por	Llave Primaria	Llave Foránea
columnas	doc_id doc_ver_version documento_version	doc_id doc_ver_version columnas

Relacionada por	Llave Primaria	Llave Foránea
envio	doc_id doc_ver_version documento_version	doc_id doc_ver_version envio
formula_integracion	doc_id doc_ver_version documento_version	doc_id doc_ver_version formula_integracion
sp_validacion	doc_id doc_ver_version documento_version	doc_id doc_ver_version sp_validacion
formula_validacion	doc_id doc_ver_version documento_version	doc_id doc_ver_version formula_validacion

Tabla: envio

Generada por la entidad de envío

Columnas	Tipo	PK	M
ins_sector	smallint	Si	Si
ins_clave	int	Si	Si
env_folio	numeric(9)	Si	Si
usu_id	char(8)	No	Si
doc_id	int	No	Si
doc_ver_version	smallint	No	Si
env_fecha	datetime	No	Si
env_ano	smallint	No	Si
env_periodo	smallint	No	Si
env_tipo_respuesta	char(1)	No	Si
env_tipo_peticion	char(1)	No	Si
env_tipo_carga	char(1)	No	Si
env_status	char(2)	No	Si
env_notificacion_enviada	bit	No	Si
env_notificacion_recibida	bit	No	Si
env_tabla_temporal	varchar(32)	No	No
env_presentar_acuse	bit	No	Si
env_bitacora_pendiente	bit	No	Si
env_automatica_disponible	bit	No	Si
env_filtro	bit	No	Si
env_tipo_reenvio	char(1)	No	Si

Indice	PK	FK	Columnas
envio_pk	Si	No	ins_sector ins_clave env_folio
usu_a_env_fk	No	Si	usu_id
doc_ver_a_env_fk	No	Si	doc_id doc_ver_version
ins_a_env2_fk	No	Si	ins_sector ins_clave

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
doc_id doc_ver_version documento_version	documento_version	doc_id doc_ver_version envio
ins_sector ins_clave institucion usu_id usuario	institucion usuario	ins_sector ins_clave envio usu_id envio

Relacionada por	Llave Primaria	Llave Foránea
bitacora status_envio_central	ins_sector ins_clave env_folio envio ins_sector ins_clave env_folio envio	ins_sector ins_clave env_folio bitacora ins_sector ins_clave env_folio status_envio_central

Tabla: envio_a_central

Generada por la relación envio_a_central

Columnas	Tipo	PK	M
ser_app_id	numeric(8)	Si	Si
ins_sector	smallint	Si	Si
ins_clave	int	Si	Si
are_id	smallint	Si	Si
doc_id	int	Si	Si

Indice	PK	FK	Columnas
envio_a_central_pk	Si	No	ser_app_id ins_sector ins_clave area_id doc_id
envio_a_central2_fk	No	Si	ser_app_id
envio_a_central_fk	No	Si	ins_sector ins_clave area_id doc_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
ins_sector ins_clave area_id doc_id documento_area_institucion ser_app_id servidor_aplicativo	documento_area_institucion servidor_aplicativo	ins_sector ins_clave area_id doc_id envio_a_central ser_app_id envio_a_central

Tabla: formula_integracion

Generada por la entidad formula_integración

Columnas	Tipo	PK	M
for_int_id	numeric(9)	Si	Si
doc_id	int	No	Si
doc_ver_version	smallint	No	Si
for_int_texto	varchar(255)	No	No
for_int_habilitada	bit	No	Si

Indice	PK	FK	Columnas
formula_integracion_pk	Si	No	for_int_id
doc_ver_a_for_int_fk	No	Si	doc_id doc_ver_version

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
doc_id doc_ver_version documento_version	documento_version	doc_id doc_ver_version formula_integracion

Relacionada por	Llave Primaria	Llave Foránea
integracion_formulas	for_int_id formula_integracion	for_int_id integracion_formulas

Tabla: formula_validacion

Generada por la entidad formula_validacion

Columnas	Tipo	PK	M
val_id	numeric(9)	Si	Si
doc_id	int	No	Si
doc_ver_version	smallint	No	Si
for_val_texto	varchar(255)	No	Si
for_val_texto_compilado	varchar(255)	No	No
for_val_nivel	char(1)	No	Si
for_val_independiente	Bit	No	Si

Indice	PK	FK	Columnas
formula_validacion_pk	Si	Si	val_id
for_val_a_doc_ver_fk	No	Si	doc_id doc_ver_version

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
doc_id doc_ver_version documento_version	documento_version	doc_id doc_ver_version formula_validacion
val_id validacion	validacion	val_id formula_validacion

Relacionada por	Llave Primaria	Llave Foránea
if_then_else	val_id formula_validacion	for3_val_id if_then_else
if_then_else	val_id formula_validacion	for_val_id if_then_else
if_then_else	val_id formula_validacion	val_id if_then_else
if_then_else	val_id formula_validacion	for2_val_id if_then_else

Tabla: grupo

Generada por la entidad grupo

Columnas	Tipo	PK	M
grupo_id	numeric(4)	Si	Si
grupo_descripcion	varchar(50)	No	No

Indice	PK	FK	Columnas
grupo_pk	Si	No	grupo_id

Lista de relaciones

Relacionada por	Llave Primaria	Llave Foránea
documento	grupo_id grupo	grupo_id documento

Tabla: if_then_else

Generada por la entidad if_then_else

Columnas	Tipo	PK	M
val_id	numeric(9)	Si	Si
for_val_id	numeric(9)	No	Si
for2_val_id	numeric(9)	No	No
for3_val_id	numeric(9)	No	No

Indice	PK	FK	Columnas
if_then_else_pk	Si	Si	val_id
if_fk	No	Si	for_val_id
then_fk	No	Si	for2_val_id
else_fk	No	Si	for3_val_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
val_id formula_validacion	formula_validacion	for3_val_id if_then_else
val_id formula_validacion	formula_validacion	for_val_id if_then_else
val_id formula_validacion	formula_validacion	val_id if_then_else
val_id formula_validacion	formula_validacion	for2_val_id if_then_else

Tabla: institucion

Generada por la entidad institucion

Columnas	Tipo	PK	M
ins_sector	smallint	Si	Si
ins_clave	int	Si	Si
ins_nombre	varchar(50)	No	No
ins_nombre_corto	varchar(20)	No	No
ins_nombre_temporal	char(10)	No	Si

Indice	PK	FK	Columnas
institucion_pk	Si	No	ins_sector ins_clave

Lista de relaciones

Relacionada por	Llave Primaria	Llave Foránea
area_institucion	ins_sector ins_clave institucion	ins_sector ins_clave area_institucion
envio	ins_sector ins_clave institucion	ins_sector ins_clave envio

Tabla: integracion_columnas

Generada por la relación integracion_columnas

Columnas	Tipo	PK	M
doc_id	int	Si	Si
doc_ver_version	smallint	Si	Si
col_orden	smallint	Si	Si
tab_ope_id	int	Si	Si
col_ope_id	smallint	Si	Si

Indice	PK	FK	Columnas
integracion_columnas_pk	Si	No	doc_id doc_ver_version col_orden tab_ope_id col_ope_id
integracion_columnas2_fk	No	Si	doc_id doc_ver_version col_orden
integracion_columnas_fk	No	Si	tab_ope_id col_ope_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
tab_ope_id col_ope_id columna_operacional	columna_operacional	tab_ope_id col_ope_id integracion_columnas
doc_id doc_ver_version col_orden columnas	columnas	doc_id doc_ver_version col_orden integracion_columnas

Tabla: integracion_formulas

Generada por la relación integracion_formulas

Columnas	Tipo	PK	M
tab_ope_id	Int	Si	Si
col_ope_id	Smallint	Si	Si
for_int_id	Numeric(9)	Si	Si

Indice	PK	FK	Columnas
integracion_formulas_pk	Si	No	tab_ope_id col_ope_id for_int_id
integracion_formulas2_fk	No	Si	tab_ope_id col_ope_id
integracion_formulas_fk	No	Si	for_int_id

Llave Primaria	Relaciones	Llave Foránea
for_int_id formula_integracion	formula_integracion	for_int_id integracion_formulas
tab_ope_id col_ope_id columna_operacional	columna_operacional	tab_ope_id col_ope_id integracion_formulas

Tabla: mensaje

Generada por la entidad mensaje

Columnas	Tipo	PK	M
men_numero	Int	Si	Si
men_severity	smallint	No	Si
men_tipo	char(1)	No	Si
men_texto	varchar(40)	No	No

Indice	PK	FK	Columnas
mensaje_pk	Si	No	men_numero

Lista de relaciones

Relacionada por	Llave Primaria	Llave Foránea
bitacora	men_numero mensaje	men_numero bitacora

Tabla: mensaje_error_val

Generada por la entidad mensaje_error_val

Columnas	Tipo	PK	M
men_err_val_error	char(7)	Si	Si
men_err_val_texto	varchar(100)	No	No

Indice	PK	FK	Columnas
mensaje_error_val_pk	Si	No	men_err_val_error

Lista de relaciones

Relacionada por	Llave Primaria	Llave Foránea
validacion	men_err_val_error mensaje_error_val	men_err_val_error validacion

Tabla: retraso_envio

Generada por la entidad retraso_envio

Columnas	Tipo	PK	M
ins_sector	smallint	Si	Si
ins_clave	int	Si	Si
area_id	smallint	Si	Si
doc_id	int	Si	Si
ret_env_ano	smallint	Si	Si
ret_env_periodo	smallint	Si	Si
ret_env_fecha	datetime	No	No
ret_env_notificacion_enviada	bit	No	Si
ret_env_notificacion_recibida	bit	No	Si

Indice	PK	FK	Columnas
retraso_envio_pk	Si	No	ins_sector ins_clave area_id doc_id ret_env_ano ret_env_periodo
doc_area_ins_a_ret_env2_fk	No	Si	ins_sector ins_clave area_id doc_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
ins_sector ins_clave area_id doc_id documento_area_institucion	documento_area_institucion	ins_sector ins_clave area_id doc_id retraso_envio

Tabla: servidor_aplicativo

Generada por la entidad servidor_aplicativo

Columnas	Tipo	PK	M
ser_app_id	numeric(8)	Si	Si
ser_app_nombre	varchar(15)	No	Si
ser_app_login	varchar(15)	No	Si
ser_app_password	varchar(15)	No	Si
ser_app_sql_server	varchar(15)	No	Si
ser_app_sql_server_login	varchar(15)	No	Si
ser_app_sql_server_pwd	varchar(15)	No	Si

Indice	PK	FK	Columnas
servidor_aplicativo_pk	Si	No	ser_app_id

Lista de relaciones

Relacionada por	Llave Primaria	Llave Foránea
envio_a_central	ser_app_id servidor_aplicativo	ser_app_id envio_a_central
status_envio_central	ser_app_id	ser_app_id

Tabla: servidor_produccion

Generada por la entidad servidor_produccion

Columnas	Tipo	PK	M
srv_id	smallint	Si	Si
srv_nombre	varchar(15)	No	Si
srv_login	varchar(15)	No	Si
srv_password	varchar(15)	No	Si

Indice	PK	FK	Columnas
servidor_produccion_pk	Si	No	srv_id

Lista de relaciones

Relacionada por	Llave Primaria	Llave Foránea
tabla_operacional	srv_id servidor_produccion	srv_id tabla_operacional

Tabla: sp_validacion

Generada por la entidad sp_validacion

Columnas	Tipo	PK	M
val_id	numeric(9)	Si	Si
doc_id	int	No	Si
doc_ver_version	smallint	No	Si
sp_orden	smallint	No	Si
sp_bd	varchar(15)	No	Si
sp_store_proc	varchar(15)	No	Si

Indice	PK	FK	Columnas
sp_validacion_pk	Si	Si	val_id
doc_ver_a_sp_val_fk	No	Si	doc_id doc_ver_version

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
doc_id doc_ver_version	documento_version	doc_id doc_ver_version
documento_version		sp_validacion
val_id validacion	validacion	val_id sp_validacion

Tabla: status_envio_central

Generada por la entidad status_envio_central

Columnas	Tipo	PK	M
ser_app_id	numeric(8)	Si	Si
ins_sector	smallint	Si	Si
ins_clave	int	Si	Si
env_folio	numeric(9)	Si	Si
sta_env_cen_status	char(2)	No	Si
sta_env_cen_status_bita	char(2)	No	Si

Indice	PK	FK	Columnas
status_envio_central_pk	Si	No	ser_app_id ins_sector ins_clave env_folio
env_a_sta_env_cen2_fk	No	Si	ins_sector ins_clave env_folio

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
ins_sector ins_clave env_folio	envio	ins_sector ins_clave env_folio
envio		status_envio_central
ser_app_id	servidor_aplicativo	ser_app_id

Tabla: tabla_operacional

Generada por la entidad tabla_operacional

Columnas	Tipo	PK	M
tab_ope_id	int	Si	Si
srv_id	smallint	No	Si
tab_ope_bd	varchar(15)	No	Si
tab_ope_tabia	varchar(25)	No	Si

Indice	PK	FK	Columnas
tabla_operacional_pk	Si	No	tab_ope_id
ser_pro_a_tab_ope_fk	No	Si	srv_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
srv_id servidor_produccion	servidor_produccion	srv_id tabla_operacional

Relacionada por	Llave Primaria	Llave Foránea
columna_operacional	tab_ope_id tabla_operacional	tab_ope_id columna_operacional

Tabla usuario

Generada de la entidad usuario

Columnas	Tipo	PK	M
usu_id	char(8)	Si	Si
ins_sector	smallint	No	Si
ins_clave	int	No	Si
are_id	smallint	No	Si
usu_password	varchar(10)	No	No
usu_nombre	varchar(40)	No	No
usu_telefono	varchar(15)	No	No

Indice	PK	FK	Columnas
usuario_pk	Si	No	usu_id
are_ins_a_usu_fk	No	Si	ins_sector ins_clave area_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
ins_sector ins_clave area_id area_institucion	area_institucion	ins_sector ins_clave area_id usuario

Relacionada por	Llave Primaria	Llave Foránea
documento_area_institucion envio	usu_id usuario usu_id usuario	usu_id documento_area_institucion usu_id envio

Tabla: validacion

Generada por la entidad validacion

Columnas	Tipo	PK	M
val_id	numeric(9)	Si	Si
men_err_val_error	char(7)	No	No
val_habilitada	bit	No	Si

Indice	PK	FK	Columnas
validacion_pk	Si	No	val_id
men_err_val_a_val_fk	No	Si	men_err_val_error

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
men_err_val_error mensaje_error_val	mensaje_error_val	men_err_val_error validacion

Relacionada por	Llave Primaria	Llave Foránea
validacion_catalogo	val_id validacion	val_id validacion_catalogo
formula_validacion	val_id validacion	val_id formula_validacion
sp_validacion	val_id validacion	val_id sp_validacion

Tabla: validacion_catalogo

Generada por la entidad validacion_catalogo

Columnas	Tipo	PK	M
doc_id	int	Si	Si
doc_ver_version	smallint	Si	Si
col_orden	smallint	Si	Si
cat_id	int	Si	Si
col_cat_orden	smallint	Si	Si
val_id	numeric(9)	Si	Si

Indice	PK	FK	Columnas
validacion_catalogo_pk	Si	No	doc_id doc_ver_version col_orden cat_id col_cat_orden val_id
col_cat_a_val_cat2_fk	No	Si	cat_id col_cat_orden
col_a_val_cat2_fk	No	Si	doc_id doc_ver_version col_orden
inhr_5601_fk	No	Si	val_id

Lista de relaciones

Llave Primaria	Relaciones	Llave Foránea
doc_id doc_ver_version	columnas	doc_id doc_ver_version
col_orden columnas		col_orden validacion_catalogo
cat_id col_cat_orden	columna_catalogo	cat_id col_cat_orden
columna_catalogo		validacion_catalogo
val_id validacion	validacion	val_id validacion_catalogo

Ya que definimos la base de datos de *metadata* estamos listos para iniciar la implementación del *middleware*, la cual desarrollaremos en el siguiente capítulo, e igualmente revisaremos el correcto funcionamiento del mismo.

Capítulo 7

Implementación del Middleware

En este capítulo se mostrarán algunos de los procesos que realiza el *middleware* como parte de un Servidor Aplicativo. Destacando entre ellas el proceso de acceso al *middleware*, el proceso de validación y el proceso de integración; así como, algunas de las pruebas realizadas con las diferentes configuraciones antes mencionadas en el capítulo IV.

7.1 Fase de Implementación

Después de haber realizado un análisis y diseño sobre el *middleware*, sólo nos queda la fase de implementación, la cual consiste en desarrollar todos los procesos que conforman el *middleware*, así como instalar la aplicación con los componentes necesarios para montar un ambiente de producción.

El trabajo que el *middleware* realiza no es apreciado ya que es un componente que como su nombre lo indica se encuentra en medio de otros componentes; sin embargo, sin el no se podría llevar a cabo la filtración e integración de la información que las Entidades Supervisadas envían a la Entidad Normativa.

La implementación del *middleware* fué desarrollada con librerías de *Open Server* y ANSI C para las plataformas de UNIX y WNT. Se realizaron alrededor de 80 funciones de las cuales sólo mostraremos algunas.

A continuación presentaremos los listados de algunos de los procesos más importantes de este sistema. En este archivo se muestran algunas de las funciones necesarias para el *middleware*, el cual se desarrolló con las librerías *Open Server* y *Open Client de Sybase*. El *middleware* como una aplicación multi-thread necesita de varios procesos y uno de

ellos es el *start handler*, donde se inicializa el ambiente, se inicializan los módulos del validador, integrador, acceso; así como, la definición de los procedimientos registrados.

El *connect handle* es otro de los eventos que tiene el *middleware*, el cual se dispara cuando existe una conexión de algún Cliente y el *disconnect handler* es el evento que registra la desconexión de un Cliente.

```

/*****
**
**      eventHndls.c
**
** Descripción:
**      Este archivo contiene los handlers para manejo de los eventos.
*****/
#include "osSA.h"
#include <errno.h>

/*****
**
**      VARIABLES EXTERNAS
**
*****/
CS_INT   SRV_SHUTDOWN_NO;
extern THREAD_LIST  threadList;
extern QUEUE_LIST  queueList;
extern SISTEMA_INFO  sistemaInfo;
extern USER_LIST  userList;

/*****
**
**      startHndI
**
** Descripción:
**      Esta función es para el manejo del evento SRV_START donde se instalan
**      los handlers de SRV_CONNECT, SRV_DISCONNECT, SRV_LANGUAGE Y SRV_STOP.
**
**      Se instalan Procedimientos Registrados.
**
** Parametros:
**      serverPtr  Apuntador a la estructura de control del Open Server.
*****/
CS_RETCODE startHndI( SRV_SERVER *serverPtr )
{
    SRV_PROC      *srvtempPtr;
    SRV_OBJID     objId;
    CS_INT        infoProcedimiento; /*informa si el procedimiento tuvo algun error*/
    THREAD_INFO   *threadInfo;
    CS_CHAR       temp[MAX_LEN_ARR];
    CS_CHAR       msgBuffer[MAX_LEN_MSG_LOG];
    CS_CHAR       *nomThread;
    CS_INT        i;
    CS_RETCODE     ret;
    char          *argv[5];
    CS_CHAR       tmp[10];
    int           pid;

    serverPtr = serverPtr; /* Keep compiler happy; but it's not happy =( */
    TRACE(TRC_FUNCTION_ENTER_RETURN,"startHndI: FUNCTION_ENTER >>> \n");
    /*
    ** Se imprime la configuración del Open Server
    */
    sprintf(msgBuffer, ".....Configuración Inicial..... \n"
        "Open Server Name:      %s\n"
        "Institucion:            %s\n"
        "Tipo Sistema:          %s\n"
    );

```

```

"SQL SERVER:          %s\n"
"SERVIDOR APLICATIVO - %s\n"
"Archivo de log:      %s\n"
"Archivo de configuracion %s\n"
"Numero max de conexiones: %d\n"
"Numero VALIDADOR_T: %d\n"
"Numero INTEGRADOR_T: %d\n"
"Nivel de Trace:      %d\n",
sistemaInfo.openServerName,
sistemaInfo.institucion,
(sistemaInfo.tipoSistema == SERVER_CENTRAL)? "SERVER CENTRAL" : "SERVER LOCAL",
sistemaInfo.sqlServerName,
sistemaInfo.sa,
sistemaInfo.errorLog,
sistemaInfo.configFile,
sistemaInfo.maxNumConnections,
sistemaInfo.numValidadorT,
sistemaInfo.numIntegradorT,
sistemaInfo.traceLevel);
srv_log((SRV_SERVER *)NULL, CS_FALSE, msgBuffer, CS_NULLTERM);

/*printf("%s", msgBuffer);*/

ret=srv_log((SRV_SERVER *)NULL, CS_TRUE, "— Servidor Aplicativo Operando ———\n", CS_NULLTERM);
CHECKEXIT(ret, "srv_log: SAR_START failed\n");

/*
** Instalacion del handler SRV_CONNECT
*/
if(srv_handle((SRV_SERVER *)NULL, SRV_CONNECT, (SRV_EVENTHANDLE_FUNC) connectHndl)==
    (SRV_EVENTHANDLE_FUNC)NULL)
    ret=CS_FAIL;
else
    ret=CS_SUCCEEDED;
CHECKEXIT(ret, "srv_handle: SRV_CONNECT failed\n");
TRACE(TRC_FUNCTION_RESULTS, "Inicializacion de EVENT HANDLERS.\n");
/*printf("Inicializacion de EVENT HANDLERS: \n");*/
TRACE(TRC_FUNCTION_RESULTS, " SRV_CONNECT\n");
/*printf(" SRV_CONNECT\n");*/

/*
** Instalacion del handler SRV_DISCONNECT
*/
if(srv_handle((SRV_SERVER *)NULL, SRV_DISCONNECT, (SRV_EVENTHANDLE_FUNC) disconnectHndl)==
    (SRV_EVENTHANDLE_FUNC)NULL)
    ret=CS_FAIL;

/*
** Instalacion del handler SRV_LANGUAGE
*/
if(srv_handle((SRV_SERVER *)NULL, SRV_LANGUAGE, (SRV_EVENTHANDLE_FUNC) languageHndl)==
    (SRV_EVENTHANDLE_FUNC)NULL)
    ret=CS_FAIL;

/* Inicializacion de la estructura DOCTO_LIST */
srv_bzero(&doctoList, sizeof(DOCTO_LIST));

/* Inicializacion de la estructura MONITOR_INFO */
srv_bzero(&monitorInfo, sizeof(MONITOR_INFO));

/* Inicializacion de la estructura QUEUE_LIST */
srv_bzero(&queueList, sizeof(QUEUE_LIST));

/* Inicializacion de la estructura THREAD_LIST */
srv_bzero(&threadList, sizeof(THREAD_LIST));

/* Inicializacion de conexiones */
iniConexion(&conexionesIntegrador);

```

```

iniConexion(&conexionesBulkCopy),

TRACE(TRC_FUNCTION_RESULTS, "Estructuras de Datos inicializadas\n"),
/*printf("Estructuras de Datos inicializadas\n");*/

TRACE(TRC_FUNCTION_RESULTS, "Creacion PROCEDIMIENTOS REGISTRADOS \n"),
/*printf("Creacion de PROCEDIMIENTOS REGISTRADOS \n");*/

/*
** Creacion del thread para la creacion de los procedimientos almacenados
**   SRV_SPAWN
*/
ret=srv_spawn(&srvttempPtr,SRV_DEFAULT_STACKSIZE,(SRV_EVENTHANDLE_FUNC)spawnFunc,(CS_VOID
*)NULL,
    SRV_C_DEFAULTPRI),
CHECKEXIT(ret,"srv_spawn: failed\n");

/*
** Definicion y creacion del procedimiento registrado accesalInfo
*/
ret=srv_regdefine(srvttempPtr,"rp_accesalInfo",CS_NULLTERM,{SRV_EVENTHANDLE_FUNC}rp_accesalInfo),
CHECKEXIT(ret,"srv_regdefine: rp_accesalInfo failed\n");
/* Definicion y creacion del procedimiento registrado integraPeticion
*/
ret=srv_regdefine(srvttempPtr,"rp_integraPeticion",CS_NULLTERM,(SRV_EVENTHANDLE_FUNC)rp_integraPeti
cion),
CHECKEXIT(ret,"srv_regdefine: rp_integraPeticion failed\n"),
/*
** Definicion y creacion del procedimiento registrado rp_compilaFormula
*/
ret=srv_regdefine(srvttempPtr,"rp_compilaformula",CS_NULLTERM,(SRV_EVENTHANDLE_FUNC)rp_compilaFor
mula);
CHECKEXIT(ret,"srv_regdefine: rp_compilaFormula failed\n"),
/*
** Definicion y creacion del procedimiento registrado procDocumento
*/
ret=srv_regdefine(srvttempPtr,"rp_procdocumento",CS_NULLTERM,(SRV_EVENTHANDLE_FUNC)rp_procdocum
ento),
CHECKEXIT(ret,"srv_regdefine: rp_procdocumento failed\n");
/*
** Definicion y creacion del procedimiento registrado rp_timer
*/
ret=srv_regdefine(srvttempPtr,"rp_timer",CS_NULLTERM,(SRV_EVENTHANDLE_FUNC)rp_timer);
CHECKEXIT(ret,"srv_regdefine: rp_timer failed\n");
ret=srv_regcreate(srvttempPtr,&infoProcedimiento);
CHECKEXIT(ret,"srv_regcreate: rp_timer failed\n"),
TRACE(TRC_FUNCTION_RESULTS, " rp_timer \n"),
/*printf(" rp_timer \n");*/
/*
** Definicion y creacion del procedimiento registrado monitor
*/
ret=srv_regdefine(srvttempPtr,"rp_monitor",CS_NULLTERM,(SRV_EVENTHANDLE_FUNC)rp_monitor);
CHECKEXIT(ret,"srv_regdefine: rp_monitor failed\n"),
ret=srv_regcreate(srvttempPtr,&infoProcedimiento),
CHECKEXIT(ret,"srv_regcreate: rp_monitor failed\n");
TRACE(TRC_FUNCTION_RESULTS, " rp_monitor \n"),
/*printf(" rp_monitor \n");*/
/*** fin de la creacion de los procedimientos registrados ***/

/*
** Creacion del thread ACCESS_T
*/
ret=srv_spawn(&srvttempPtr,SRV_DEF_STACKSIZE*4,(SRV_EVENTHANDLE_FUNC)accessHndl,NULL,SRV_C_
DEFAULTPRI);
CHECKRET(ret,"srv_spawn: accessHndl failed\n"),

/* Creacion del thread NOTIFICADOR_T

```

```

*
ret=srv_spawn(&srvtmpPtr,SRV_DEF_STACKSIZE,notificadorHndi,NULL,SRV_C_DEFAULTPRI);
CHECKRET(ret,"srv_spawn: notificadorHndi failed\n");
/*
** Creacion del thread MNT_T
*/
ret=srv_spawn(&srvtmpPtr,SRV_DEF_STACKSIZE*4,(SRV_EVENTHANDLE_FUNC)mntHndi,NULL,SRV_C_DE
FAULTPRI);
CHECKRET(ret,"srv_spawn: mntHndi failed\n");
/*
** Creacion de los threads VALIDADOR_TI
*/
for(i=1; i <= sistemaInfo.numValidadorT; i++)
{
    ret=srv_spawn(&srvtmpPtr,SRV_DEF_STACKSIZE*4,(SRV_EVENTHANDLE_FUNC)validadorHndi,NULL,SRV_
C_DEFAULTPRI);
    CHECKRET(ret,"srv_spawn: VALIDADOR_T failed\n");
    strcpy(temp,"VALIDADOR_T");
    itoa(i,tmp,10);
    strcat(temp,tmp);
    nomThread = srv_alloc(strlen(temp)+1);
    strcpy(nomThread,temp);
    ret=srv_thread_props(srvtmpPtr,CS_SET,SRV_T_USTATE,nomThread,
        strlen(temp),(CS_INT *)NULL);
    CHECKRET(ret,"srv_thread_props: VALIDADOR_T failed\n");
    ret=srv_thread_props(srvtmpPtr,CS_SET,SRV_T_USERDATA,(CS_VOID *)threadInfo,
        sizeof(threadInfo),(CS_INT *)NULL);
    CHECKRET(ret,"srv_thread_props: VALIDADOR_T failed\n");
}/* fin de la creacion de los threads de tipo validador */

/*
** Creacion de los threads INTEGRADOR_TI
*/
for(i=1; i <= sistemaInfo.numIntegradorT; i++)
{
    ret=srv_spawn(&srvtmpPtr,SRV_DEF_STACKSIZE*5,(SRV_EVENTHANDLE_FUNC)integradorHndi,NULL,SRV
_C_DEFAULTPRI);
    CHECKRET(ret,"srv_spawn: integradorHndi failed\n");
    strcpy(temp,"INTEGRADOR_T");
    itoa(i,tmp,10);
    strcat(temp,tmp);
    nomThread = srv_alloc(strlen(temp)+1);
    strcpy(nomThread,temp);
    ret=srv_thread_props(srvtmpPtr,CS_SET,SRV_T_USTATE,nomThread,
        strlen(temp),(CS_INT *)NULL);
    CHECKRET(ret,"srv_thread_props: integradorHndi failed\n");
    ret=srv_thread_props(srvtmpPtr,CS_SET,SRV_T_USERDATA,(CS_VOID *)threadInfo,
        sizeof(threadInfo),(CS_INT *)NULL);
    CHECKRET(ret,"srv_thread_props: integradorHndi failed\n");
}/* fin de la creacion de los threads de tipo validador */

/*
** Creacion de las colas para los diferentes tipos de threads
*/
TRACE(TRC_FUNCTION_RESULTS, "Creacion de MESSAGES QUEUEUS:\n");
/*printf("Creacion de MESSAGES QUEUEUS:\n");*/
if( createMessageQ(ACCESS_Q_NAME,&objId) == CS_FAIL)
{
    TRACE(TRC_FUNCTION_RESULTS,"startHndi: createMessageQ ACCESS_Q_NAME failed\n");
    TRACE(TRC_FUNCTION_ENTER_RETURN,
        "startHndi: TRC_FUNCTION RETURN (CS_FAIL)<<<\n");
    return (CS_FAIL);
}
TRACE(TRC_FUNCTION_RESULTS, " ACCESS_Q \n");
/*printf(" ACCESS_Q \n");*/
if( createMessageQ(VALIDADOR_Q_NAME,&objId) == CS_FAIL)

```

```

{
    TRACE(TRC_FUNCTION_RESULTS,"startHnd: createMessageQ VALIDA_Q_NAME failed\n");
    TRACE(TRC_FUNCTION_ENTER_RETURN,
        "startHnd: TRC_FUNCTION_RETURN (CS_FAIL)<<<\n");
    return (CS_FAIL);
}
TRACE(TRC_FUNCTION_RESULTS, " VALIDA_Q \n");
/*printf( " VALIDA_Q \n");*/

if( createMessageQ(INTEGRADOR_Q_NAME,&objId) == CS_FAIL)
{
    TRACE(TRC_FUNCTION_RESULTS,"startHnd: createMessageQ INTEGRADOR_Q_NAME failed\n");
    TRACE(TRC_FUNCTION_ENTER_RETURN,
        "startHnd: TRC_FUNCTION_RETURN (CS_FAIL)<<<\n");
    return (CS_FAIL);
}
TRACE(TRC_FUNCTION_RESULTS, " INTEGRADOR_Q \n");
/*printf( " INTEGRADOR_Q \n");*/

if( createMessageQ(NOTIFICADOR_Q_NAME,&objId) == CS_FAIL)
{
    TRACE(TRC_FUNCTION_RESULTS,"startHnd: createMessageQ NOTIFICADOR_Q_NAME failed\n");
    TRACE(TRC_FUNCTION_ENTER_RETURN,
        "startHnd: TRC_FUNCTION_RETURN (CS_FAIL)<<<\n");
    return (CS_FAIL);
}
TRACE(TRC_FUNCTION_RESULTS, " NOTIFICADOR_Q \n");
/*printf( " NOTIFICADOR_Q \n");*/

/*
** Definicion del Evento SRV_SHUTDOWN
*/
if( (SRV_SHUTDOWN_NO = srv_define_event( serverPtr, SRV_QUEUED, (CS_CHAR*)"SRV_SHUTDOWN",
    CS_NULLTERM) ) != (CS_INT)0 )
    LOGGER(ret,"startHandler>No se definio el evento: srv_define_event \n");

if (srv_handle((SRV_SERVER*)NULL, SRV_SHUTDOWN_NO.shutdownHnd)
    == (SRV_EVENTHANDLE_FUNC)NULL )
    ret = CS_FAIL;
else
    ret = CS_SUCCEED;
LOGGER(ret,"srv_handle:SRV_SHUTDOWN:startHandler");*/

TRACE(TRC_FUNCTION_ENTER_RETURN,"startHnd: FUNCTION_RETURN (CS_SUCCEED) <<< \n");
return (CS_SUCCEED);
} /* fin de startHnd */

/*****
**
**      (connectHnd)
**
**  Descripcion:
**      Funcion donde se valida que el usuario que se conecta sea un usuario
**      valido, ademas de crearse una cola para este, por la cual puede recibir
**      informacion.
**
**  Parametros:
**      srvprocPtr  Apuntador a la estructura SRV_PROC.
**
*****/
CS_RETCODE connectHnd( SRV_PROC  _srvprocPtr )
{
    CS_CHAR      login[MAX_LEN_ARR];
    CS_INT      loginlen=MAX_LEN_ARR;
    CS_CHAR      passwd[MAX_LEN_ARR];
    CS_INT      passwdlen=MAX_LEN_ARR;
    CS_CHAR      hname[MAX_LEN_ARR];
    CS_INT      outlen;
    CS_INT      spid;

```



```

CS_INT         type;
CS_CHAR        msgBuffer[MAX_LEN_ARR];
SRV_OBJID      msgQueueID;
CS_CHAR        queueName[MAX_LEN_ARR];
CS_SERVERMSG   msg;
THREAD_INFO    *threadInfo;
CS_RETCODE     ret;
CS_CHAR        tmp[10];
CS_BOOL        administrador;

TRACE(TRC_FUNCTION_ENTER_RETURN,"connectHnd: FUNCTION_ENTER >>>\n");

/*
** Si se activado el evento de SHUTDOWN, el usuario no podra conectarse.
*/
if(sistemaInfo.shutdown == CS_TRUE)
{
    threadInfo->loginFail = CS_TRUE;
    srv_senddone(srvprocPtr,SRV_DONE_FINAL | SRV_DONE_ERROR,
        CS_TRAN_UNDEFINED,(CS_INT)0);
    TRACE(TRC_FUNCTION_ENTER_RETURN,
        "connectHnd: TRC_FUNCTION_RETURN(CS_SUCCEED) SHUTDOWN IS TRUE. \n");
    return CS_FAIL;
}

/*
** Obtencion del usuario y su password que se ha conectado
*/
ret=srv_thread_props(srvprocPtr,CS_GET,SRV_T_USER,login,loginlen,&outlen);
CHECKEXIT(ret,"srv_thread_props: connectHnd SRV_T_USER.\n");
login[outlen] = '\0';
ret=srv_thread_props(srvprocPtr,CS_GET,SRV_T_PWD,passwd,passwdlen,&outlen);
CHECKEXIT(ret,"srv_thread_props: connectHnd SRV_T_PWD.\n");
passwd[outlen] = '\0';

sprintf(msgBuffer,">>>>>> RPC TYPE %d, HOST[%s] LOGIN[%s]\n",
    type, hname, login);
TRACE(TRC_FUNCTION_RESULTS,msgBuffer);
/*
** Si se trata de un rpc de administracion,
** como sp_who, sp_ps -> terminar conexion con exito
*/
/*if ( ( strcmp(login,"ssa_oper")!=0) || ( strcmp(login,"")!=0) )*/
if ( type == SRV_TSUBPROC || type == SRV_TSITE )
{
    srv_senddone(srvprocPtr, SRV_DONE_FINAL, CS_TRAN_UNDEFINED,
        (CS_INT)0);
    return (CS_SUCCEED);
}

/*
** Se verifica que el usuario sea un usuario valido, es decir
** que este en la lista de usuarios de la institucion o que
** sea el login administrador.
*/
administrador = CS_FALSE;
if( buscaUser(login, passwd) == 0)
{
    if ( strcmp(login,sistemaInfo.mntLogin) != 0 ||
        strcmp(passwd,sistemaInfo.mntPwd) != 0)
    {
        srv_bzero(&msg,sizeof(CS_SERVERMSG));
        msg.msgnumber = ERROR_LOGIN;
        msg.seventy = SEGURIDAD_SEVERITY;
        strcpy(msg.text,"Login failed");
        msg.textlen = strlen(msg.text);
        msg.status = (CS_FIRST_CHUNK|CS_LAST_CHUNK);
        srv_sendinfo(srvprocPtr, &msg, CS_TRAN_UNDEFINED);
        srv_senddone(srvprocPtr,SRV_DONE_FINAL | SRV_DONE_ERROR,

```

```

        CS_TRAN_UNDEFINED,(CS_INT)0);
        sprintf(msgBuffer,"connectHndl: Conexion fallida del usuario <%s>\n".login);
        TRACE(TRC_FUNCTION_RESULTS,msgBuffer);
        TRACE(TRC_FUNCTION_ENTER_RETURN,
            "connectHndl TRC_FUNCTION_RETURN (CS_SUCCEED)<<<\n");
        return (CS_FAIL);
    }
    else
    {
        administrador = CS_TRUE,
    }
}

/*
** Se crea una estructura THREAD_INFO asociada a la conexion.
*/
threadInfo = srv_alloc ( sizeof(THREAD_INFO ) );
srv_bzero(threadInfo, sizeof(THREAD_INFO));
if (administrador == CS_TRUE )
{
    threadInfo->tipo=ADMON_THREAD;
}
else
{
    threadInfo->tipo = USER_THREAD ,
    threadInfo->msgQId = msgQueueID,
}

/*
** Encolar threadInfo en threadList
*/
insertThread ( &threadList, threadInfo );

/*
** Aceptar la conexion por parte del cliente.
*/
srv_senddone(srvprocPtr, SRV_DONE_FINAL, CS_TRAN_UNDEFINED,
(CS_INT)0),

TRACE(TRC_FUNCTION_ENTER_RETURN, "connectHandler. FUNCTION_RETURN (CS_SUCCEED) <<<\n");
return (CS_SUCCEED);

} /* Fin de la funcion connectHndl */

.....
..
..      (disconnectHndl
..
..  Descripcion
..      Funcion donde se checa al usuario que se ha desconectado para
..      eliminar su cola y dar de baja sus servicios
..
..  Parametros:
..      srvprocPtr   Apuntador a la estructura SRV_PROC.
..
.....
CS_RETCODE disconnectHndl( SRV_PROC *srvprocPtr)
{
    CS_CHAR          login[MAX_LEN_ARR];
    CS_INT           loginLen=MAX_LEN_ARR;
    CS_INT           outLen;
    CS_CHAR          passwd[MAX_LEN_ARR];
    CS_INT           passwdLen=MAX_LEN_ARR;
    THREAD_INFO      *thread;
    CS_CHAR          msgBuffer[MAX_LEN_ARR];
    CS_CHAR          hname[MAX_LEN_ARR];
    CS_RETCODE       ret;
    CS_INT           type;

```

```

TRACE(TRC_FUNCTION_ENTER_RETURN,"disconnectHndl: FUNCTION_ENTER >>>\n");

/*
** Se obtiene la estructura THREAD_INFO del
** usuario que se esta desconectando.
*/
ret=srv_thread_props(srvprocPtr,CS_GET,SRV_T_USER,login,loginlen,&outlen);
CHECKEXIT(ret,"srv_thread_props: connectHndl SRV_T_USER.\n");
login[outlen] = '\0';
ret=srv_thread_props(srvprocPtr,CS_GET,SRV_T_PWD,passwd,passwdlen,&outlen);
CHECKEXIT(ret,"srv_thread_props: connectHndl SRV_T_PWD.\n");
passwd[outlen] = '\0';

sprintf(msgBuffer,"<<<<<< RPC TYPE %d, HOST[%s] LOGIN[%s]\n",
        type, hname, login );
TRACE(TRC_FUNCTION_RESULTS,msgBuffer);

/*
** Si se trata de un rpc de administracion,
** como sp_who, sp_ps -> terminar conexion con exito
**
if ( type == SRV_TSUBPROC || type == SRV_TSITE || (buscaUser(login, passwd) == 0) ||
    strcmp(login,sistemaInfo.mntLogin) != 0 || strcmp(passwd,sistemaInfo.mntPwd) != 0)
{
    return (CS_SUCCEED);
}

/*
** Si es un usuario normal, se borra la cola asociada a la conexion.
**
if (thread->tipo == USER_THREAD)
{
    destroyMessageQ( NULL, &(thread->msgQId) );
}

/*
** Se borra el THREAD_INFO de THREAD_LIST
**
borraThread(&threadList,thread);
srv_free(thread);

TRACE(TRC_FUNCTION_ENTER_RETURN,"disconnectHndl: FUNCTION_RETURN <<< \n");
return CS_SUCCEED;

} /* Fin de la funcion disconnectHndl */

```

Uno de los procesos más importantes que realiza el *middleware* es la verificación de la información que las Entidad Supervisadas envían a la Entidad Normativa, ésta consiste en aplicar varios procesos de depuración de la información por parte de la Entidad Normativa. El listado de este proceso se muestra a continuación, donde además se pueden observar los diferentes procesos de validación (*store procedures*, catálogos o fórmulas).

```

/*****
**
** validadorHndl c
**
** Descripción:
** Este archivo contiene el thread de validacion que se encarga
** de aplicar las reglas de validacion a los documentos.
**
*****/

```

```

#include "osSA.h"

/*****
**
** PROTOTIPOS LOCALES
**
*****/
CS_RETCODE getValidaciones(CS_COMMAND *,CS_VOID *,CS_INT);
CS_RETCODE validacionSP(CS_CONNECTION *,PETICION *,CS_INT *);
CS_RETCODE validadorError(CS_CONNECTION *,PETICION *,CS_INT, CS_CHAR *);
CS_RETCODE getExpInd(CS_COMMAND *,PETICION *,CS_INT);
CS_RETCODE insertaHistorico(CS_COMMAND *,PETICION *,CS_INT);
CS_RETCODE validacionExpr(CS_CONNECTION *,PETICION *,CS_INT *);
CS_RETCODE validacionCat(CS_CONNECTION *,PETICION *,CS_INT *);

/*****
**
** validadorHndI
**
** Descnpcion.
** Thread que se encarga de aplicar las reglas de validacion a los documentos.
**
*****/
CS_RETCODE validadorHndI( SRV_PROC   *srvProc )
{
    CS_CONNECTION *conn_ptr;
    SRV_OBJID      msgQidV;
    SRV_OBJID      msgQidI;
    PETICION       *peticion;
    PETICION_MNT   *peticionMnt;
    RESULT_INFO    *resultInfo;
    CS_CHAR        bufferLog[MAX_LEN_MSG_LOG];
    CS_CHAR        sqlcommand[MAX_LONG_SQL_COMMAND];
    CS_CHAR        tmp[10];
    CS_CHAR        statusPeticion[3];
    CS_INT         info;
    CS_INT         i;
    CS_INT         status;
    CS_INT         integraHabil;
    CS_RETCODE     ret;

    sprintf(bufferLog,"validadorHndI: FUNCTION ENTER >>>\n");
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);

    srvProc = SRV_CURPROC,
    /*
    ** Abre una conexion constante hacia el SQL SERVER
    ** utilizan un login especial para el VALIDADOR_T
    */
    ret = ct_con_alloc(sistemaInfo.cnx_ptr,&conn_ptr);
    CHECKRET(ret,"validadorHndI: ct_con_alloc\n");

    ret = ct_con_props(conn_ptr,CS_SET,CS_USERNAME,sistemaInfo.validadorLogin,CS_NULLTERM NULL);
    CHECKRET(ret,"validadorHndI: ct_con_props user\n");

    ret = ct_con_props(conn_ptr,CS_SET,CS_PASSWORD,sistemaInfo.validadorPwd,CS_NULLTERM NULL);
    CHECKRET(ret,"validadorHndI: ct_con_props password\n");

    ret = ct_con_props(conn_ptr,CS_SET,CS_HOSTNAME,sistemaInfo.openServerName,CS_NULLTERM NULL);
    CHECKRET(ret,"validadorHndI: ct_con_props hostname\n");

    ret = ct_connect(conn_ptr,sistemaInfo.sqlServerName,CS_NULLTERM);
    CHECKRET(ret,"validadorHndI: ct_connect\n");

    /*
    ** Ejecuta ssa_getPeticionesPorValidar
    ** para recuperar las peticiones que se quedaron en un estado anterior al de validacion.
    ** Estas peticiones deberan ser procesadas con el algoritmo que se describe para esta
    */
}

```

```

** funcion
estas peticiones tienen status CE y tipo_servicio = "I"

/*
** Se obtienen los OBJECT_ID de las messages queues (INTEGRADOR_Q y VALIDADOR_Q).
*/
ret = srv_getobjid(SRV_C_MQUEUE,VALIDADOR_Q_NAME,CS_NULLTERM,&msgQidV,&info);
CHECKRET(ret, "validadorHndI:srv_getobjid");

ret = srv_getobjid(SRV_C_MQUEUE,INTEGRADOR_Q_NAME,CS_NULLTERM,&msgQidI,&info);
CHECKRET(ret, "validadorHndI:srv_getobjid");

while(CS_TRUE)
{
    /*
    ** Lee de la cola VALIDADOR_Q y se queda en estado de sleep hasta que lea algo.
    ** EL haber leído algo significa que se hizo una petición al VALIDADOR_T.
    */
    ret = readMessageQ(msgQidV,(CS_VOID *)&peticion,SRV_M_WAIT);
    CHECKRET(ret, "validadorHndI:readMessageQ\n");
    TRACE(TRC_FUNCTION_RESULTS,"validadorHndI: Se leyo una petición\n");

    /*
    ** Se borra la bitacora de VALIDACION que pudiera tener asociada el envío, ya que
    ** anteriormente se pudieron haber realizado varios envíos para validacion solo.
    */
    sprintf(sqlcommand,"exec ssa_borraBitacora %d, %d, %d, 'VALIDACION'",
        peticion->folio, peticion->cveInstitucion,peticion->cveSector),
    sprintf(bufferLog,"validadorHndI SQLcommand: %s\n",sqlcommand);
    TRACE(TRC_FUNCTION_ENTER_RETURN,bufferLog);

    /*
    ** Se ejecuta ssa_getEnvioInfo para obtener información del envío,
    ** en particular se debe obtener la información del documento
    ** asociado y la tabla temporal de donde se toman los datos.
    */
    sprintf(sqlcommand,"exec ssa_getEnvioInfo %d, %d, %d",
        peticion->folio, peticion->cveInstitucion,peticion->cveSector),
    sprintf(bufferLog,"validadorHndI: SQLcommand: %s\n",sqlcommand);
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);

    /*
    ** Se borra el historico que pudiera tener asociada el envío, ya que
    ** anteriormente se pudieron haber realizado varios envíos para validacion solo
    */
    sprintf(sqlcommand,"delete HIST_%d_%d where env_folio = %d",
        peticion->documento->idDocto,peticion->documento->version,peticion->folio);
    sprintf(bufferLog,"validadorHndI: SQLcommand: %s\n",sqlcommand);
    TRACE(TRC_FUNCTION_ENTER_RETURN,bufferLog);

    /*
    ** Se ejecuta ssa_getSpValidacion y se guardan en PETICION
    ** los stored procedures de validacion.
    */
    sprintf(sqlcommand,"exec ssa_getSpValidacion %d, %d",
        peticion->documento->idDocto,
        peticion->documento->version),

    /*
    ** Se ejecuta ssa_getValidaCatalogo y se guardan en PETICION
    ** las validaciones contra catalogo
    */
    sprintf(sqlcommand,"exec ssa_getValidaCatalogo %d, %d",
        peticion->documento->idDocto,
        peticion->documento->version);

    /*
    ** Se ejecuta ssa_getExpInd y se guardan en PETICION
    ** las validaciones con expresiones
    */
}

```

```

*/
sprintf(sqlcommand,"exec ssa_getExplnd %d, %d",
        petición->documento->idDocto,
        petición->documento->version);
/*
** Se obtiene el formato de las columnas del documento, para despues poder realizar
** la carga de informacion de la tabla temporal al HISTORICO.
*/
        if (getFormato(conn_ptr,petición->documento) != CS_SUCCEED)
        {
                validadorError(conn_ptr,petición, PROGRAM_FUNC,
                        "ERROR:getFormato()");
                continue;
        }

        /* Status inicial de la validacion */
        status = CS_TRUE;

        /*
        ** Se crea el comando que nos permitira la insercion de la informacion en la tabla temporal en
        ** el HISTORICO. El HISTORICO tiene como nombre HIST_docId_docioVersion
        ** Hay que recordar que el HISTORICO tiene como columnas el folio, la clave de la institucion,
        ** el sector, el numero de renglon y las columnas de la tabla temporal del tipo adecuado
        ** La primera columna de la tabla temporal es una columna numerica que se utilizara como
        ** numero de renglon.
        */
        if((strcmp(petición->documento->columnas.columna[i]->tipo,NUMERIC_TYPE) == 0)
||
0))
        {
                strcat(sqlcommand,"convert(");
                strcat(sqlcommand,petición->documento->columnas.columna[i]->tipo);
                strcat(sqlcommand,"");
                itoa(petición->documento->columnas.columna[i]->longitud,tmp,10);
                strcat(sqlcommand,tmp);
                strcat(sqlcommand,"");
                itoa(petición->documento->columnas.columna[i]->decimales,tmp,10);
                strcat(sqlcommand,tmp);
                strcat(sqlcommand,"), col");
                itoa(i+2,tmp,10);
                strcat(sqlcommand,tmp);
                strcat(sqlcommand,"");
        }
        else
        {
                strcat(sqlcommand,"convert(");
                strcat(sqlcommand,petición->documento->columnas.columna[i]->tipo);
                strcat(sqlcommand," , col");
                itoa(i+2,tmp,10);
                strcat(sqlcommand,tmp);
                strcat(sqlcommand,"");
        }
        } /* for */

        strcat(sqlcommand," from ");
        strcat(sqlcommand,sistemaInfo.mntLogin);
        strcat(sqlcommand,"");
        strcat(sqlcommand,petición->tablaTemporal);
        sprintf(bufferLog,"validadorHnd: %s\n",sqlcommand);
        TRACE(TRC_FUNCTION_INSERTS,bufferLog);

        ret=SQLCommand(conn_ptr,sqlcommand,noElaboraResultados,NULL);

}

/*
** Si no existieron errores
*/

```

```

if(ret== CS_SUCCEED)
{
    sprintf(bufferLog,"validadorHndI: antes validacionCat %d\n",status);
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);

    if (validacionCat(conn_ptr, peticion, &status)!=CS_SUCCEED)
    {
        validadorError(conn_ptr,peticion, PROGRAM_FUNC, "ERROR.validacionCat()");
        continue;
    }

    sprintf(bufferLog,"validadorHndI: antes validacionExpr %d\n",status);
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);

    if (validacionExpr(conn_ptr, peticion, &status)!=CS_SUCCEED)
    {
        validadorError(conn_ptr,peticion, PROGRAM_FUNC, "ERROR.validacionExpr()");
        continue;
    }

    sprintf(bufferLog,"validadorHndI: antes validacionSP %d\n",status);
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);

    if (validacionSP(conn_ptr, peticion, &status)!=CS_SUCCEED)
    {
        validadorError(conn_ptr,peticion, PROGRAM_FUNC, "validacionSP()");
        continue;
    }

    sprintf(bufferLog,"validadorHndI: despues validacionSP %d\n",status);
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);

    /*
    ** Si no existieron errores de validacion.
    */
    if (status == CS_TRUE )
    {
        if (peticion->servicio == SERV_VALIDACION)
        {
            envioSoloValidacion(conn_ptr,peticion);
            continue;
        }

        peticionMnt = srv_alloc(sizeof(PETICION_MNT));
        srv_bzero(peticionMnt,sizeof(PETICION_MNT));

        peticionMnt->tipoCmd = SQL_CMD_NO_RESULTS;
        peticionMnt->respuesta = SIN_RESPUESTA;
        sprintf(sqlcommand,"drop table %s.%s",DB_NAME,peticion->tablaTemporal);
        peticionMnt->comando = srv_alloc(strlen(sqlcommand)+1);
        strcpy(peticionMnt->comando,sqlcommand);

        ret = writeMessageQ(NULL,MNT_Q_NAME,(CS_VOID *)peticionMnt,

        CHECKRET(ret, "validadorHndI:wroteMessageQ");
        strcpy(peticion->status,"VO");

        /*
        ** Escribimos en la bitacora el
        ** hecho de que la validacion
        ** haya resultado exitosa.
        ** y cambiamos el status del envio.
        */
        sprintf(sqlcommand, "exec ssa_creaBitacora %d, %d, %d,
'VALIDACION','%s',0,0,NULL",
        peticion->folio,
        peticion->cveInstitucion,

```

```

        petición->cveSector,
        VALIDACION_EXITOSA);

    sprintf(bufferLog,"validadorHndI: %s\n", sqlcommand);
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);

    if (SQLCommand(conn_ptr,sqlcommand,noElaboraResultados,NULL) !=
CS_SUCCEED)
    {
        validadorError(conn_ptr,petición, SQL_COMMAND,
            "SQLCommand(ssa_creaBitacora)");
        continue;
    }
}
/*
** Se determina si el documento debe pasar por el proceso de integracion o
** debe quedarse simplemente en el HISTORICO, para que las autoridades analicen
** la informacion. Requerimiento de BANXICO.
*/
    sprintf(sqlcommand,"exec ssa_getIntegracionhabilitada %d, %d, %d, %d",
        petición->documento->dDocto,
        petición->cveInstitucion,
        petición->cveSector,
        petición->areald);

    sprintf(bufferLog,"validadorHndI: %s\n", sqlcommand);
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);

    sprintf(bufferLog,"validadorHndI: integraHabil %d\n", integraHabil);
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);

    if (integraHabil == 1)
        strcpy(statusPetición,"VO");
    else
        strcpy(statusPetición,"SV");

    sprintf(sqlcommand,"exec ssa_setStatusEnvio %d, %d, %d, %s",
        petición->folio,
        petición->cveInstitucion,
        petición->cveSector,
        statusPetición);

    sprintf(bufferLog,"validadorHndI: %s\n", sqlcommand);
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);
}
/*
** SE escribe en INTEGRA_Q_NAME si el documento tiene habilitada la opcion de integracion.
*/
    if (integraHabil == 1)
    {
        ret =
wntMessageQ(&msgQIdI,INTEGRADOR_Q_NAME,petición,SRV_M_NOWAIT);
        CHECKRET(ret,"validadorHndI.writeMessageQ");
    }
    else if ((petición->tipo & PETICION_SINCRONA) == PETICION_SINCRONA)
    {
        /* Envía el resultado de la petición al f'tread de conexión */

        resultInfo = srv_alloc(sizeof(RESULT_INFO));
        srv_bzero(resultInfo,sizeof(RESULT_INFO));

        resultInfo->msg = srv_alloc(strlen(PROCESO_LOCAL_EXITOSO)+1);
        strcpy(resultInfo->msg,PROCESO_LOCAL_EXITOSO);

        resultInfo->tipoMsg = FINAL_MSG;

        ret=wntMessageQ(&petición->thread->msgQId,NULL,(CS_VOID
*)resultInfo, SRV_M_NOWAIT);

```



```

        desalojaPeticion(peticion);

        CHECKRET(ret,"validadorHndI: writeMessageQIn");
    }
}
else
{
    /*
    ** Si existieron errores de validacion.
    */

    /*
    ** Se borra la informacion que se habia cargado al HISTORICO.
    */
    sprintf(sqlcommand,"DELETE %s..HIST_ %d_ %d "
            " WHERE env_folio = %d AND "
            " ins_clave = %d AND "
            " ins_sector = %d ",
            HIST_BD,
            peticion->documento->idDocto,
            peticion->documento->version,
            peticion->folio,
            peticion->cveInstitucion,
            peticion->cveSector);
    sprintf(bufferLog,"validadorHndI: %s\n", sqlcommand);
    TRACE(TRC_FUNCTION_RESULTS,bufferLog);

    SQLCommand(conn_ptr,sqlcommand,noElaboraResultados,NULL);

    validadorError(conn_ptr,peticion, VALIDACION_PROCC,
        "Validacion fallida");
    continue;
}
}
else
{
    /*
    ** Si existieron errores
    */

    validadorError(conn_ptr,peticion, VALIDACION_PROCC,
        "Creacion del HISTORICO fallida");
    continue;
}
}

/*
** Cierra la conexion hacia el SQL SERVER
*/
ret = ct_close(conn_ptr,CS_UNUSED);
CHECKRET(ret,"validadorHndI: ct_close\n");
ret = ct_con_drop(conn_ptr);
CHECKRET(ret,"validadorHndI ct_con_drop\n");

sprintf(bufferLog,"validadorHndI: FUNCTION RETURN (CS_SUCCEED)\n");
TRACE(TRC_FUNCTION_ENTER_RETURN,bufferLog);

return CS_SUCCEED;

```

Otro de los procesos que realiza el *middleware* es el de la integración de la información enviada por las Entidades Supervisadas a la Entidad Normativa en un repositorio central o *data marts* definidos en la *metadata*. Este proceso es realizado una vez que la información ha sido depurada o validada con éxito. A continuación se muestra el listado de este proceso, donde se observará la integración que se realiza de la información ya validada a los repositorios centrales.

```

.....
..

```

```

** integrador.c
** Descripción:
** Este archivo contiene el código del thread integrador
**
...../
#include "integrador.h"
...../
**
**      P R O T O T I P O S
**
...../
/* CS_RETCODE integradorHndl( SRV_PROC      *srv_proc ), ver osSA.h */

CS_RETCODE enviaAServidores ( CS_COMMAND      *, PETICION      *, CS_INT),
CS_RETCODE integra_peticion ( CS_CONNECTION   *, PETICION      *);
CS_RETCODE envia_bitacora   ( CS_CONNECTION   *, PETICION      *),
CS_RETCODE envia_envio      ( CS_CONNECTION   *, PETICION      *),
CS_RETCODE envia_historico  ( CS_CONNECTION   *, PETICION      *),
CS_RETCODE bitacora_pendiente ( CS_CONNECTION  *, PETICION      *),
CS_RETCODE elabora_envioPen ( CS_COMMAND      *, CS_VOID      *, CS_INT);
CS_VOID rollback            ( INFO_DYNAMIC    *, PETICION      *);
CS_RETCODE inserta_datos    ( CS_CONNECTION   *, PETICION      *);

/...../
**
** integradorHndl
**
** Descripción:
** Thread que realiza la integración de los documentos en las bases
** de datos operacionales
**
...../
CS_RETCODE integradorHndl( SRV_PROC      *srv_proc )
{
    CS_RETCODE      ret;
    CS_CONNECTION *conn_ptr;
    CS_INT          idConn,
    CS_INT          info,
    SRV_OBJID      msgQid,
    PETICION        *recvMsg,
    RESULT_INFO    *info_msg,
    CS_CHAR         sqlstrng[MAX_LEN_MSG_LOG],
    CS_CHAR         bufferLog[MAX_LEN_MSG_LOG],

    TRACE(TRC_FUNCTION_ENTER_RETURN,"integrador: FUNCTION_ENTER >>> \n"),

    srv_proc = SRV_CURPROC;

    /* Abre una conexión permanente al SQL Server */

    if (sistemaInfo.tipoSistema == SERVER_LOCAL)
    { /* conexión con bulk copy habilitado */
        while ((ret=obtieneConexion(&conexionesIntegrador,&conn_ptr,sistemaInfo.sqlServerName,
            sistemaInfo.integradorLogin,sistemaInfo.integradorPwd,
            &idConn,TRUE)) == CS_BUSY);
        CHECKRET(ret,"integrador obtieneConexion\n");
    }
    else /* conexión con bulk copy deshabilitado */
    {
        while ((ret=obtieneConexion(&conexionesIntegrador,&conn_ptr,sistemaInfo.sqlServerName,
            sistemaInfo.integradorLogin,sistemaInfo.integradorPwd,
            &idConn,FALSE)) == CS_BUSY);
        CHECKRET(ret,"integrador obtieneConexion\n");
    }

    ret = srv_getobjid(SRV_C_MQUEUE,INTEGRADOR_Q_NAME,CS_NULLTERM,&msgQid,&info),
    CHECKRET(ret,"integrador srv_getobjid\n");
}

```

```

while (1 == 1)
{
    ret = readMessageQ(msgQId,(CS_VOID *)&recvMsg,SRV_M_WAIT);
    CHECKRET(ret,"integrador: readMessageQ\n");

    if (recvMsg->folio == 0) /* Peticion del programa cliente temporizador para la elaboracion de envio
pendientes */
    {
        if (sistemaInfo.tipoSistema == SERVER_LOCAL)
        {
            /* ELABORA PENDIENTES */

            ret = elabora_pendientes(conn_ptr);
            CHECKRET(ret,"integrador: elabora_pendientes\n");

        }
    }
    else
    { /* ELABORA LA PETICION RECIBIDA */

        if (sistemaInfo.tipoSistema == SERVER_CENTRAL)
        {
            ret = integra_peticon(conn_ptr,recvMsg);
            if (ret == CS_FAIL)
            {
                /*
                ** Escribimos en la bitacora el
                ** hecho de que la integracion
                ** no haya resultado exitosa.
                */
                sprintf(sqlstring, "exec ssa_creaBitacora %d, %d, %d,
'INTEGRACION','%s',0,0,NULL",
                    recvMsg->folio,
                    recvMsg->cveInstitucion,
                    recvMsg->cveSector,
                    INTEGRACION_NO_EXITOSA);
                sprintf(bufferLog,"integrador: %s\n",sqlstring);
                TRACE(TRC_FUNCTION_RESULTS,bufferLog);

                ret=SQLCommand(conn_ptr,sqlstring,noElaboraResultados,NULL);
                CHECKRET(ret,"integrador: SQLCommand: exec ssa_creaBitacora\n");

                sprintf(sqlstring,
                    "exec ssa_setStatusEnvio %d,%d,%d,'IF",
                    recvMsg->folio,
                    recvMsg->cveInstitucion,
                    recvMsg->cveSector);
            }
            else
            {
                /*
                ** Escribimos en la bitacora el
                ** hecho de que la integracion
                ** haya resultado exitosa.
                */
                sprintf(sqlstring, "exec ssa_creaBitacora %d, %d, %d,
'INTEGRACION','%s',0,0,NULL",
                    recvMsg->folio,
                    recvMsg->cveInstitucion,
                    recvMsg->cveSector,
                    INTEGRACION_EXITOSA);

                sprintf(bufferLog,"integrador: %s\n",sqlstring);
                TRACE(TRC_FUNCTION_RESULTS,bufferLog);
            }
        }
    }
}

```

```

        sprintf(sqlstring,
            "exec ssa_setStatusEnvio %d,%d,%d,%d",
            recvMsg->folio,
            recvMsg->cveInstitucion,
            recvMsg->cveSector);
    }
}
else
{
    /* Envía el documento a cada servidor central indicando por el result set de
    ssa_getServidorApp */

    sprintf(sqlstring,"exec ssa_getServidoresApp %d,%d,%d,%d",
        recvMsg->documento->idDocto,recvMsg->cveInstitucion,
        recvMsg->cveSector,recvMsg->areald);

    if (ret == CS_FAIL)
    {
        /*
        ** Escribimos en la bitacora el
        ** hecho de que la integracion
        ** no haya resultado exitosa.
        */
        sprintf(sqlstring, "exec ssa_creaBitacora %d, %d, %d
            recvMsg->folio,
            recvMsg->cveInstitucion,
            recvMsg->cveSector,
            PROCESO_LOCAL_NO_EXITOSO);

        sprintf(bufferLog,"integrador. %s\n",sqlstring);
        TRACE(TRC_FUNCTION_RESULTS,bufferLog);
        TRACE(TRC_FUNCTION_RESULTS,
            "integrador: FUNCTION_RESULT integracion local fa.l <<< \n");
    }
    else
    {
        /*
        ** Escribimos en la bitacora el
        ** hecho de que la integracion
        ** haya resultado exitosa.
        */
        sprintf(sqlstring,"exec ssa_creaBitacora %d, %d, %d
            recvMsg->folio,
            recvMsg->cveInstitucion,
            recvMsg->cveSector,
            PROCESO_LOCAL_EXITOSO);

        sprintf(bufferLog,"integrador. %s\n",sqlstring);
        TRACE(TRC_FUNCTION_RESULTS,bufferLog);
    }
}

if ((recvMsg->tipo & PETICION_SINCRONA) == PETICION_SINCRONA)
if (recvMsg->thread->msgQId > 0)
{ /* Envía el resultado de la peticion al thread de conexion */

    info_msg = srv_alloc(sizeof(RESULT_INFO)),
    srv_bzero(info_msg,sizeof(RESULT_INFO)),

    info_msg->msg = srv_alloc(strlen(PROCESO_LOCAL_EXITOSO)+1)
    strcpy(info_msg->msg,PROCESO_LOCAL_EXITOSO),

    info_msg->tipoMsg = FINAL_MSG,

```

```

ret=writeMessageQ(&recvMsg->thread->msgQId,NULL,(CS_VOID *)info_msg,
SRV_M_NOWAIT);
CHECKRET(ret,"Integrador: writeMessageQn");
}
desalojaPetición(recvMsg);
}
srv_yield();
}
TRACE(TRC_FUNCTION_ENTER_RETURN,"Integrador: FUNCTION_RETURN (CS_SUCCEED) <<< \n");
return CS_SUCCEED;
} /* fin de integrador */

```

7.2 Fase de pruebas

La prueba de software es un elemento crítico para garantizar la calidad del software y representa una última revisión de las especificaciones del diseño y la codificación.

El proceso de prueba al igual que el de la programación debe avanzar en etapas, siendo cada una de ellas la continuación lógica de la etapa anterior.

- **Prueba de funciones.** La prueba de funciones ó de unidades es el nivel básico en donde se prueban las funciones que componen un módulo para garantizar que operarán de manera correcta. En un sistema de diseño apropiado, cada función debe tener una sola especificación definida con claridad. No debe existir dificultad al diseñar casos de prueba para asegurar que las funciones cumplan con su especificación. Las funciones no deben depender de otras de su mismo nivel, para posibilitar la prueba de cada función como una entidad aislada, sin la presencia de otras funciones.
- **Prueba del sistema.** La prueba del sistema (a veces llamada prueba de integración) se lleva a cabo cuando se integran los subsistemas para conformar el sistema completo. En esta etapa, el proceso de prueba detecta los errores en el diseño y la codificación. También confirma que el sistema proporcione las funciones específicas y características dinámicas planteadas en la definición de requerimientos.

Prueba Real

Una de las pruebas realizadas dentro del Sistema es la simulación de un envío de información de la Entidad Supervisada hacia la Entidad Central. La siguiente prueba realizada utiliza la configuración de un CLIENTE conectado a un *middleware* local. A continuación se describen los pasos para realizar un envío; así como, la funcionalidad del sistema.

El CLIENTE es el medio de comunicación entre Entidad Supervisada y la Entidad Normativa. Para acceder a la aplicación CLIENTE se requiere indicar la clave y contraseña del usuario de la Entidad Supervisada definida en la *metadata*. En la figura 7.1 se muestra el acceso a la aplicación, además de la configuración de conexión.

Acceso

CLIENTE

Version 1.20
Septiembre 1999

USUARIO : 0105300

CONTRASEÑA :

SERVIDOR : STI04105S

IP ADDRESS: 172.16.205.130 SQL: STI04105S

HOSTNAME: pas SSA: SSA.NT

✓ Aceptar ✗ Cancelar

Figura 7.1 Acceso al CLIENTE

Una vez que el usuario ha sido validado, se le mostrará la Ventana Principal del CLIENTE, posteriormente se podrá elegir la tarea que se quiere realizar. En la figura 7.2 se muestra la importación manual, donde se observan los diferentes grupos de documentos existentes al igual de los documentos pertenecientes a cada uno de estos. Para poder realizar esta importación se deberá indicar el Año y Periodo que reporta la Entidad Supervisada.

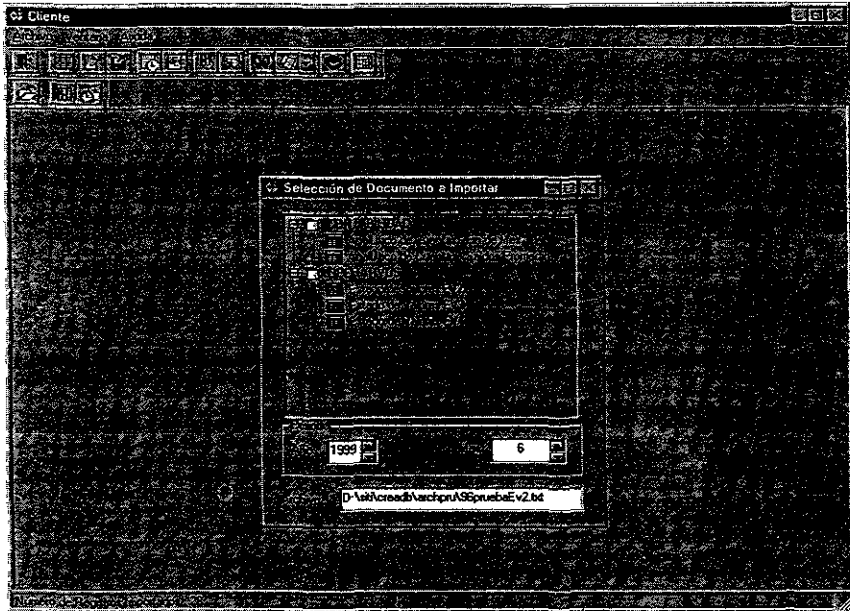


Figura 7.2 Importación manual.

Una vez realizado este proceso con éxito, estaremos listos para la filtración y validación del documento que la Entidad Supervisada reporta a la Entidad Normativa. En la figura 7.3 veremos la lista de documentos posibles a validar; recordaremos que el proceso de validación se encuentra definido para cada documento en la *metadata* (en las tablas de validacion, validacion_catalogo, sp_validacion y formula_validfacion), el cual realiza la validación contra catálogos, validación por *stored procedure* o bien por fórmulas.

En este momento el middleware recibe la petición de validación y es encolada en la cola del thread de validación (VALIDADOR_Q), si el validador está desocupado toma la petición y comienza con su tarea.

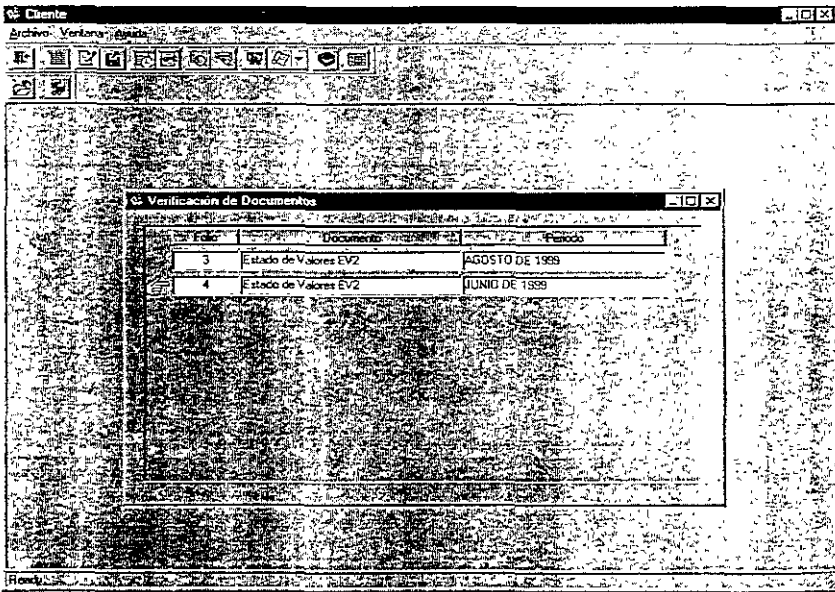


Figura 7.3 Validación de un documento.

Si la validación fue correcta, aparecerá un mensaje de que el documento ha sido validado con éxito como es que se muestra en la figura 7.4. El estatus en el que se encuentra el documento enviado por la Entidad Supervisada se registra en la *metadata* (en la tabla de envío) de tal forma que podemos apreciar si el documento de cierta Entidad Supervisada ya fue enviado, así como la fecha en la que realizó este proceso. Para este caso el estatus registrado sería validación exitosa (VO).

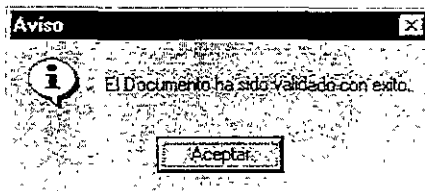


Figura 7.4 Validación de un documento con éxito

Si la validación no fue correcta, aparecerá una ventana de seguimiento de documento, donde pondremos observar los errores en que columna o renglón se encuentra (figura 7.5). Esta lista de errores de validación es registrada en la *metadata* (en la tabla de bitácora) y el estatus en el que se encuentra el documento es validación fallida (VF).

Seguimiento de Documento

Sector : 17 Folio : 4

Clave : 13

Numero	Fase	Mensaje	Registro	Columna	Fecha y Hora
	CARGA	El documento Estado de Valores EV2 fue importado manualmente.	0	0	8/26/99 13:54:32
	CARGA	El documento Estado de Valores EV2 esta siendo capturado manualmente.	0	0	10/14/99 22:25:15
	VALIDACION	EV20103: Institucion inexistente(10201082)	1	1	10/14/99 22:41:53
	VALIDACION	EV20502: Moneda no existe en catalogo	1	5	10/14/99 22:41:53
	VALIDACION	EV20602: TIPOV no existe en Catalogo	1	6	10/14/99 22:41:53
	VALIDACION	EV20603: Tipo-Valor No corresponde al EV2:CM.	1	6	10/14/99 22:41:53
	VALIDACION	EV20702: CLAVEV No existe en Catalogo	1	7	10/14/99 22:41:53
	VALIDACION	EV20703: Clave-Valor No corresponde al EV2:52.	1	7	10/14/99 22:41:53
	VALIDACION	EV20103: Institucion inexistente(10201082)	2	1	10/14/99 22:41:53
	VALIDACION	EV20502: Moneda no existe en catalogo	2	5	10/14/99 22:41:53
	VALIDACION	EV20602: TIPOV no existe en Catalogo	2	6	10/14/99 22:41:53
	VALIDACION	EV20603: Tipo-Valor No corresponde al EV2:CM	2	6	10/14/99 22:41:53

Página 1 de 15

Figura 7.5 Validación de un documento fallido.

Para este caso en el que el proceso de validación resultó fallido el documento se tendrá que modificar, esto se puede realizar en el proceso de Captura y modificación del documento. Un documento se puede validar tantas veces como se requiera siempre y cuando no se haya enviado a la Entidad Normativa. En la Figura 7.6 se muestra la captura del documento para su modificación.

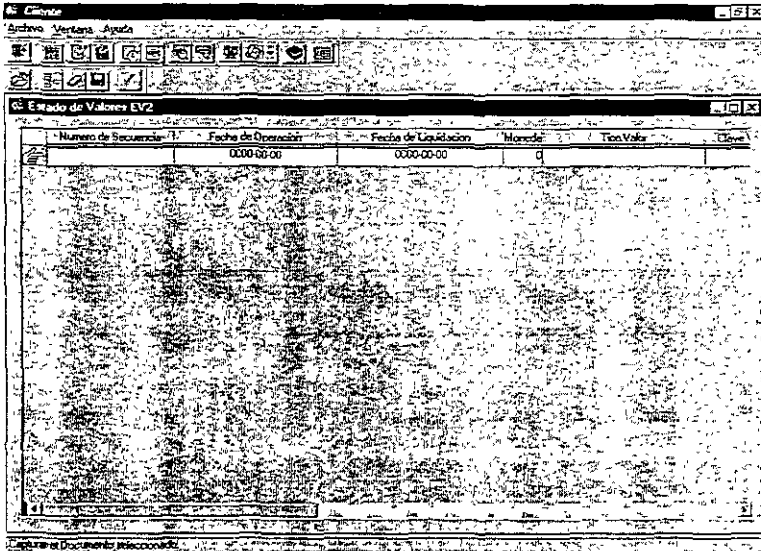


Figura 7.6 Captura del Documento.

Cuando el documento ha sido validado con éxito, el siguiente paso es la integración de éste hacia el repositorio Central o *data marts*. Este proceso es el envío de la Entidad Supervisada a la Entidad Normativa. En la figura 7.7 se muestra el envío de la información .

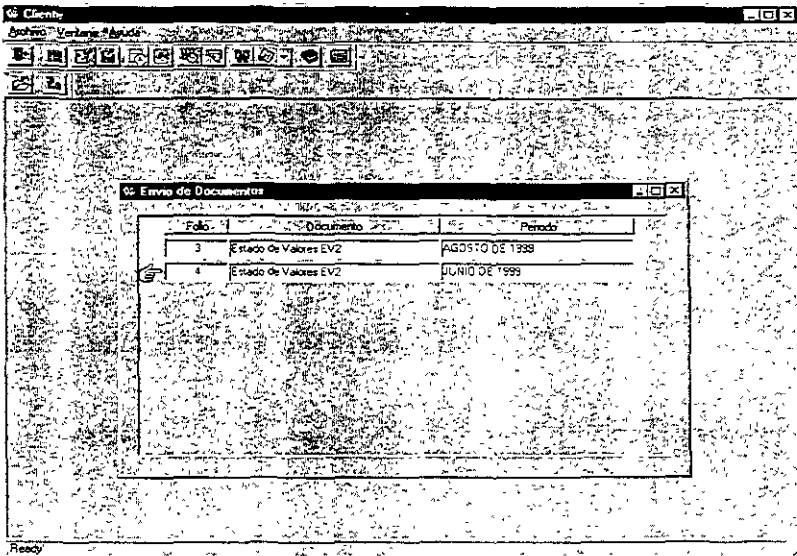


Figura 7.7 Integración del documento enviado por la Entidad Supervisada a la Entidad Normativa.

Si la integración de la información al repositorio Central de la Entidad Normativa no tuvo mayor problema, se generará un acuse de recibo, que es el comprobante de la Entidad Supervisada para justificar que ya realizó su envío de información. En la figura 7.8 se observan los procesos realizados por la Entidad Supervisada para el envío de la información a la Entidad Normativa.

Acuse de Recibo 13 - DE CASA DE

Folio : 4
 Documento : Estado de Valores EV2
 Correspondiente : JUNIO DE 1999
 Fecha de Recepcion : 8/26/99 13:54:29
 Usuario: Usuario 1 de DE

Fase	Mensaje	Fecha y Hora
CARGA	El documento Estado de Valores EV2 fue importado manualmente.	26/08/1999 13:54:29
CARGA	El documento Estado de Valores EV2 esta siendo capturado manualmente	14/10/1999 22:27:46
VALIDACION	SSA_CENTRAL. El documento ha sido validado con éxito. Regs=27	14/10/1999 22:27:46
INTEGRACION	SSA_CENTRAL. El documento no fue integrado correctamente	14/10/1999 22:27:46
CARGA	El documento Estado de Valores EV2 esta siendo capturado manualmente.	14/10/1999 22:27:46

Figura 7.8 Acuse de recibo.

Este Acuse representa el término del proceso. Una vez integrada la información se podrá realizar una consulta al HISTÓRICO como se muestra en la figura 7.9.

Institucion	No Secuencia	Fecha Operacion	Fecha Liquidacion	Moneda	Tipo Valor	Clave Valor	Empresa	Seno	Posicion	Clave Op	Ora
1020108298C	1	3/27/98 00:00:00	3/31/98 00:00:00		QCM	52	ACCIAR	B	TS	COL	
1020108298C	2	3/27/98 00:00:00	3/31/98 00:00:00		QCM	52	ACCIPAT	B	TS	COL	
1020108298C	3	3/27/98 00:00:00	3/31/98 00:00:00		QCM	52	ACTIVAL	B	TS	COL	
1020108298C	4	3/27/98 00:00:00	3/31/98 00:00:00		QCM	1	AHMSA	**	TS	COL	
1020108298C	5	3/27/98 00:00:00	3/31/98 00:00:00		QCM	52	BANABUR	A	TS	COG	
1020108298C	6	3/27/98 00:00:00	3/31/98 00:00:00		QCM	52	BANAFI2	A	TS	COG	
1020108298C	7	3/27/98 00:00:00	3/31/98 00:00:00		QCM	52	BANAFIN	IA	TS	COG	
1020108298C	8	3/27/98 00:00:00	3/31/98 00:00:00		QCM	1	BMSB	A	TS	COL	
1020108298C	9	3/27/98 00:00:00	3/31/98 00:00:00		QCM	51	BNMICORP	IA	TS	COF	
1020108298C	10	3/27/98 00:00:00	3/31/98 00:00:00		QCM	1	ICAMESA	-	TS	COL	
1020108298C	11	3/27/98 00:00:00	3/31/98 00:00:00		QCM	1	CEMEX	A	TS	COL	
1020108298C	12	3/27/98 00:00:00	3/31/98 00:00:00		QCM	1	CEMEX	CPO	TS	COL	
1020108298C	13	3/27/98 00:00:00	3/31/98 00:00:00		QCM	1	CERAMIC	UB	TS	COL	
1020108298C	14	3/27/98 00:00:00	3/31/98 00:00:00		QCM	1	CERAMIC	ULD	TS	COL	
1020108298C	15	3/27/98 00:00:00	3/31/98 00:00:00		QCM	1	CIFRA	IV	TS	COL	
1020108298C	16	3/27/98 00:00:00	3/31/98 00:00:00		QCM	1	CIFRA	IC	TS	COL	
1020108298C	17	3/27/98 00:00:00	4/1/98 00:00:00		QCM	1	CIFRA	IV	TS	COL	
1020108298C	18	3/31/98 00:00:00	4/2/98 00:00:00		QCM	1	CIFRA	V	TS	COL	

Figura 7.9 Consulta del Histórico.

Todos estos procesos realizados además de que son registrados en la *metadata* como hemos mencionado, también son registrados en el log del *middleware*. A continuación se muestra parte de este archivo donde veremos incluso la configuración de este.

```

<<<<<<SSA_LOCAL>>>>>>>><<<<1.1.5>>>><<<<<<May 25 1999>>>>>>>>
May 25 14:19:45 1999: startHndi: FUNCTION_ENTER >>>
.....Configuracion Inicial.....
Open Server Name:      SSA_LOCAL
Institucion:
Tipo Sistema:         SERVER LOCAL
SQL SERVER:           0410SS
SERVIDOR APLICATIVO CNBV: SRV_APLICATIVO
Archivo de log:       LOCAL.log
Archivo de configuracion: LOCAL.cfg
Numero max de conexiones: 40
Numero VALIDADOR_T:  1
Numero INTEGRADOR_T: 1
Nivel de Trace:      255
May 25 14:19:45 1999: --- Servidor Aplicativo Operando ---
May 25 14:19:45 1999: Inicializacion de EVENT HANDLERS:
May 25 14:19:45 1999: SRV_CONNECT
May 25 14:19:45 1999: SRV_DISCONNECT
May 25 14:19:45 1999: SRV_LANGUAGE
May 25 14:19:45 1999: SRV_BULK
May 25 14:19:45 1999: Estructuras de Datos inicializadas
May 25 14:19:45 1999: Creacion PROCEDIMIENTOS REGISTRADOS.
May 25 14:19:45 1999: rp_ossa_shutdown
May 25 14:19:45 1999: rp_accesalInfo
May 25 14:19:45 1999: rp_activaPeticon
May 25 14:19:45 1999: rp_bitacoraMsg
May 25 14:19:45 1999: rp_integraPeticon
May 25 14:19:45 1999: rp_comoilaFormula
May 25 14:19:45 1999: rp_procDocumento
    
```

```

May 25 14:19:45 1999: rp_timer
May 25 14:19:45 1999: rp_reenviaHistorico
May 25 14:19:45 1999: rp_updReglasIntegracion
May 25 14:19:45 1999: rp_monitor
May 25 14:19:45 1999: rp_monitorMsg
May 25 14:19:45 1999: rp_conexion
May 25 14:19:45 1999: rp_config
May 25 14:19:45 1999: insertThread: TRC_FUNCTION_ENTER >>>
May 25 14:19:45 1999: insertThread: TRC_FUNCTION_RETURN (TRUE) <<<
May 25 14:19:45 1999: insertThread: TRC_FUNCTION_ENTER >>>
May 25 14:19:45 1999: insertThread: TRC_FUNCTION_RETURN (TRUE) <<<
May 25 14:19:45 1999: insertThread: TRC_FUNCTION_ENTER >>>
May 25 14:19:45 1999: insertThread: TRC_FUNCTION_RETURN (TRUE) <<<
May 25 14:19:45 1999: insertThread: TRC_FUNCTION_ENTER >>>
May 25 14:19:45 1999: insertThread: TRC_FUNCTION_RETURN (TRUE) <<<
May 25 14:19:45 1999: Creacion de MESSAGES QUEUEUS:
May 25 14:19:45 1999: CreateMessageQ: FUNCTION ENTER >>> [ACCES_Q]
May 25 14:19:45 1999: CreateMessageQ:<<< FUNCTION RETURN (CS_SUCCEED) OBJID [3]
May 25 14:19:45 1999: ACCESS_Q
May 25 14:19:45 1999: CreateMessageQ: FUNCTION ENTER >>> [VALIDADOR_Q]
May 25 14:19:45 1999: CreateMessageQ:<<< FUNCTION RETURN (CS_SUCCEED) OBJID [4]
May 25 14:19:45 1999: VALIDA_Q
May 25 14:19:45 1999: CreateMessageQ: FUNCTION ENTER >>> [INTEGRADOR_Q]
May 25 14:19:45 1999: CreateMessageQ:<<< FUNCTION RETURN (CS_SUCCEED) OBJID [5]
May 25 14:19:45 1999: INTEGRADOR_Q
May 25 14:19:45 1999: conexionMutex
May 25 14:19:45 1999: obteneMutex
May 25 14:19:45 1999: elaboraPen
May 25 14:19:45 1999: startHndl: FUNCTION_RETURN (CS_SUCCEED) <<<
May 25 14:19:45 1999: startHndl: 'JOC -SSSA_LOCAL -DSIT04105S -CLOCAL.cfg
May 25 14:19:45 1999: accessHndl: FUNCTION ENTER >>>
May 25 14:19:45 1999: mntHndl: FUNCTION ENTER >>>
May 25 14:19:45 1999: validadorHndl: FUNCTION ENTER >>>
May 25 14:19:45 1999: integrador: FUNCTION_ENTER >>>
May 25 14:19:45 1999: ReadMessageQ: FUNCTION ENTER OBJID [5]>>>
May 25 15:17:49 1999: connectHndl: FUNCTION_ENTER >>>
May 25 15:17:49 1999: >>>>>> RPC TYPE 2, HOST[] LOGIN[01059001]
May 25 15:17:49 1999: CreateMessageQ: FUNCTION ENTER >>> [CONNECT_Q12]
May 25 15:17:49 1999: CreateMessageQ:<<< FUNCTION RETURN (CS_SUCCEED) OBJID [13]
May 25 15:17:49 1999: insertThread: TRC_FUNCTION_ENTER >>>
May 25 15:17:49 1999: insertThread: TRC_FUNCTION_RETURN (TRUE) <<<
May 25 15:17:49 1999: connectHandler: FUNCTION_RETURN (CS_SUCCEED) <<<
May 25 15:17:49 1999: languageHndl: FUNCTION_ENTER >>>
May 25 15:17:49 1999: languageHndl: comando ->select @@langid<-
May 25 15:17 49 1999: languageHndl: FUNCTION_RETURN >>>
May 25 15:17:49 1999: languageHndl: FUNCTION_ENTER >>>
May 25 15:17 49 1999: languageHndl comando ->select dateformat from master.dbo.syslanguages where langid = 999<-
May 25 15:17:49 1999: languageHndl: FUNCTION_RETURN >>>
May 25 15:17:49 1999: languageHndl: FUNCTION_ENTER >>>
May 25 15:17:49 1999: languageHndl: comando ->select @@@ncharsize<-
May 25 15:17:49 1999 languageHndl: FUNCTION_RETURN >>>
May 25 15:17:49 1999: languageHndl: FUNCTION_ENTER >>>
May 25 15:17:49 1999. languageHndl: comando ->use dbssa_metadata <-
May 25 15:17:49 1999: languageHndl. FUNCTION_RETURN >>>
May 25 15:17:49 1999: languageHndl: FUNCTION_ENTER >>>
May 25 15:17:49 1999: languageHndl. comando ->select USER_NAME(uid) from dbo.sysusers where SUSER_NAME(suid)
= '01059001'<-
May 25 15:17:49 1999: languageHndl: FUNCTION_RETURN >>>
May 25 15:17:49 1999: languageHndl: FUNCTION_ENTER >>>
May 25 15:17:49 1999 languageHndl: comando ->select USER_NAME(u.uid) from dbo.sysusers u, dbo.sysalternates
where SUSER_NAME(a.suid) = '01059001' and SUSER_NAME(u.suid) = SUSER_NAME(altsuid)<-
May 25 15:17:49 1999: languageHndl: FUNCTION_RETURN >>>
May 25 15:17:49 1999: rp_procDocumento: FUNCTION ENTER >>>
May 25 15:17:49 1999: EnvFolio = 16, InsClave = 59 InsSector = 1
TipoRes = S:6, TipoServ =V Usuario.01059001
May 25 15:17:49 1999: WriteMessageQ: FUNCTION ENTER>>>
May 25 15:17:49 1999: WriteMessageQ: -QUEUE Flags(32) [VALIDADOR_Q]
May 25 15:17 49 1999: writeMessageQ: <<< FUNCTION RETURN (CS_SUCCEED) OBJ ID[4][VALIDADOR_Q]
May 25 15:17:49 1999: ReadMessageQ. FUNCTION ENTER OBJID [13]>>>

```

```

May 25 15:17:49 1999: QUEUE Flags(16) [CONNECT_Q12]
May 25 15:17:49 1999: ReadMessageQ Se leyo de [VALIDADOR_Q]
May 25 15:17:49 1999: ReadMessageQ <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:49 1999: validadorHndI ---->@@@VALIDADOR_T01 SE LEYO UNA PETICION
May 25 15:17:49 1999: SQLCommand FUNCTION_ENTER >>>
[exec ssa_borraBitacora 16, 59, 1, 'VALIDACION']
May 25 15:17:49 1999: envia_comando FUNCTION_ENTER >>>
May 25 15:17:49 1999: handle_results: FUNCTION_ENTER >>>
May 25 15:17:49 1999: handle_results: CS_STATUS_RESULT ---
May 25 15:17:49 1999: noElaboraResultados FUNCTION_ENTER >>>
May 25 15:17:49 1999: bind_columns FUNCTION_ENTER >>>
May 25 15:17:49 1999: bind_columns <<< FUNCTION_RETURN
May 25 15:17:49 1999: getRenglon: (CS_SUCCEEDED) <<<
May 25 15:17:49 1999: getRenglon: (CS_SUCCEEDED) <<<
May 25 15:17:49 1999: desaloja_renglon: FUNCTION_ENTER >>>
May 25 15:17:49 1999: desaloja_renglon: <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:49 1999: noElaboraResultados <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:49 1999: handle_results: CS_CMD_SUCCEEDED ---
May 25 15:17:49 1999: handle_results: CS_CMD_DONE ---
May 25 15:17:49 1999: handle_results: FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:49 1999: termina_comando FUNCTION_ENTER (CS_SUCCEEDED)
May 25 15:17:49 1999: SQLCommand <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:49 1999: SQLCommand FUNCTION_ENTER >>>
[exec ssa_getEnvioInfo 16, 59, 1]
May 25 15:17:49 1999: envia_comando FUNCTION_ENTER >>>
May 25 15:17:49 1999: handle_results: FUNCTION_ENTER >>>
May 25 15:17:49 1999: handle_results: CS_ROW_RESULT ---
May 25 15:17:49 1999: getEnvioInfo FUNCTION_ENTER >>>
May 25 15:17:49 1999: bind_columns FUNCTION_ENTER >>>
May 25 15:17:49 1999: bind_columns <<< FUNCTION_RETURN
May 25 15:17:49 1999: getRenglon: (CS_SUCCEEDED) <<<
May 25 15:17:49 1999: getRenglon: (CS_SUCCEEDED) <<<
May 25 15:17:49 1999: desaloja_renglon: FUNCTION_ENTER >>>
May 25 15:17:55 1999: obtieneCount: FUNCTION_RETURN (CS_SUCCEEDED) <<<
May 25 15:17:55 1999: handle_results: CS_CMD_DONE ---
May 25 15:17:55 1999: handle_results: FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:55 1999: termina_comando FUNCTION_ENTER (CS_SUCCEEDED)
May 25 15:17:55 1999: SQLCommand <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:55 1999: validacionCat: numRengHist 12 numRengTmp 12
May 25 15:17:55 1999: SQLCommand FUNCTION_ENTER >>>
[if object_id("#tmp") is not null drop table #tmp]
May 25 15:17:55 1999: envia_comando FUNCTION_ENTER >>>
May 25 15:17:55 1999: termina_comando FUNCTION_ENTER (CS_SUCCEEDED)
May 25 15:17:55 1999: SQLCommand <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:55 1999: validacionCat: SELECT DISTINCT h.renglon, h.env_folio, h.ins_sector, h.ins_clave INTO #tmp
FROM HIST_1_1 h, dbssa_catalogos cuentas c WHERE h.env_folio = 16 AND h.ins_sector = 1 AND h.ins_clave = 59 AND
h.s_cuenta = c.s_cuenta
May 25 15:17:55 1999: SQLCommand FUNCTION_ENTER >>>
[SELECT DISTINCT h.renglon, h.env_folio, h.ins_sector, h.ins_clave INTO #tmp FROM HIST_1_1 h,
dbssa_catalogos..cuentas c WHERE h.env_folio = 16 AND h.ins_sector = 1 AND h.ins_clave = 59 AND h.s_cuenta =
c.s_cuenta]
May 25 15:17:55 1999: envia_comando FUNCTION_ENTER >>>
May 25 15:17:58 1999: termina_comando FUNCTION_ENTER (CS_SUCCEEDED)
May 25 15:17:58 1999: SQLCommand <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:58 1999: SQLCommand FUNCTION_ENTER >>>
[SELECT count(*) FROM HIST_1_1 WHERE env_folio = 16 AND ins_sector = 1 AND ins_clave = 59]
May 25 15:17:58 1999: envia_comando FUNCTION_ENTER >>>
May 25 15:17:58 1999: handle_results: FUNCTION_ENTER >>>
May 25 15:17:58 1999: handle_results: CS_ROW_RESULT ---
May 25 15:17:58 1999: obtieneCount: FUNCTION_ENTER >>>
May 25 15:17:58 1999: bind_columns FUNCTION_ENTER >>>
May 25 15:17:58 1999: bind_columns <<< FUNCTION_RETURN
May 25 15:17:58 1999: getRenglon: (CS_SUCCEEDED) <<<
May 25 15:17:58 1999: getRenglon: (CS_SUCCEEDED) <<<
May 25 15:17:58 1999: desaloja_renglon: FUNCTION_ENTER >>>
May 25 15:17:58 1999: desaloja_renglon: <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:58 1999: obtieneCount: FUNCTION_RETURN (CS_SUCCEEDED) <<<
May 25 15:17:58 1999: handle_results: CS_CMD_DONE ---
May 25 15:17:58 1999: handle_results: FUNCTION_RETURN (CS_SUCCEEDED)

```

```

May 25 15:17:58 1999: termina_comando FUNCTION_ENTER (CS_SUCCEEDED)
May 25 15:17:58 1999: SQLCommand <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:58 1999: SQLCommand FUNCTION_ENTER >>>
[select count(*) from #tmp]
May 25 15:17:58 1999: envia_comando FUNCTION_ENTER >>>
May 25 15:17:58 1999: handle_results: FUNCTION_ENTER >>>
May 25 15:17:58 1999: handle_results: CS_ROW_RESULT --
May 25 15:17:58 1999: obtieneCount: FUNCTION_ENTER >>>
May 25 15:17:58 1999: bind_columns: FUNCTION_ENTER >>>
May 25 15:17:58 1999: bind_columns <<< FUNCTION_RETURN
May 25 15:17:58 1999: getRenglon: (CS_SUCCEEDED) <<<
May 25 15:17:58 1999: getRenglon: (CS_SUCCEEDED) <<<
May 25 15:17:58 1999: desaloja_renglon: FUNCTION_ENTER >>>
May 25 15:17:58 1999: desaloja_renglon: <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:58 1999: obtieneCount: FUNCTION_RETURN (CS_SUCCEEDED) <<<
May 25 15:17:58 1999: handle_results: CS_CMD_DONE --
May 25 15:17:58 1999: handle_results: FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:58 1999: termina_comando FUNCTION_ENTER (CS_SUCCEEDED)
May 25 15:17:58 1999: SQLCommand <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:58 1999: validacionCat: numRengHist 12 numRengTmp 12
May 25 15:17:58 1999: validacionCat: FUNCTION_RETURN <<<
May 25 15:17:58 1999: validadorHndl: antes validacionExpr 1
May 25 15:17:58 1999: validacionExpr: FUNCTION_ENTER >>>
May 25 15:17:58 1999: SQLCommand FUNCTION_ENTER >>>
[SELECT * FROM HIST_1_1 WHERE env_folio = 16 AND ins_sector = 1 AND ins_clave = 59]
May 25 15:17:58 1999: envia_comando FUNCTION_ENTER >>>
May 25 15:17:58 1999: handle_results: FUNCTION_ENTER >>>
May 25 15:17:58 1999: handle_results: CS_ROW_RESULT --
May 25 15:17:58 1999: validaRenglon: FUNCTION_ENTER >>>
1999: obtieneOperandos: FUNCTION_RETURN <<<
May 25 15:17:58 1999: SQLCommand FUNCTION_ENTER >>>
[SELECT count(cuenta),Sum(s_cuenta) FROM HIST_1_1 WHERE env_folio = 16 AND ins_sector = 1 AND ins_clave = 59]
May 25 15:17:58 1999: envia_comando FUNCTION_ENTER >>>
May 25 15:17:59 1999: desaloja_renglon: FUNCTION_ENTER >>>
May 25 15:17:59 1999: desaloja_renglon: <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:59 1999: procesaSPValidacion: stauts={0} FUNCTION_RETURN <<< (CS_SUCCEEDED)
May 25 15:17:59 1999: handle_results: CS_CMD_SUCCEEDED --
May 25 15:17:59 1999: handle_results: CS_CMD_DONE --
May 25 15:17:59 1999: handle_results: FUNCTION_RETURN (CS_SUCCEEDED)
'VO'
May 25 15:17:59 1999: SQLCommand FUNCTION_ENTER >>>
[exec ssa_setStatusEnvio 16, 59, 1, 'VO']
May 25 15:17:59 1999: envia_comando FUNCTION_ENTER >>>
May 25 15:17:59 1999: handle_results: FUNCTION_ENTER >>>
May 25 15:17:59 1999: handle_results: CS_STATUS_RESULT --
May 25 15:17:59 1999: noElaboraResultados: FUNCTION_ENTER >>>
May 25 15:17:59 1999: bind_columns: FUNCTION_ENTER >>>
May 25 15:17:59 1999: enviaSoloValidacion: DELETE dbssa_metadatos..HIST_1_1 WHERE env_folio = 16 AND
ins_clave = 59 AND ins_sector = 1
May 25 15:17:59 1999: SQLCommand FUNCTION_ENTER >>>
[DELETE dbssa_metadatos..HIST_1_1 WHERE env_folio = 16 AND ins_clave = 59 AND ins_sector = 1 ]
May 25 15:17:59 1999: envia_comando FUNCTION_ENTER >>>
May 25 15:17:59 1999: handle_results: FUNCTION_ENTER >>>
May 25 15:17:59 1999: handle_results: CS_CMD_SUCCEEDED --
May 25 15:17:59 1999: handle_results: CS_CMD_DONE --
May 25 15:17:59 1999: handle_results: FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:59 1999: termina_comando FUNCTION_ENTER (CS_SUCCEEDED)
May 25 15:17:59 1999: SQLCommand <<< FUNCTION_RETURN (CS_SUCCEEDED)
May 25 15:17:59 1999: WriteMessageQ: FUNCTION_ENTER >>>
May 25 15:17:59 1999: WriteMessageQ: +QUEUE Flags(32) [CONNECT_Q12]
May 25 15:17:59 1999: writeMessageQ: <<< FUNCTION_RETURN (CS_SUCCEEDED) OBJ ID[13][CONNECT_Q12]

```

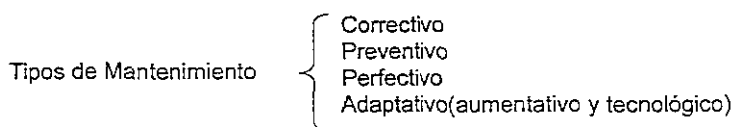
7.3 Fase de mantenimiento

El mantenimiento se clasifica en varios tipos, por lo que se revisarán a continuación.

Tipo Mantenimiento

La adaptación de un sistema existente a las nuevas necesidades es una posibilidad siempre abierta en todos los sistemas de nueva implantación. El mantenimiento ligado a estas adaptaciones obliga al analista a evaluar las nuevas necesidades y volver a las fases adecuadas del análisis, del diseño y la implantación de sistemas.

En esta sección examinaremos cuatro tipos de mantenimiento: Correctivo, Preventivo, Perfectivo y Adaptativo, en este último se encuentran el aumentativo y el tecnológico.



Correctivo

Esta primera actividad del mantenimiento se da ocasionalmente cuando en la etapa de prueba del software no se hayan descubierto todos los errores latentes de un sistema. Durante el uso del sistema se encontrarán errores, los cuales deben ser informados al equipo de desarrollo. El proceso que incluye el diagnóstico y corrección de uno o más errores se denomina Mantenimiento Correctivo.

Preventivo

Este tipo de mantenimiento es para prevenir errores. En este tipo de mantenimiento se puede considerar el mantenimiento a la información que se maneja al actualizarla, para que los resultados dados por el sistema sean correctos.

Perfectivo

Esta actividad de mantenimiento se da cuando un paquete de software tiene éxito. A medida que se usa el software, se reciben de los usuarios recomendaciones sobre nuevas posibilidades acerca de modificaciones a funciones ya existentes. Para satisfacer estas peticiones se lleva a cabo el mantenimiento perfectivo.

El mantenimiento perfectivo comprende también los cambios solicitados al programador del sistema

Adaptativo (Aumentativo y Tecnológico)

La vida útil estimada del software de aplicación puede fácilmente sobrepasar los diez años, pero considerando la evolución del ambiente, en la práctica éste puede volverse obsoleto. Por lo tanto, el mantenimiento adaptativo es una actividad que modifica al software para que las interacciones se ejecuten adecuadamente con su entorno cambiante.

El mantenimiento adaptativo se debe a cambios en el ambiente del programa y a la adaptación de nuevas unidades o módulos.

Un estudio hecho por *Lientz* y *Swanson* (1980) descubrió que alrededor del 65% del mantenimiento era perfecto, el 18% adaptativo, y el 17% correctivo.

Aumentativo

Este tipo de mantenimiento se da cuando se incluyen nuevas funciones que no se contemplaron al inicio del desarrollo del sistema y surgen como una necesidad del usuario.

Tecnológico

Esta actividad que contribuye al mantenimiento se da debido a todo cambio importante en la informática. Si en un ciclo de 36 meses surgen nuevas generaciones de Hardware, regularmente aparecen nuevos sistemas operativos o nuevas versiones de los antiguos; y frecuentemente se mejoran o modifican los equipos periféricos y otros elementos de sistemas.

DIAGRAMA DE MEJORAS DEL SISTEMA

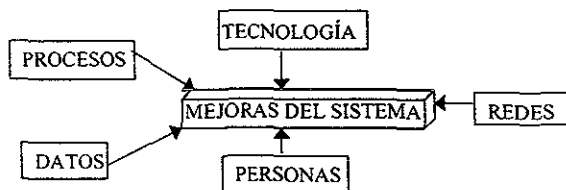


Figura 7.11 Elementos que intervienen en la mejora del sistema

Así terminamos el desarrollo de esta aplicación y a continuación en la última sección de este trabajo presentaremos nuestras conclusiones, el glosario de términos relativos al mismo, la bibliografía que utilizamos para llevar a cabo el tema y finalmente los apéndices que hemos considerado necesarios para complementar el tema.

Conclusiones

Al finalizar el presente trabajo podemos concluir que el *Data Warehouse* es una colección de información corporativa derivada directamente de los sistemas operacionales y de algunos datos de origen externo, a los que se les ha aplicado procesos de análisis, selección y transferencia, además no interfiere con los sistemas operacionales y cuenta con la capacidad para soportar la constante necesidad de consultas estructuradas, reportes analíticos y entrega de información que se requiere para uno de los activos más importantes para toda organización, que es la toma de decisiones.

Un *Data Warehouse* no es ni un producto de software ni un motor o tecnología de base de datos en particular, sino una serie de componentes y procesos que en conjunto forman la "Arquitectura *Data Warehouse*".

Existe una diferencia entre los componentes de la arquitectura que realizan las tareas y las herramientas que pueden apoyar las tareas.

Es conveniente que no se case el *Data Warehouse* con una marca o producto específico, lo mejor es considerar un ambiente que permite el uso de una variedad de productos y soluciones para aprovechar la experiencia de diversos especialistas y no de un único proveedor, es decir que el proveedor sea un buen integrador de diversas herramientas.

La *metadata* es uno de los componentes más importantes y necesarios de un *Data Warehouse*, una clara evidencia de esto es el mapeo que se realiza de las bases de datos operacionales hacia el *Data Warehouse*. Este mapeo o transformación de datos es uno de nuestros principales objetivos, y el diseño realizado de la *metadata* lo satisface ya que es aquí donde se define la extracción de datos de las bases de datos operacionales, la transformación que sufren estos datos y las reglas de integración que se utilizaron para el depósito de los datos en un repositorio o bien en los *datamarts*.

Otro de los componentes esenciales en la arquitectura del *Data Warehouse* es el *middleware*, su función principal es la de extraer, validar e integrar la información, esta funcionalidad se implementó por medio de una aplicación *Open Server/Open Client*.

Se utilizaron una serie de estructuras de datos, procedimientos remotos y *stored procedures* en el *middleware* para tener una mejor *performance* del sistema.

Se puede decir que nuestro *middleware* es un administrador de tareas, ya que puede estar recibiendo información para validar y a la vez estar validando, integrando o enviando información.

Por lo tanto, concluimos que la metadata es la parte vital del middleware, ya que es en ella donde se basa para la ejecución de los procesos de transformación e integración de la información de una manera escalable y configurable.

El *middleware* desarrollado no sólo cuenta con la capacidad de soportar un *Data Warehouse* para un sistema bancario, sino que es capaz de soportar las transacciones de un *DW* para el sistema financiero de un país.

Además, hemos encontrado que entre otras cosas, el diseño de esta aplicación ha resultado ser realmente configurable y adaptable, de tal modo que las modificaciones que se tienen que hacer a la aplicación en el futuro son mínimas, puesto que en caso de que cambien las reglas del negocio sólo se tienen que cambiar las definiciones y esto no afecta el código de la aplicación.

Para nosotros es muy importante mencionar que la formación proporcionada por la Facultad de Ingeniería nos permitió llevar a cabo la realización de este importante trabajo en nuestra vida profesional.

Bibliografía

Libros

Harjinder S. Hill y Prakash C. Rao, *Data Warehousing*, Prentice Hall, México, 1996.

Michael C. Rengruber and William W. Gregory, *The Data Modeling Handbook*, John Wiley and Sons, Inc., USA, 1994.

Michael J. Corey and Michael Abbey, *ORACLE Data Warehousing*, Osborne/McGraw Hill, España, 1997.

Roger S. Pressman, *Ingeniería de Software, un Enfoque Práctico*, Mc. Graw Hill, México, 1988.

Richard Fairley, *Ingeniería de Software*, Mc. Graw Hill, México, 1989.

Tom Hammergren, *Datawarehousing, Building the Corporate Knowledgebase*, International Thompson, USA, 1996.

Vidette Poe with Contributions of Lauro L. Reeves, *Building a Data Warehouse for Decision Support*, Prentice Hall, USA, 1997.

W.H Inmon, *Building The Data Warehouse*, John Wiley & Sons, Inc., USA, 1996.

Revistas

Adolfo Guzmán Arenas, *Uso y Diseño de Mineros de Datos*, Soluciones Avanzadas, Xview, S.A. de C.V., México, 1996.

Bernardo Miramón Commons, *Datawarehousing, Estrategias Generales de Implantación*, Soluciones Avanzadas, Xview, S.A. de C.V., México, 1996.

Bruce Jenks, *Datawarehouse de Niveles*, Soluciones Avanzadas, Xview, S.A. de C.V., México, 1996.

Ivan Pech Escalante, *Datawarehouse-Soluciones Avanzadas*, Xview, S.A. de C.V., México,- 1996.

José Antonio Sánchez Paredes, *Una Receta para Construir un Repositorio Analítico de Información (Datawarehouse)*, Soluciones Avanzadas, Xview, S.A. de C.V., México, 1996

Kamran Parsaye, *OLAP and Data Mining: Bridging the Gap-Database Programming and Design*, Miller Freeman Publications, USA, 1997.

Karl Mc. Demott, *Muchos Datos, Poca Información Útil Para un Análisis y Toma de Decisiones*, Soluciones Avanzadas, Xview, S.A. de C.V., México, 1996.

Ralph Kimball, *Datawarehouse Architect-DBMS*, Miller Freeman Publications, USA, 1997.

Richard P. Sherman, *Metadata: The Missing Link*, DBMS-Miller Freeman Publications, USA, 1997.

Ulises Castillo, *Middleware: No salga a hacer Datawarehousing sin él*, Soluciones Avanzadas, Xview, S.A. de C.V., México, 1996.

Artículos

Cristobal Juárez C., *Bases de Datos Relacionales, Bases de Datos Multidimensionales y Data Warehouse*, México, 1996.

E. F. Codd, S.B. Codd and C.T. Salley, *Providing OLAP (On Line Analytical Processing) to User-Analysts: An IT Mandate*, E.F. Codd and Associates, USA, 1998.

Héctor Díaz de Salas, *Data Warehouse, Seminario Estratégico*, México, 1996.

Iván Pech, *Data Warehouse Arquitectura*, México, 1996.

Varios, *Primer Seminario Estratégico de Data Warehouse*, Information News and Knowledge, México, 1996.

Apéndice Ingeniería de Software

A

Ingeniería de Software

La Ingeniería de Software consiste en el establecimiento y uso de principios y metodologías de ingeniería robustos, orientados a obtener software de calidad que sea funcional y que se adapte a las necesidades actuales y futuras de la empresa.

La Ingeniería de Software abarca un conjunto de tres elementos claves: métodos, herramientas y procedimientos, que facilitan el controlar el proceso de desarrollo de software y suministran las bases para construir software de alta calidad de una forma productiva.

Los métodos de la Ingeniería de Software suministran el "cómo" construir técnicamente el software, abarcan tareas como:

- Planificación y estimación de proyectos.
- Análisis de los requerimientos del sistema y del software.
- Diseño de los datos.
- Arquitectura de programas y procedimientos.
- Codificación.
- Pruebas.
- Mantenimiento.

Las herramientas de la Ingeniería de Software suministran un soporte automático o semiautomático para los métodos (por ejemplo herramientas CASE).

Los procedimientos de la Ingeniería de Software definen la secuencia en la que se aplican los métodos, la secuencia en la generación de documentos, los controles de calidad, y las guías que facilitan el establecer el desarrollo del software.

El ciclo de vida

El ciclo de vida en la Ingeniería de Software (figura A1), exige un enfoque sistemático, semi-secuencial del desarrollo del software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, pruebas y mantenimiento.

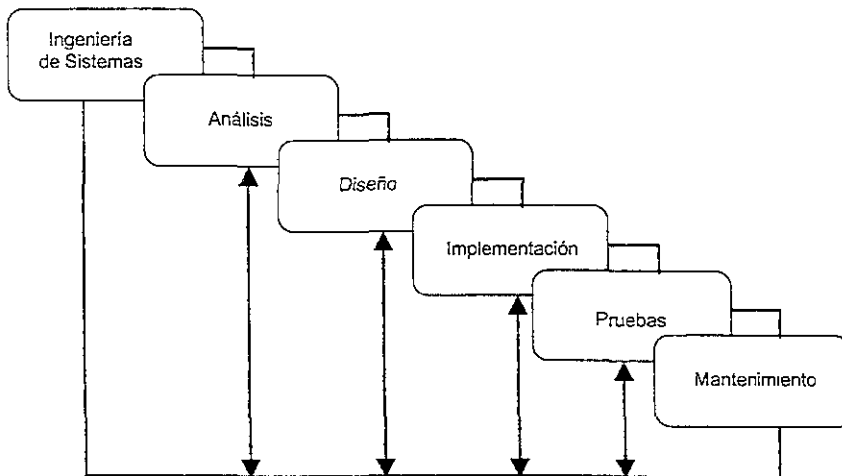


Figura A1. El ciclo de vida clásico.

Las etapas de la Ingeniería de Software se aplican tanto a procesos, como a datos.

En el caso de procesos, el ciclo de vida nos conduce a la creación de programas que automatizan los procesos. Para poder automatizar procesos, antes se deben analizar y diseñar los datos.

La aplicación de las etapas de la Ingeniería de Software a datos conduce a la creación de un Modelo de Datos, que puede ser, un esquema de archivos planos, o bien una Base de Datos. Este modelo será utilizado en el proceso de automatización de procesos.

No todos los sistemas basados en computadora hacen uso de una Base de Datos, para aquellos que lo hacen, la Base de Datos es la guía para todas las funciones del sistema.

El análisis, diseño e implementación de una Base de Datos forman parte del ciclo de vida de la Ingeniería de Software y son actividades que no dependen del desarrollo de alguna

aplicación en particular y pueden iniciarse una vez que se ha definido el alcance del sistema de la información.

El ciclo de vida en la automatización de procesos (desarrollo de software)

Ingeniería y análisis del sistema

Debido a que el software es siempre un componente de un sistema mayor, el trabajo comienza estableciendo los requerimientos de todos los elementos del sistema y luego asignando algún subconjunto de estos requerimientos al software.

En esta etapa se fijan las limitantes del software, los alcances, sus funciones, sus interfaces (conexión con otros elementos del sistema global), los datos utilizados y los generados, etc.

Análisis

Para comprender la naturaleza de los programas a construir, el Ingeniero de Software debe comprender el dominio de los procesos a automatizar, las funciones, rendimientos e interfaces requeridas.

En esta etapa se analizan detalladamente cada uno de los procesos involucrados, también se documentan. Este proceso se realiza en estrecha comunicación con el cliente.

La documentación generada en esta etapa, refleja, en forma detallada, los requerimientos del sistema.

Diseño

El diseño de software es un proceso que se enfoca a la generación de la arquitectura del software y detalle procedimental.

El proceso de diseño traduce los requerimientos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación.

Implementación

En el caso de procesos, esta etapa se conoce comúnmente como codificación. En esta etapa, el diseño debe traducirse en una forma legible para la máquina.

Si el diseño se ejecuta de una manera detallada, la codificación puede realizarse mecánicamente.

Pruebas

Las pruebas se enfocan sobre la lógica interna del software, asegurando que todas las sentencias sean probadas, y sobre las funciones externas, realizando pruebas para asegurar que la entrada definida producirá los resultados que se requieren.

Mantenimiento

El software sufrirá cambios después de que sea liberado. Los cambios pueden ser debido a nuevas adaptaciones (mantenimiento adaptativo) o a que se han encontrado errores (mantenimiento correctivo).

El ciclo de vida en la generación de un modelo de datos (desarrollo de bases de datos)

Ingeniería y análisis del sistema

En esta etapa se determina la información que maneja el sistema, las características de esta información. Se determina también el alcance del modelo que se desea generar y sus limitantes (la información que se deberá considerar).

En esta etapa se deben reflejar las reglas del negocio como requerimientos del modelo de datos. *Hay que puntualizar que cada empresa tiene reglas y requerimientos diferentes.*

Por ejemplo, si se desea desarrollar un modelo de datos para un banco, se debe determinar la información del cliente que se utiliza, los tipos de cuentas que maneja el banco, restricciones en cuanto a movimientos, reglas de integridad, etc.

Los requerimientos necesarios se obtienen en base a:

- Entrevistas con el cliente.
- Cuestionarios.
- Lectura de manuales de procedimientos, reglamentos, reportes, etc.
- Prototipos.

Los requerimientos para el modelo de datos deben reflejar el sistema de información de la empresa, así como también futuros cambios.

El documento de requerimientos generado es la base y guía del análisis y diseño de la Base de Datos.

Análisis

Esta etapa es una etapa de retroalimentación con el cliente, en la que seguramente se proponen y cambian requerimientos.

En esta etapa se analizan los requerimientos y se obtiene una descripción del sistema de información a modelar, se genera el Modelo Conceptual de Datos.

El Modelo Conceptual de Datos refleja los datos, alcances, restricciones y reglas del sistema de información. Este modelo es independiente del Modelo de Datos a utilizar (jerárquico, red, relacional, etc.).

Diseño

El diseño de Bases de Datos incluye el Diseño Lógico y Físico de la Base de Datos.

En el Diseño Lógico se hace un mapeo del Modelo Conceptual de Datos al Modelo Lógico de Bases de Datos a utilizar.

En el Diseño Físico se determina la forma en como se implementará la Base de Datos, en base al DBMS utilizado (índices, dispositivos, esquemas de protecciones, etc.).

Implementación

En esta etapa se genera la Base de Datos de acuerdo al Diseño Físico.

Mantenimiento y Administración

En esta etapa se generan cambios a la base de datos y se llevan a cabo tareas de administración que permitan recuperar la base de datos en caso de que sucedan fallas.

Apéndice Conceptos Básicos de Bases de Datos

B

Modelo Conceptual de Datos

El Modelo Conceptual de Datos o Modelo Lógico de Datos es el esquema resultado en la etapa de análisis, que representa los requerimientos de datos del Sistema de Información que se está modelando.

El Modelo Conceptual cumple con las características estándares de ser:

Claro y preciso. Esto es, debe de describir completamente los datos, la semántica de los mismos, sus relaciones, la seguridad, la integridad y los alcances del Sistema de Información que representa.

Constante. Es decir, debe contemplar requerimientos presentes y futuros para evitar cambios constantes en el modelo.

Flexible. Debe de poderse adaptar a nuevos requerimientos sin tener que sufrir cambios significativos. Un buen modelo conceptual no debe permitir que cambios en el Modelo Conceptual afecten al Modelo de Vistas. El esquema de vistas debe ser derivable del Modelo Conceptual.

Integración de aplicaciones. El Modelo Conceptual debe de servir como base para el desarrollo de las múltiples aplicaciones que se den en el Sistema de Información, no solamente de aquellas que se proyecten en las etapas de planeación para el Sistema de Información en Bases de Datos, sino también para aquellas que se presenten durante toda la vida útil de la Base de Datos. Independiente del software o hardware utilizado para su implementación.

El modelo Entidad-Relación proporciona una ayuda gráfica conocida como Diagrama Entidad-Relación (DER), que permite la representación de la estructura lógica del modelo de datos.

El análisis para la generación del Modelo Conceptual de Datos debe tomar como entrada los Requerimientos del Sistema de Información de la empresa.

El procedimiento realizado para la generación del Modelo Conceptual de Datos es el siguiente:

- Seleccionar las Entidades y el tipo de entidad.
- Seleccionar las Relaciones entre las entidades que están dentro del alcance de integración de la institución.
- Determinar el tipo de relación entre las entidades.
- Determinar el grado de asociación o cardinalidad entre las entidades.
- Asignar atributos a las entidades y relaciones.

Entidades. Es un objeto que existe y que es distinguible de otros objetos. La institución le reconoce en forma independiente y puede ser definido en forma única, además de contar con atributos que la describen.

Existen varias convenciones para la representación de DER's. Para representar las entidades en este trabajo se utilizaron rectángulos identificados con el nombre de la entidad, además de la lista de atributos.

Las entidades se obtienen del análisis de los requerimientos de la empresa, cada una de las entidades tienen múltiples instancias las cuales deben poder ser identificadas unas de otras por medio de sus atributos. Si las instancias no pueden ser identificadas unas de otras, entonces no se trata de una entidad. Una entidad es descrita por más de un atributo.

Atributo. Es una propiedad de una entidad. Los atributos describen, identifican, clasifican o determinan el estado de una entidad. Estos deben ser asignados solamente a una entidad.

Por definición, toda ocurrencia de una entidad debe ser identificable en forma única. Es por ello, que se debe encontrar un identificador para las instancias de la entidad, es decir una llave.

Super llave. Es un conjunto de uno o más atributos que, tomados en conjunto, permiten identificar en forma única una instancia dentro del conjunto de entidades.

Llave candidato. El concepto de Super llave no es suficiente, puesto que puede contener atributos extraños. Las Super llaves con el menor conjunto de atributos se conocen como llaves candidato y es posible que existan diferentes conjuntos de atributos que pueden servir como llaves candidato.

Llave primaria. Una llave primaria es una llave candidato que ha sido seleccionada por el diseñador de la base de datos como el medio de identificar las instancias de una entidad.

Las características necesarias para una llave primaria son:

- Única.
- Conocible en cualquier tiempo.

Las características deseables de una llave primaria son las siguientes:

- Estable.
- No descriptiva.
- Pequeña y simple.

Cuando la llave primaria no cumple con las características mencionadas, se puede crear una llave primaria no natural, es decir, una que no haya sido obtenida de los atributos mencionados en los requerimientos de la institución. Este tipo de llaves primarias no tienen significado alguno en el contexto de la empresa.

Relaciones. Es el vínculo o conexión que existe entre dos o más entidades. Las asociaciones o relaciones se obtiene del análisis de los requerimientos de la institución. Generalmente las relaciones son verbos, pero no todos los verbos representan las relaciones.

Se deberán ignorar los verbos que:

- Denoten actividades que no tengan importancia para la institución.
- Aquellos que denoten procesos.

Cardinalidad de la relación. El grado de asociación o cardinalidad, representa en forma cuantitativa, el número de instancias de una entidad con las que puede estar asociada una instancia de otra entidad.

Una asociación puede tener tres tipos de cardinalidad:

TIPO	
1:1	(Uno a uno)
1:N	(Uno a muchos)
M:N	(Muchos a muchos)

La cardinalidad de las relaciones refleja las reglas de la empresa presentados en los requerimientos de la misma. Las reglas de empresa son definiciones que se obtienen del análisis de los requerimientos.

Cardinalidad Máxima. La cardinalidad máxima representa el número máximo de instancias con las que esta relacionada una instancia de una entidad. Para determinar la cardinalidad máxima, primero se deberá leer la relación en una dirección y luego en la otra.

Cardinalidad Mínima. La cardinalidad mínima representa el número mínimo de instancias con las que esta relacionada una instancia de una entidad.

Las reglas de empresa que excluyen relaciones 1:0 se traducen como la obligatoriedad de que una entidad participe en una relación con otra entidad. Cuando no se excluyen las relaciones 1:0, se trata de relaciones opcionales, sin obligatoriedad. Una entidad con obligatoriedad se puede considerar como una entidad débil puesto que para existir depende de la existencia de otra entidad.

El procedimiento utilizado para determinar la cardinalidad mínima de una relación es semejante al utilizado para la cardinalidad máxima, solamente que se plantea la obligatoriedad de la relación.

La cardinalidad de un atributo indica el número de valores que existen para un atributo en una instancia de una entidad. La cardinalidad mínima puede ser opcional, lo que indica que una instancia puede no tener valor para el atributo, u obligatoria. Los atributos que conforman la llave primaria tienen cardinalidad mínima obligatoria.

Generalmente la cardinalidad mínima no se representa en el DER, solamente se documenta en el diccionario de datos.

La cardinalidad máxima puede ser singular, lo que indica que una instancia debe tener un solo valor para el atributo, o plural.

Contenido del Diccionario

Para cada entidad:

- Nombre.
- Descripción.
- Llave primaria.
- Número aproximado de instancias.

Para cada relación:

- Nombre (entidad-relación-entidad)
- Descripción
- Cardinalidad máxima
- Cardinalidad mínima

Para cada atributo:

- Nombre.
- Tipo: tipo de dato asociado al atributo.
- Dominio: el conjunto de valores válidos para el atributo. El dominio está determinado, en primera instancia por el tipo de dato, y en segunda por las reglas de empresa que apliquen al atributo.
- Cardinalidad. máxima y mínima.
- Descripción.

Integridad. El concepto de integridad se refiere a que los datos deben de ser válidos. Se definen los siguientes tipos de integridad:

- Integridad de Entidades. Cada tupla en una relación debe ser única.
- Integridad de Dominio. Cada valor de un atributo pertenece a un dominio previamente definido. Si la llave primaria es única, se garantiza que una tupla es única en una relación.
- Integridad Referencial. Cada llave foránea debe estar asociada a una llave primaria válida, o debe tener un valor nulo.

- **Integridad del Negocio.** Los datos de la base de datos deben cumplir con las reglas del negocio.

Para la llave foránea:

Insert/Update

No permitir el insert/update si no existe una llave primaria asociada.

Asignar valor nulo a la llave foránea sino existe la correspondiente llave primaria.

Asignar valor default a la llave foránea sino existe la correspondiente llave primaria.

Si la llave primaria asociada a la llave foránea no existe, dar de alta la tupla asociada a la llave primaria.

Para la llave primaria

Para Update

Modificar todas las llaves foráneas asociadas.

Evitar la modificación si existen llaves foráneas asociadas.

Para Delete

Borrar todas las llaves foráneas asociadas.

Evitar el borrado si existen llaves foráneas asociadas.

Asignar NULL a las llaves foráneas asociadas

Asignar un valor de default a las llaves foráneas asociadas.

Reglas de Codd

En 1985 Codd publicó una serie de reglas o principios que un DBMS debe cumplir para considerarse "*Completamente relacional*".

Regla Cero

Un DBMS debe manejar los datos almacenados utilizando únicamente sus características relacionales.

Regla 1: Representación de la información

Toda la información, en el nivel lógico, debe ser representada como valores en tablas.

Regla 2: Garantía de acceso

Debe ser posible acceder cualquier dato en la base de datos si se hace referencia al nombre de la base de datos, la columna y el valor de la llave primaria.

Regla 3 Representación de valores nulos

Se deben poder representar valores nulos, sin importar el tipo de dato. El valor nulo debe ser distinto de cero o cualquier valor numérico, así como de la cadena vacía.

Regla 4 Catálogo Relacional

El catálogo de la base de datos, que contiene la descripción lógica de la misma, debe ser representado como datos ordinarios.

Regla 5 Lenguaje comprensible

Sin importar el número de lenguajes soportados, se debe proporcionar un lenguaje que, por medio de sentencias expresadas como cadenas de caracteres, permita la definición de datos y vistas, la manipulación de datos, la definición de las reglas de integridad y los esquemas de autorización.

Regla 6 Modificación de vistas

Una vista que sea teóricamente modificable, puede ser modificada.

Regla 7 Operación Insert, Delete y Update

Una relación que puede ser utilizada para consulta, también lo puede ser para las operaciones insert, update y delete.

Regla 8. Independencia física

Las aplicaciones de la base de datos son inmunes a cambios relacionados con los métodos de acceso a los datos o con los esquemas de almacenamiento de la base de datos.

Regla 9 Independencia lógica

Cambios generados a nivel lógico que no afecten la información a nivel lógico, no requieren de modificaciones en las aplicaciones.

Regla 10 Regla de integridad

Las reglas de integridad de entidades e integridad referencial deben ser indicadas en el lenguaje que proporciona el DBMMS y almacenadas como datos en el catálogo, no es necesario programarlas en las aplicaciones.

Regla 11 Independencia de la distribución de datos

Los programas de aplicación y los comandos del usuario que utilicen el lenguaje de definición de datos no deben sufrir cambios por efectos de distribución de la base de datos.

Regla 12 No subversión

El DBMS debe tener el control total de los datos, no es posible que por medio de lenguajes de bajo nivel se pueden violar las reglas de integridad impuestas en la base de datos.

Normalización

Normalización es una técnica desarrollada para asegurar que las estructuras de datos sean eficientes. Los beneficios de la Normalización son:

- Minimiza la redundancia.
- Libera de dependencias indeseables de inserción, borrado y actualización.
- Minimiza la reestructuración de datos cuando se introduce algo nuevo. Se mejora la independencia de datos, permitiendo que las extensiones a la base tengan poco o ningún efecto sobre los programas o aplicaciones que tiene acceso a ella.
- No se introducen restricciones artificiales a las estructuras de datos

Aunque se han definido más estados de normalización, sólo se han aceptado ampliamente tres. Estos se conocen como **primera, segunda y tercera forma normal**.

El normalizar es un proceso ascendente, en el que se parte de un universo de relaciones y atributos, y se avanza de forma en forma hasta llegar a la tercera forma normal.



Las etapas de normalización se muestran adelante con un ejemplo, dada la relación no normalizada:

ORDEN(#orden, fecha, #proveedor, nombre_proveedor, direccion_proveedor, #producto, #producto, descripcion_producto, cantidad_producto, precio_total_producto, precio_total_orden)

Primera Forma Normal

Un registro en primera forma normal no incluye grupos repetidos. Es decir cada uno de sus campos debe tener un sólo valor.

En la relación ORDEN se observa que para una misma orden habrá varios productos, por lo que #producto y otros atributos serán grupos repetidos. En la primera forma normal habría que separar:

ORDEN(#orden, fecha, #proveedor, nombre_proveedor, dirección_proveedor, precio_total_orden)

PRODUCTO_ORDENADO(#orden, #producto, descripcion_producto, precio_producto, cantidad_producto, precio_total_producto)

Segunda Forma Normal

Cada atributo depende de la totalidad de la llave, y no sólo de una parte de ella.

Se puede observar en PRODUCTO_ORDENADO que descripcion_producto depende sólo de #producto, y no tiene que ver con #orden. En la segunda forma normal quedaría:

ORDEN (#orden, fecha, #proveedor, nombre_proveedor, direccion_proveedor, precio_total_orden)

PRODUCTO (#producto, descripcion_producto, precio_producto)

PRODUCTO_ORDENADO (#orden, #producto, cantidad_producto, precio_total_producto)

Tercera Forma Normal

Todos los atributos dependen solamente de la llave y no de otros atributos no llave.

En la relación ORDEN se presenta este problema.

nombre_proveedor y direccion_proveedor depende de #proveedor

De modo que las tablas en la tercera forma normal quedarian como:

ORDEN(#orden, fecha, #proveedor, precio_total_orden)

PROVEEDOR(#proveedor, nombre_proveedor, direccion_proveedor)

PRODUCTO(#producto, descripcion_producto, precio_producto)

PRODUCTO_ORDENADO(#orden, #producto, cantidad_producto, precio_total_producto)

Redundancia

La redundancia de la información no implica necesariamente redundancia. La d. La redundancia se puede definir como la replicación de hechos.

Un Diseño de Bases de Datos con tablas normalizadas minimiza la redundancia de datos, pero, ¿que sucede con el rendimiento?

Pueden existir consultas críticas que involucren la obtención de datos varias tablas, en este caso el costo de los joins puede ser muy alto. Existen algunas aplicaciones que requieren el cálculo de datos en base a ciertas columnas de una o varias tablas. El cálculo de estos datos implica tiempo. Por ejemplo, El cálculo de salario mensual de un empleado en base a los campos prestaciones, salario_gravable, bono, incentivo.

En muchas ocasiones solamente una parte de los datos de una tabla se utiliza constantemente.

Un buen diseño debe:

- Minimizar la redundancia de información.
- Fácil de comprender.

Las tablas que componen el diseño no deben de estar demasiado fragmentadas, ni tampoco deben contener demasiada información, ya que sería difícil determinar la entidad que representan.

Es necesario que las tablas tengan una estructura que permitan fácilmente determinar la información que representan.

- Ayudar a que el performance de las consultas y movimientos en la BD sea aceptable.

La información no utilizada debe separarse.

La información que se accesa junta debe permanecer como tal.

En un ambiente de operación real pueden existir aplicaciones de cualquier tipo sobre la Base de Datos, por lo que hay que determinar cuales son las consultas o movimientos más críticos e importantes. Para estos hay que determinar:

- La frecuencia con que se llevan a cabo.
- El número de usuarios concurrentes que las ejecutan.
- El volumen de información procesada.
- Procesos en ejecución cuando estos se realizan.
- El tipo de operación (en línea o batch).
- El usuario que las ejecuta.
- El tiempo de respuesta requerido.

Ventajas y Desventajas de la Normalización

- Reduce la redundancia de datos.
- Las tablas son más pequeñas, lo que implica que la Base de Datos ocupa menos espacio de disco.
- Como las tablas son más pequeñas los accesos a disco para leer una tabla se reducen.
- Las modificaciones son más eficientes.
- Pueden reducir la contención de datos.

Desventajas

- Debido a que el número de tablas se incrementa, para las consultas que requieren de mucha información, muy probablemente se tengan que llevar a cabo joins entre varias tablas.

- Una consulta que involucra joins tiene, generalmente, un tiempo de respuesta mayor a una consulta sobre una sola tabla.

Desnormalización

- La desnormalización tiene como objetivo obtener un mejor performance de cierto tipo de aplicaciones sobre la Base de Datos.
- Cuando se desnormaliza se debe partir de que existe un diseño normalizado.
- La desnormalización se da a nivel de columna o tabla.
- La desnormalización mejora el performance de cierto tipo de aplicaciones, por lo que hay que tener conocimiento de como se van ha utilizar los datos.
- La desnormalización reduce el número de joins, el uso de llaves foráneas y se puede reducir el número de tablas.
- El costo de desnormalizar se refleja al momento de hacer actualizaciones y mantener la integridad de la Base de Datos.

Tácticas de Desnormalización

Redefinición de columnas

Las columnas que se van a duplicar deben de cumplir con:

- Ser del mismo tipo.
- Tener el mismo dominio.
- Tener el mismo nombre.

Las columnas a duplicar no deben ser muy volátiles, es decir, no deben cambiar constantemente, ya que de lo contrario se tendría que requerir un control más estricto para mantener la integridad de la Base de Datos.

La duplicación puede darse mediante una copia exacta o como una columna calculada con base en otras.

Duplicación de columnas, columnas derivadas

Una columna derivada es aquella cuyos valores son obtenidos de un cálculo con base en los registros o columnas de una o varias tablas:

- Suma de varias columnas.
- Conteo de registros.

Redefinición de columnas

La redefinición de columnas se da sobre llaves primarias cuando estas son muy grandes.

Se redefine la llave primaria para que esta sea más pequeña. Lo que provoca que:

- Las llaves foráneas en otras tablas sean más pequeñas.
- Los joins sean más rápidos, ya que se hacen con campos de relación más pequeños.

Redefinición de tablas

El performance de las aplicaciones de la Base de Datos se puede ver afectado por alguna de las siguientes razones:

- Los renglones contienen más atributos de los que se necesitan, de modo que el tamaño del registro es muy grande y el tiempo de lectura de n registros se ve afectado.
- La tabla contiene renglones que son muy poco utilizados, lo que provoca que la lectura de datos sea más lenta.

Existen dos métodos para redefinir tablas:

- **Duplicación.**
Ventajas
 - Se reduce la contención de datos.
 - En el caso de que se requiera hacer una consulta que involucre a todas las columnas, solamente se definirá sobre una tabla. No se requiere joins.

Desventajas

- Requiere más espacio en disco

La duplicación de tablas no representa un esquema de desnormalización.

Para mantener el modelo conceptual que percibe el usuario, se pueden definir vistas sobre las tablas resultantes de la duplicación.

- **Segmentación.**
Ventajas
 - Reduce la contención de datos.
 - Requiere menos espacio en disco.
 - Mantiene la integridad no requiere de esfuerzo adicional

Desventajas

- Una consulta que involucre la información de todas las columnas requiere de un join.

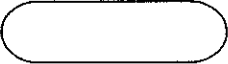

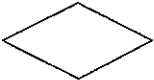



Para elegir entre un método y otro se deben considerar los siguientes aspectos.

- Integridad de la Base de Datos.
- Redundancia.
- Espacio en disco adicional.



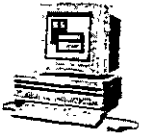




Apéndice C

Simbología

Simbología especial utilizada

Símbolo	Definición
Simbología para Diagramas de Flujo de Datos	
	Inicio o terminación de programa
	Proceso predefinido
	Toma de decisión
	Entrada manual de datos
	Dirección de flujo del proceso
	Conector para continuar el flujo del proceso

**Simbología para Esquemización de Arquitecturas
De Servidores Aplicativos**

	Sybase SQL Server
	Aplicación Open Server-Open Client
	Aplicación PowerBuilder
	Aplicación en lenguaje C, desarrollada exclusivamente con funciones estándar y del sistema operativo.
	Thread
	Base de Datos
	Cola de mensajes(message queue)

Apéndice Glosario

D

4GL. Lenguaje de programación de 4ta generación.

Ad hoc. Consulta que se realiza sin ser planeada previamente, o que no ha seguido ninguna preparación. También se aplica a reportes que se hacen sobre la marcha de acuerdo a alguna necesidad temporal.

AIX. Versión de UNIX propietaria de IBM.

Algoritmos de Encriptación. Algoritmos que permiten codificar la información.

API. Siglas de Application Program Interface. Programa de aplicación en modo gráfico.

Archivo DELTA. Archivo que se utiliza para comparar en un momento dado el estado de la base de datos.

Archivo LOG. Archivo donde se registran los cambios sufridos en una aplicación.

Arquitectura Cliente/Servidor. La arquitectura Cliente/Servidor es un nuevo esquema para trabajar y generar las aplicaciones de las grandes empresas, en él se distribuye gran parte de la carga del tiempo de proceso de las aplicaciones en los clientes, y se permite al servidor hacer operaciones más importantes que mostrar una ventana o recorrer las opciones de un menú, además, incorpora características de seguridad en el manejo de la información y también ofrece un control central general.

AS/400. Minicomputadora de IBM.

Atributos. Es una propiedad de una Entidad. Los atributos describen, identifican, clasifican o determinan el estado de una entidad.

Backup. Respaldo de la información, generalmente en medios magnéticos y de menor velocidad de acceso.

BSDLC. Ciclo de vida del Diseño de Sistemas hacia Atrás.

Bitácora. Es un documento en donde se guarda registro de cada transacción de cierto tipo, el tipo de transacción que registre define el tipo de bitácora de que se trata.

CASE. Siglas de *Computer Aided Software Engineering* (Ingeniería de *software* apoyada con computadora). Una metodología de desarrollo de *software* soportada por herramientas basadas en computación para el apoyo a tareas de análisis, diseño y desarrollo.

Ciclo de vida de los sistemas. El Ciclo de vida de los sistemas es una de las metodologías de desarrollo de *software*, ésta tiene su origen en la forma en que evoluciona la mayoría de los proyectos de *software* de sistemas.

Client Library. Librerías de apoyo de Sybase para el *Open Client*.

CORBA. Programa de aplicación similar a las APIs. Se refiere a una arquitectura de objetos distribuidos.

Consulta. Una petición formal y claramente especificada de información del DW o del mercado de datos planteada por un usuario o una herramienta operada por un usuario.

Cyrano. Cyrano es una herramienta muy amplia para realizar monitoreo de transacciones sobre bases de datos creada por Sybase.

Datamart. Implementación de Data Warehouse con un ámbito de datos y funciones de data warehouse más pequeño y restringido, que sirve a un departamento único o una parte de la organización. Una organización generalmente tiene varios datamarts.

DB2. Manejador de bases de datos de IBM. Es un producto ancla de la estrategia de Data Warehouse de IBM.

Dbase. Manejador de Bases de Datos utilizado generalmente en pequeñas aplicaciones en computadoras personales o de escritorio.

DBMS. Manejador de Bases de Datos.

DDE. Aplicación con interfaz gráfica similar a OLE y CORBA

Desnormalización.

Diccionario de Datos. Un compendio de definiciones y especificaciones para categorías de datos y sus relaciones.

Dimensión. Los DW organizan un gran depósito de datos históricos y operacionales usando múltiples dimensiones para establecer categorías. Una muy importante es el tiempo.

DMW. Siglas de *Data Base Mining Workstation*. Herramienta de propósito general para crear y utilizar modelos de datos neuronales.

Documento. En general se le llama documento a una serie de datos que son enviados en conjunto desde una de las entidades supervisadas hacia la entidad supervisora central.

Drill-down y Drill up. Son técnicas de navegación de datos para usuarios que analizan a detalle la información (down) o adicionan los datos a un nivel de sumanzación (up) dentro de una dimension

DSS. Sistemas para el Soporte de Decisiones. Sistemas automatizados de aplicación que ayudan a la organización a tomar decisiones relacionadas con el negocio

EDA/SQL. *Enterprise Data Access/SQL.* Herramienta que propone *Information Builders* para Data Warehouse.

EIS. Sistemas de Información Ejecutiva. Término común que usan los sistemas de consulta y generación de reportes que ejecutan adiciones y resúmenes directamente en los datos operacionales, algunas veces guardando datos resumidos y agregados en forma privada, y proporcionando capacidad de consultas y generación de reportes a quienes toman decisiones (ejecutivos).

Encolar. El procesador central debe distribuir su tiempo para atender a todos los procesos de la aplicación, para ello, los procesos se tienen que interrumpir y para que se reanuden, se tienen que "formar" en "colas" de procesos, a esto se le llama encolar un proceso.

Entidad Supervisada. Es una institución financiera que debe entregar algún tipo de informe a la Entidad Supervisora.

Estándares de programación. Los estándares de programación, como se refieren en este trabajo son, principalmente, convenciones de estilo de programación, que faciliten el mantenimiento de las aplicaciones desarrolladas, así como su documentación para archivo.

Escalabilidad. Es la propiedad de algunos equipos de poder aumentar su capacidad de proceso, almacenamiento, etc. sin necesidad de sufrir cambios sustanciales en su configuración.

Estatus de la bitácora. El estatus de la bitácora se refiere a los registros almacenados en alguna de las bitácoras, que nos pueden ayudar a determinar el estado de alguna transacción, particularmente de aquellas que fallan.

Estatus de las peticiones. Al igual que el estatus de la bitácora, se refiere a los registros que se refieren a las peticiones, en este caso, hablamos del estatus particular de una parte de la bitácora.

Esquema copo de nieve. El esquema de copo de nieve es una extensión del esquema de estrella, donde cada uno de los puntos de la estrella se extiende en más puntos. En esta forma de esquema, las tablas de dimensión del esquema de estrella tienen más normas. Las ventajas que ofrece el esquema copo de nieve son un mejor desempeño de la consulta debido a la minimización del almacenamiento en disco para los datos y un mejor desempeño al juntar tablas más pequeñas con normas en lugar de tablas grandes y sin normas. El esquema copo de nieve también incrementa la flexibilidad de las aplicaciones debido a la normalización y por ende, a la menor granularidad de las dimensiones.

Esquema de estrella. Como el nombre sugiere, el esquema de estrella es un paradigma de modelado que tiene en medio un objeto único conectado a varios objetos en forma radial. El esquema de estrella refleja la visión del usuario en una consulta empresarial, hechos tales como ventas, compensaciones, pagos o facturas calificadas por una o más dimensiones (por mes, producto, región geográfica). El objeto en el centro de la estrella se llama tabla de hechos y los que se conectan desde la periferia se llaman tablas de dimensión.

Extracción de datos. Modalidad del descubrimiento del análisis de datos, o analizar datos de detalle para revelar relaciones, patrones y asociaciones insospechados o desconocidos.

Extracción. Actividad relacionada con transferir datos de bases de datos operacionales (fuentes de datos) al Data warehouse.

Fabricante Ancla. Fabricante cuyos productos son utilizados para definir las estrategias de software y hardware de un proyecto de *DW*. A partir de un fabricante ancla, se contratan servicios o productos compatibles a los proporcionados por éste.

Filtrar y cotejar. Eliminar datos sólo operacionales y seleccionar datos operacionales que pide el modelo de diseño de datos del *DW*, por ejemplo, cotejarlos con el área o segmento temáticos deseados

Filtro. Un filtro se refiere a seleccionar información específica de entre los datos, o bien, impedir que algunas personas tengan acceso a la información.

FOCUS. Tecnología de bases de datos.

Foreign keys. Llave foránea. Llave que hace referencia a otras tablas donde es llave primaria.

Front-ends. Programas generalmente de ambiente gráfico que permiten manipular información para la emisión de reportes finales, tales como Excel, PowerPoint, etc

Gateways. Puentes de acceso para comunicación entre diferentes equipos de cómputo a través de interconexión de redes de computadoras.

Gigabyte (GB). Unidad de información igual a un billón de bytes.

GIS. Se aplica a la integración de sistemas de ubicación.

Granularidad. Término que se utiliza en los Data Warehouses para expresar el nivel de detalle. A más alto nivel de granularidad, más bajo nivel de detalle, más bajo nivel de granularidad mas alto nivel de detalle.

Heurístico. Modo de análisis en el cual el siguiente paso es determinado por el resultado del actual paso de análisis. Utilizado por el proceso de soporte a la toma de decisiones.

IDAPI. Es un front-end.

IDIS. Information Discovery System. Herramienta para descubrimiento de conocimientos propuesta por Information Discovery.

Ingeniería de software. La ingeniería de software se refiere a un conjunto de procedimientos y estándares que se utilizan para dar reutilizabilidad a los proyectos de desarrollo de aplicaciones.

Integridad de la información. La integridad de la información es una métrica para determinar que los datos sean eficientes como materia prima de los sistemas de información.

Integridad Referencial. En el esquema entidad-relación, todas las llaves foráneas se tienen que encontrar asociadas con una llave primaria válida, o en su defecto, tener un valor NULL.

IMS. Tecnología de bases de datos.

JCL. Siglas de Job Control Language. Lenguaje de programación en ambientes de desarrollo de mainframes. Se utiliza para generar procesos que se ejecutan en batch.

LINUX. Sistema operativo similar a UNIX de distribución gratuita.

Logín. Solicitud de palabra de acceso para cualquier sistema o aplicación.

Macintosh. Computadoras personales con formato diferente a las IBM compatibles.

Main Frames. Equipos físicamente muy grandes que se utilizaban como los grandes equipos de cómputo, por su capacidad de almacenamiento y de proceso.

Mantenimiento. El mantenimiento de software se refiere a casi cualquier modificación que se realice sobre el programa una vez que éste ha sido liberado para producción.

Métrica. Unidad de medida.

Metadata. Ver la definición en el capítulo I.

Middleware. Es el término que se aplica al software que intercambia información en forma transparente entre aplicaciones y bases de datos. El middleware ofrece un mecanismo de conexión abstracta entre el software de aplicación y la base de datos, y oculta al programador de aplicaciones los elementos específicos que dependen de la implementación.

Modelo Conceptual de Datos. Es una forma de representar la información que nos facilita el diseño del modelo entidad-relación para bases de datos.

Modelo Entidad Relación. Éste es una representación gráfica que ayuda a representar el "mundo real" en función de los datos y sus relaciones.

MoreRecentUtilice/LestRecentUtilice (MRU/LRU). Esta es una filosofía que se utiliza para el manejo del caché para grandes volúmenes de información, y se basa en mantener en el caché los datos más recientemente utilizados, mientras que todos los datos (menos recientemente utilizados) entran en él; éstos se sacan conforme haya otros datos más utilizados.

MOM. Siglas de Message Oriented Middleware. Middleware orientado al control y manejo de mensajes.

MVS. Sistema operativo de IBM.

Navegadores de cubo.

Neural IDIS. Herramienta que combina la tecnología de red neuronal con descubrimiento de reglas.

Normalización. Es una técnica de modelado de datos que se utiliza comúnmente en los sistemas OLTP. El objetivo de esta técnica de diseño, es el mantener una estructura de datos en la que los datos de una entidad tenga una sola relación uno a uno con su llave principal.

NT. Sistema operativo para redes de Microsoft.

OC timer. Proceso del Open Client de Sybase que se utiliza para disparar procesos y eventos.

ODBC. Se trata de un front-end.

OLAP. Siglas de On-Line Analytical Processing (procesamiento analítico en línea). Tecnología de análisis multidimensional de datos y capacidad de generación de reportes.

OLE. Siglas de Object Link Embeded.

OLTP. Siglas de On-Line Transaction Processing (procesamiento de transacciones en línea). El término se usa para definir cualquier sistema de software que reúne datos usando las transacciones (en el momento en que ocurren) entre la fuente de datos y la base de datos.

Open Client. Herramienta de desarrollo de Sybase enfocado a la arquitectura Cliente-Servidor específicamente a la parte del cliente.

Open Server. Herramienta de desarrollo de Sybase enfocada a la arquitectura Cliente-Servidor específicamente a la parte del Servidor.

OS/2. Sistema operativo de IBM

Petición. En el esquema Cliente-Servidor, se llama petición a todo intercambio de información que se genere desde el cliente, y que deba ser atendido por el servidor.

Procedimiento Remoto. Es un programa que se ejecuta en una computadora distinta de aquella que originó su ejecución.

Power Builder. Herramienta de desarrollo orientada a objetos fabricada por Sybase

Protocolo. Programa que se utiliza como enlace para comunicar a dos equipos. TCP/IP, IPX, NETBEUI, etc.

Raw data. Datos puros sin procesar.

RDBMS. Siglas de Relational Database Management System (sistema de administración de bases de datos relacionales). Sistema para almacenar datos construido alrededor del modelo relacional con base en tablas, columnas y vistas.

Reingeniería. Procedimiento mediante el cual se rehacen los procesos con nuevas tecnologías para modificar áreas susceptibles de mejora.

Relaciones. En el modelo Entidad-Relación, las relaciones son precisamente representaciones esquemáticas de la relación que existe entre los datos de las entidades.

Remote Procedure Call (RPC). Un RPC es un proceso especial que genera una petición para que se ejecute un procedimiento remoto.

Replication. "Replication" es un término en inglés cuyo significado se refiere a réplica, y es volver a enviar información que se ha recibido, en este trabajo particularmente se refiere a los Replication Servers.

Replication Server. Servidor de Replicación. Es el proceso mediante el cual se crean y mantienen asociadas copias de los datos originales en diferentes sitios. En una arquitectura de replicación existe el que publica los datos y el que los recibe. El que publica es el que tiene el control del origen de la información.

Repositorio. Almacén de datos de tamaño considerable, en el cual se pueden almacenar grandes cantidades de información.

ROLAP. Procesamiento de Análisis en Línea Relacional

Rolling. Proceso para la carga de la información consistente en el almacenamiento de los datos sumerizados de distintos niveles.

Roll-up.

S/370. Sistemas modelo 370 de IBM.

SA. Ver Servidor Aplicativo.

SAG-CLI. Se trata de un front-end.

SDLC. Ciclo de vida del Diseño de Sistemas.

Server Library. Librerías de apoyo para la herramienta Open Server de Sybase.

Servidor. Es una parte medular del esquema Cliente-Servidor, y es una de las computadoras que realizan los procesos importantes en las aplicaciones.

Servidor Aplicativo. Un servidor aplicativo tiene la función principal de almacenar grandes volúmenes de información que debe estar disponible para muchos usuarios simultáneos, característicamente, los usuarios sólo pueden leer esta información. En el Capítulo 4 encontramos una definición adecuada a este trabajo.

Servicios de validación. En nuestro trabajo nos referimos como servicios de validación a los procesos de validación de datos que se ejecutan en un servidor determinado.

Sleep. "Sleep" es un estado, en este estado, los programas se encuentran sólo esperando a que sean requeridos.

Sybase. Sybase es una corporación que se ha dedicado al desarrollo de manejadores de bases de datos, y recientemente al desarrollo de herramientas con arquitectura cliente-servidor.

Sistemas de misión crítica. Un término de uso común para describir una aplicación de software que es esencial para la operación continua de una empresa. La falla de este sistema afecta la viabilidad de la empresa.

Sistemas Operacionales. Un sistema operacional es el que apoya a las actividades diarias de la operación de las empresas. Generalmente corre en un ambiente OLTP.

SMIS. Siglas de *Sales and Marketing Intelligent System*. Herramienta para usuarios y profesionales en tecnología de la información, que accesa y analiza la información empresarial de misión crítica.

SMTI. Siglas de System Management Tools Initiative. Agrupación de Oracle que cubre las necesidades de administración de sus tareas y la vigilancia del desempeño.

Snow-flake. Ver Esquema copo de nieve.

SPSS. Aplicación estadística.

SQL. *Structured Query Language*. Lenguaje estructurado para consultas. Es el lenguaje estándar para el acceso a bases de datos relacionales. SQL se define y estandariza mediante comités internacionales ANSI.

SQL Server. Servidor de SQL utilizado por Sybase.

Store Procedure. Procedimiento almacenado en la metadata y que es llamado mediante otros procedimientos.

Terabytes (TB). Unidad de información igual a un trillón de bytes.

Thread. Proceso de ejecución interna que se ejecuta de acuerdo a las queues, y que se "duermen" o "despiertan".

Tipo de dato. Un tipo de dato es una forma de clasificación de los datos primitivos en categorías o tipos específicos, a partir de los cuáles se pueden generar estructuras, crear tablas, manejar fórmulas, en general, definir el "mundo real".

Triggers. Un trigger es un tipo de bandera, en la mayoría de los ambientes de los manejadores de bases de datos, al tener éstos un valor específico desencadenan (disparan) procesos diversos.

Transacción. Operación que se realiza para actualizar, añadir, consultar o borrar información en una base de datos.

UNIX. Sistema operativo

Usuario. Tal vez la parte más importante de los sistemas en general, el usuario es el beneficiario de todo el esfuerzo de desarrollo, y es quien determina la mayor parte de la funcionalidad y el diseño del mismo.

Validación. Proceso de verificación de la información, dónde se determina si es válida de acuerdo a los parámetros establecidos.

Warehouse. Almacén de información.

WTI. Siglas de *Warehouse Technology Initiative* Agrupación que conforma Oracle con algunos socios para integrar proyectos de Data Warehouse.