Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería

Escuela de Ingeniería Civil en Informática

Desarrollo de una metodología que permita a empresas el desarrollo de un Data Warehouse y su integración con Sistemas Workflow utilizando herramientas de libre distribución y/o bajo costo.

Tesis para optar al título de
Ingeniero Civil en Informática

PROFESOR PATROCINANTE:
ING. JUAN PABLO SALAZAR FERNANDEZ

ESTEBAN ALBERTO KEMP DE LA HOZ

VALDIVIA – CHILE

2005

Valdivia, 29 de junio de 2005

De: Juan Pablo Salazar Fernández
Profesor Auxiliar
Instituto de Informática

A: Miguelina Vega R.
Directora Escuela de Ingeniería Civil en Informática

Ref: Calificación proyecto de título

De mi consideración:

Habiendo revisado el trabajo de titulación **"Desarrollo de una metodología que permita a empresas el desarrollo de un Data Warehouse y su integración con sistemas Workflow utilizando herramientas de libre distribución y/o bajo costo",** presentado por el alumno sr. Esteban Kemp De la Hoz, mi evaluación del mismo es la siguiente:

Nota: 7,0 (siete coma cero).

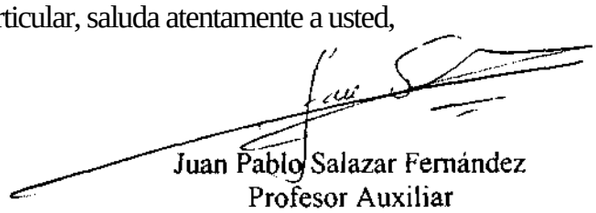
Fundamento de la nota:

El presente trabajo de titulación se planteó como objetivo el acercamiento a las PYMEs a herramientas de apoyo a la gestión, definiendo una metodología, evaluando e integrando herramientas de libre distribución y desarrollando un prototipo que validara lo anterior. Al respecto, los objetivos han sido cumplidos a cabalidad, generándose además una serie de conclusiones útiles para futuros trabajos y que tienen que ver principalmente con las barreras a la entrada que establecen las PYMEs frente a este tipo de tecnologías y que vienen dadas por aspectos no sólo económicos sino también culturales.

Además, este trabajo constituye un aporte para la Universidad Austral de Chile ya que permite generar vínculos con empresas de la zona y explorar alternativas de transferencia tecnológica que ayuden a mejorar la competitividad de las PYMEs.

Aspecto	**Evaluación**
Cumplimiento de objetivos	7
Satisfacción de alguna necesidad	6.8
Aplicación del método científico	7
Interpretación de los datos y obtención de conclusiones	7
Originalidad	7
Aplicación de criterios de análisis y diseño	7
Perspectivas del trabajo	7
Coherencia y rigurosidad lógica	7
Precisión del lenguaje técnico	6.8

Sin otro particular, saluda atentamente a usted,



Juan Pablo Salazar Fernández
Profesor Auxiliar
Instituto de Informática

Valdivia, 11 Julio del 2 005

DE : Miguelina Vega Rosales

Profesor Instituto Informática

A: Dirección Escuela Ingeniería Civil en Informática

Informo a usted que el Proyecto de Título " Desarrollo de una Metodología que permitan a empresas el desarrollo de un Data Warehouse y su Integración con Sistemas Workflow utilizando herramientas de libres distribución y/o bajo costo", presentado por el señor Esteban Alberto Kemp de la Hoz, cumple con el objetivo general propuesto, que es construir una metodología para el desarrollo Data Warehouse que Integre Sistemas Workflow para empresas de tamaño medio.

La metodología de trabajo y el lenguaje utilizado es el adecuado, sin embargo, no queda claro si la implantación fue realizada completamente. Por lo anteriormente expuesto, califico este proyecto de título con nota 6,5 (seis, cinco).

Atentamente

Miguelina Vega R.

Universidad Austral de Chile
Instituto de Informática

Valdivia, 11 de julio de 2005.

De : Luis Hernán Vidal Vidal.
A : Sra. Miguelina Vega R.
Directora de Escuela de Ingeniería Civil en Informática.
Ref.: Informa Calificación Trabajo de Titulación.

MOTIVO: Informar revisión y calificación del Proyecto de Título "Desarrollo de una metodología que permita a empresas el desarrollo de un Data Warehouse y su integración con Sistemas Workflow utilizando herramientas de libre distribución y/o bajo costo", presentado por el alumno Esteban Alberto Kemp de la Hoz, que refleja lo siguiente:

Se logró el objetivo planteado que permitió construir una metodología para el desarrollo de data warehouse que integre sistemas workflow para empresas de tamaño medio.

La revisión hecha sobre los estándares y tecnologías, junto al desarrollo propuesto, se presenta como una buena referencia para futuros trabajos en esta área.

Cumplimiento del objetivo propuesto.	7,0
Satisfacción de alguna necesidad.	7,0
Aplicación del método científico.	7,0
Interpretación de los datos y obtención de conclusiones.	7,0
Originalidad.	7,0
Aplicación de criterios de análisis y diseño.	7,0
Perspectivas del trabajo.	7,0
Coherencia y rigurosidad lógica.	7,0
Precisión del lenguaje técnico en la exposición, composición, redacción e ilustración.	7,0
Evaluación Tesis.	**7,0**

Por todo lo anterior expuesto califico el trabajo de titulación del Sr. Esteban Alberto Kemp de la Hoz con nota 7,0 (siete coma cero).

Sin otro particular, se despide atentamente.

Ing. Luis Hernán Vidal Vidal.
Profesor Instituto de Informática.
Facultad de Ciencias de la Ingeniería.
Universidad Austral de Chile.

Agradecimientos.

A mi madre Gloria y a mi padre Gary por su apoyo incondicional, su paciencia a lo largo de este proceso y sus esfuerzos por brindarme la educación y valores que he recibido.

A mi hermana Andrea y a mis abuelos Alba y Alberto por su ayuda y cariño durante toda mi carrera que contribuyó a que este proceso fuese más fácil.

A mi polola Gabriela por su perseverancia, paciencia y amor para empujarme día a día a alcanzar este objetivo.

Índice.

Índice de Figuras.

Síntesis.

El siguiente proyecto de tesis propone una metodología formal para la construcción de un sistema data warehouse en empresas PYMEs integrando herramientas de workflow para los procesos de extracción, transformación y carga de datos y proponiendo, para la implementación, la utilización de herramientas de libre distribución.

La metodología propuesta es de carácter evolutivo y descentralizado utilizando el enfoque propuesto por Kimball [Kim+, 98] de dimensiones y hechos conformados, que permite a las organizaciones emprender proyectos de data warehouse de manera gradual y programada a lo largo del tiempo.

Proponemos a través de la metodología la integración costo-efectiva de las fuentes de datos operacionales de una organización en un repositorio lógico común, orientado a realizar gestión y a enriquecer los procesos de toma de decisiones mediante la entrega de información relevante, correcta y oportuna.

Proponemos un conjunto de herramientas de libre distribución bajo la arquitectura J2EE que cumplen con los requerimientos de un data warehouse en sus diferentes niveles y que permiten a una PYME alcanzar una implementación costo-efectiva.

Abstract.

The following thesis project proposes a formal methodology for the construction process of a data warehouse system focused on PYMEs with the integration of workflow tools for the extraction, transform and load process and, for the implementation phase, the use of open source tools.

The proposed methodology is of evolutionary and decentralized character, based on the approach of Kimball [Kim+, 98] centered on conformed dimensions and facts, which allow to the organization to undertake a data warehouse project in a gradual and programmed way in the time.

Through the methodology, we propose, the cost-effective integration of the operational data sources of an organization in a common logical repository oriented to answer management questions and to enrich the processes of decision making with excellent, correct and opportune information.

We proposed a set of open source tools over the J2EE architecture that fill all the requirement of a data warehouse in all his levels, and that allow to a PYME to reach a cost-effective implementation.

Introducción.

La incuestionable importancia que tienen las pequeñas y medianas empresas (PYMEs) en la economía nacional, como generadoras de empleo (sobre 1.700.000 empleos [Ale, 2004]) y de desarrollo, sumado a la creciente globalización económica de la que son parte, les ha planteado a éstas un gran desafío, en cuanto a la imperiosa necesidad de automatizar y optimizar sus procesos de negocios y de toma de decisiones, con el objetivo de sostener y mejorar su competitividad en el mercados.

El manejo correcto y oportuno de la información es sin duda alguna un factor crítico de éxito para estas organizaciones. “La información”, es el corazón de cada negocio, y el ingrediente natural de enriquecer sus procesos de toma de decisiones.

Consideramos que la información es similar a un activo para la organización, cuyo valor económico en teoría corresponde al valor que el mercado está dispuesto a pagar por esta información. Sin embargo, claramente, esto no es suficiente ya que la información no esta a la venta y es generalmente utilizada sólo en procesos internos de toma de decisión de la empresa.

> “El valor de la información puede ser visto como la diferencia económica entre una decisión correcta y una decisión equivocada, en donde la decisión está basada en esa información. Mientras más grande sea la diferencia entre decisión correcta y errónea, mayor será la importancia de contar con una buena información”. [Bit, 2002].

En la práctica, por diversas circunstancias, y a pesar de que la importancia de la información para una organización pareciese de Perogrullo, muchas empresas no realizan los esfuerzos necesarios para mejorar la calidad

y disponibilidad de la información que sus propios procesos de negocio generan. En el mundo PYME esto se agudiza, la vorágine del día a día mantiene los ojos vendados para muchas de estas organizaciones, que no logran ver más allá de lo evidente y de sus problemas que deben resolver cotidianamente.

Durante las últimas décadas la informática ha provisto de las tecnologías necesarias para permitir a las organizaciones mejorar sus procesos de toma de decisiones. Es así como han proliferado en las grandes organizaciones sistemas como los DSS (Decision support system), sistemas de data warehouse, Data Mining entre otros.

Estas tecnologías se han mantenido relativamente fuera del alcance de las PYMES, esto aparentemente debido a su alto costo. Sin embargo, esta no es la única razón; en general, pareciera que para los empresarios PYME las tecnologías de información no se presentan como un obstáculo relevante para su desarrollo y por lo tanto no invierten en ellas; sin embargo, esto diverge radicalmente con el enfoque de investigadores, legisladores y autoridades públicas, para los cuales las tecnologías de información tienen un alto grado de importancia [Rob, 03].

Creemos que las PYMEs deben realizar los esfuerzos necesarios para mejorar sus procesos de toma de decisiones, incorporando nuevas tecnologías de información.

Consideramos que el proceso de incorporación de nuevas tecnologías de información en las PYMEs debe ser progresivo, cada nueva inversión debe ser realizada en conciencia de su impacto y de su forma de uso. Las PYMEs deben maximizar la utilización de cada tecnología en la que inviertan, evitando el hecho común de realizar una inversión y posteriormente utilizar solo un porcentaje menor de las características de la nueva tecnología.

Nos proponemos demostrar que el costo no es una limitante para estas empresas y que es posible optimizar, mediante la integración de tecnologías, los procesos de toma de decisión.

En este documento presentamos una metodología formal para la construcción de un sistema data warehouse enfocado a empresas del segmento PYME, las cuales no tienen capacidad económica para acceder a las sofisticadas herramientas comerciales que abundan en el mercado.

Proponemos una metodología evolutiva e independiente del software, que permita a las empresas afrontar este desafío de manera progresiva y realista.

Proponemos, también, un conjunto de herramientas de software open source que validen los requerimientos de la metodología a costo de licenciamiento cero.

Por último, proponemos la integración de la tecnología Workflow para la automatización y control de aquellos procesos rutinarios que involucra un sistema data warehouse.

Objetivos.

Objetivos Generales.

- Construir una metodología para el desarrollo de data warehouse que integre sistemas workflow para empresas de tamaño medio.

Objetivos Específicos.

- Estudiar y analizar las metodologías actuales para el desarrollo de Data warehouse y de sistemas workflow.

- Formular una metodología que permita a una empresa de tamaño medio integrar sus fuentes de información de manera de poder realizar análisis y tomar decisiones de manera costo-efectiva.
- Estudiar y definir herramientas que permitan aplicar la metodología propuesta de manera costo efectiva.
- Validar esta metodología en alguna empresa del sur de Chile mediante el desarrollo de un caso práctico.

Parte 1: Modelo de desarrollo de un Data Warehouse.

1. Características Generales.

Las empresas, hoy en día, utilizan diferentes sistemas de información (SI) que varían según su propósito, pudiendo clasificarlos de la siguiente manera [KEN, 1991]:

- Sistemas de procesamiento de transacciones.
- Sistemas de automatización de oficina y de empleo de conocimiento.
- Sistemas de información gerencial.
- Sistemas de apoyo a la decisión.
- Sistemas expertos, e inteligencia artificial.

En las organizaciones existen distintos tipos de SI. Desde el punto de vista de la estructura funcional, los SI se forman alrededor de las funciones de la empresa (Recursos Humanos, Producción, Mercadotecnia, etc.) y cada una de estas funciones comprende actividades en tres niveles de la organización, ver figura 1:

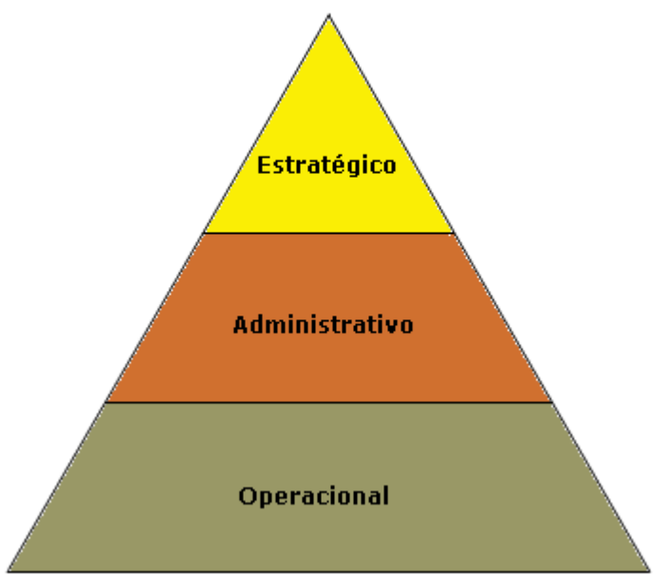


Figura 1: Niveles de la organización

Dentro de los sistemas orientados a proveer de información de gestión a los niveles administrativos y estratégico de las organizaciones se encuentran los sistemas data warehouse, que permiten utilizar la información generada a nivel operacional por la empresa como una potente herramienta a la hora de generar información de gestión.

Para lograr este objetivo se plantea la reorganización de la información operacional en un sistema paralelo y redundante, es decir, un rediseño de la estructura lógica de la información, esta vez con miras al negocio (y no a la operación), de manera de generar un repositorio de información fuertemente orientado al tema.

Un sistema data warehouse se concibe como un sistema que se alimenta periódicamente de diversos sistemas operacionales, manteniendo la congruencia y que está íntegramente orientado a atender consultas. Permite el almacenamiento organizado de información histórica, bajo la certeza que ésta no variará en el tiempo (a diferencia de un sistema operacional), ésta es una característica muy importante ya que provee a los niveles ejecutivos de información confiable acerca de hechos del pasado, sobre la cual poder basar las decisiones que afectarán a la empresa en el futuro.

Los proyectos de data warehousing, en general, son proyectos de largo plazo que involucran múltiples etapas a través de las diferentes áreas de negocio de una organización y cuya principal dificultad se encuentra en percibir correctamente las características y naturaleza del negocio, así como también, en el direccionamiento adecuado de los requerimientos de los usuarios.

1.1 Costos v/s Valor de un data warehouse.

Para cualquier proyecto es importante e inevitable realizar un análisis que involucre y sopese tanto los costos como los beneficios. Para el caso de un proyecto de data warehousing los costos principales que intervienen son los costos de construcción del sistema y los costos de mantención y operación del mismo. En cuanto a los beneficios se consideran la mejor entrega de información, la mejora del proceso de toma de decisiones y el valor agregado sobre los procesos empresariales.

1.1.1 Costos de un data warehouse.

Costos De Construcción.

Los ítems que involucran costos de construir un data warehouse son similares a los de cualquier proyecto de tecnología de información. Éstos pueden ser clasificados en dos categorías [Wol, 2002]:

- **RRHH**: Se requiere la colaboración del personal de la empresa que domine los conceptos y característica del negocio, además de los especialistas tecnológicos.
- **Tecnología:** El proyecto puede ir asociado a nuevas tecnologías las cuales tendrán un costo determinado.
- **Tiempo:** El costo de oportunidad asociado a no poseer el sistema.

Costos De Operación.

Una vez que está construido y entregado un data warehouse debe ser soportado para que tenga valor empresarial. Son justamente estas actividades de soporte, la fuente de continuos costos operacionales para un data warehouse. Se pueden distinguir tres tipos de costos de operación [Wol, 2002]:

- Evolutivos
- Crecimiento
- Cambios

1.1.2 Valor de un data warehouse.

El valor de un DW queda descrito en tres dimensiones [Wol, 2002]:

- Mejorar la entrega de Información: información completa, correcta, consistente, oportuna y accesible. Información que la gente necesita, en el tiempo que la necesita y en el formato que la necesita.
- Mejorar el Proceso de Toma de Decisiones: con un mayor soporte de información se obtienen decisiones más rápidas; así también, el personal responsable de las decisiones de negocio adquiere mayor confianza en sus propias decisiones y en las del resto, y logra un mayor entendimiento de los impactos de sus decisiones.
- Impacto Positivo sobre los Procesos Empresariales: cuando al personal se le da acceso a una mejor calidad de información, la empresa puede lograr por sí sola:
    - Eliminar los retardos de los procesos empresariales que resultan de información incorrecta, inconsistente y/o no existente.
    - Integrar y optimizar procesos empresariales a través del uso compartido e integrado de las fuentes de información.
    - Eliminar la producción y el procesamiento de datos que no son usados ni necesarios, producto de aplicaciones mal diseñadas o ya no utilizadas.

1.1.3 Balance Costo v/s Valor.

Lograr una cuantificación económica de los factores de valor no es fácil ni natural, a diferencia de los factores de costos, agregar valor económico a los factores de valor resulta ser en extremo complejo y subjetivo. Una alternativa es hacer una valoración desde la perspectiva de costos evitables, relacionados con los “costos de no disponer en la organización de información apropiada”, tanto a un nivel técnico como de procesos empresariales (en especial, para el proceso de Toma de Decisiones). [Nad, 2004].

En un estudio encargado a la compañía Gartner Group por 20 vendedores y consultores, se encontró un Retorno Promedio Total de la Inversión (Return On Investment-ROI) de 401% en 2.3 años. El estudio se realizó sobre 62 organizaciones que implementaron sistemas de apoyo gerencial basados en un data warehouse. En este estudio se excluyeron los proyectos fracasados, así como los ejecutados por fuera del cronograma, debido a que sólo interesan los proyectos que fueron ejecutados e implementados correctamente desde el punto de vista de todas las áreas de Ingeniería de Software (fundamentalmente Planificación y Gestión de Cambios) (ver tabla 1) [Nad, 04].

Tipo ROI	Valor
ROI promedio total	401%
ROI promedio del proyecto más grande	322%
ROI promedio del modelo complementario de datos	533%
ROI mediano	160%
Período de reembolso promedio	2.3 Años

Tabla 1. ROI de proyectos de data warehouse.

1.2 Elementos de un data warehouse.

Los elementes básicos de un data warehouse se describen el la figura 2:

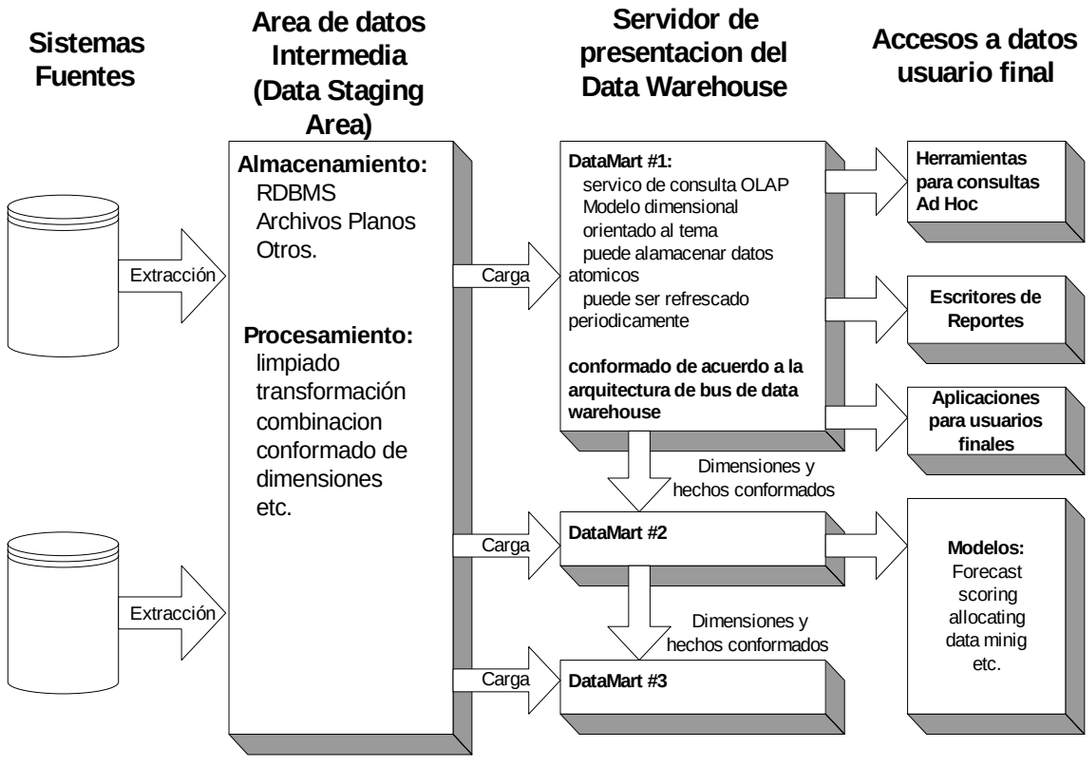


Figura 2: Elementos de un data warehouse

- **Sistemas fuentes.** Un sistema operacional de registro, cuya función es capturar las transacciones del negocio [Kim+,98]. El sistema fuente tiene por objetivo mantener la información actualizada y disponible en todo momento. Las consultas ejecutadas contra este sistema son generalmente pequeñas y parte del flujo normal de transacciones.
- **Área de datos intermedia (Data Staging Area).** Un área de almacenamiento y un conjunto de procesos que limpian, transforman,

combinan, archivan y preparan los datos de origen para su uso en el data warehouse. [Kim+,98]

- **Servidor de presentación.** La maquina física en la cual la información que contiene el data warehouse, es organizada y almacenada para ser consultada directamente por los usuarios finales, las aplicaciones de reporte y otras aplicaciones. [Kim+,98]. Una de las tecnologías más utilizadas a este nivel es la tecnología OLAP (On-Line Analytical Process) que permite navegar y analizar información contenida en bases de datos multidimencionales de manera dinámica realizando operaciones típicas como drill-down, slice, dice, roll-up etc.
- **Acceso a datos de usuarios finales.** El cliente del data warehouse, un conjunto de aplicaciones o interfases mediante las cuales el usuario tiene acceso a los datos almacenados en el data warehouse.

1.3 Metodología de desarrollo de un data warehouse.

El proceso completo de construcción abarcará tres grandes etapas identificadas como: **Análisis de Requerimientos**, **Diseño** e **Implementación**, cada una de las cuales a su vez incluye una serie de sub-etapas que se propone deberán ejecutarse conforme se señala en el figura 3.

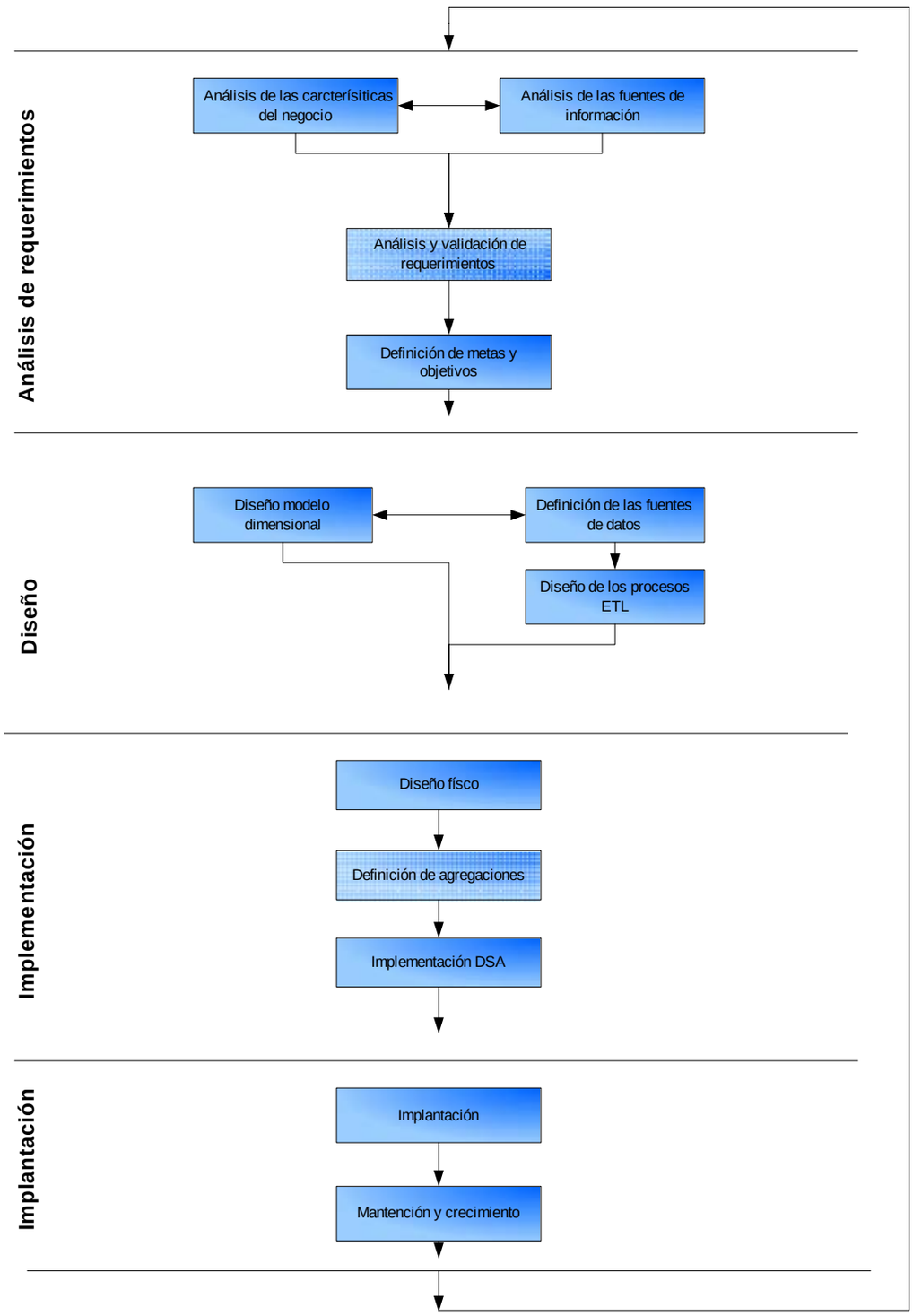


Figura 3: Metodología de Desarrollo.

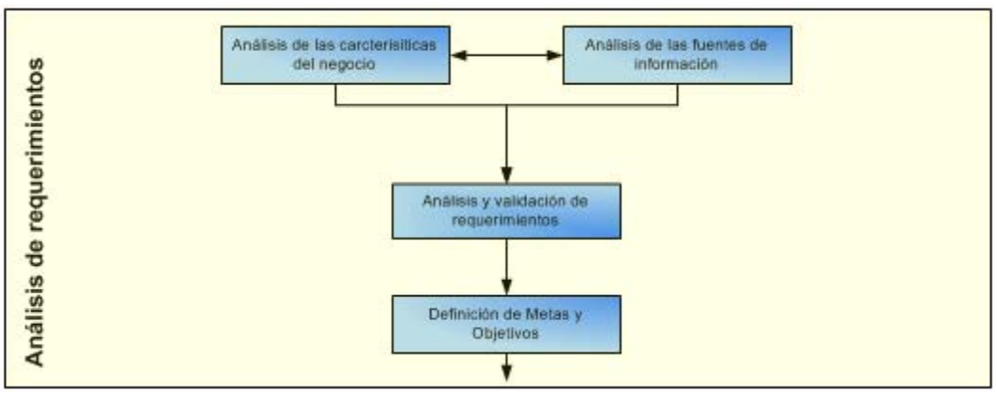
En adelante, se describirá en detalle cada una de las etapas de la metodología de desarrollo.

2. Análisis de Requerimientos.

2.1 Objetivos.

Al finalizar esta etapa se debe haber definido:

- Metas generales y específicas del proyecto.
- Las características generales y específicas del negocio.
- Los factores críticos de éxito del negocio.
- La información que debe estar disponible en el sistema, cómo se organiza y qué tan frecuente deben ser las actualizaciones.
- Validar los requerimientos y necesidades desde la perspectiva de una PYME.



2.2 Análisis de las características del negocio.

Como es de imaginar la clave principal para el éxito de nuestro proyecto es la correcta captura de los requerimientos desde los usuarios del negocio. Este es, entonces, el lugar por donde debemos comenzar.

Debemos entrevistarnos directamente con los usuarios del negocio. Por ningún motivo debemos prescindir de conversar con quienes serán nuestros usuarios finales. Si bien esto parece de Perogrullo, existen situaciones en las cuales nos podemos sentir tentados en omitir las conversaciones con los usuarios finales, sustentados en la seguridad que parece mostrar algún experto en sistema sobre el conocimiento de sus usuarios y considerando que es mucho más cómodo sostener entrevistas con los expertos en sistemas o con algún DBA, con los cuales tenemos un lenguaje común.

Sin embargo las verdaderas claves del negocio no se encuentran en los expertos en sistemas. Las claves del negocio las encontraremos en los usuarios, cada uno de los cuales aportará con las piezas que nos permitirán armar el puzzle del negocio.

Debemos, entonces, como primera tarea coordinar entrevistas con los diferentes usuarios que utilizarán nuestros sistemas. Al establecer entrevistas formales con los usuarios del negocio debemos, como segunda tarea, establecer un lenguaje en común; esto es, aclarar las terminologías comunes dentro de la empresa, no debemos olvidar que es muy probable que una misma palabra tenga significados diferentes desde nuestra perspectiva técnica a la perspectiva del usuario del negocio.

Al mismo tiempo que capturamos la información de los usuarios del negocio debemos comprobar la disponibilidad de esa información en los

sistemas fuentes de datos a través de entrevistas con los expertos de sistemas (punto 2.3).

La definición de los usuarios del negocio a los que entrevistaremos es muy importante. Dependiendo el tamaño de la empresa puede que no sea posible entrevistar a todos los usuarios relacionados con el negocio, si este es el caso, debemos obtener una muestra representativa. Para esto, podemos comenzar por obtener el organigrama de la empresa y definir junto con el impulsor (sponsor) del proyecto un conjunto de usuarios que atraviese verticalmente la organización. Otro factor importante a considerar para esta selección, es la estructura informal de la organización, es decir obtener una evaluación subjetiva de las capacidades, habilidades e importancia de los diferentes usuarios. En todas las organizaciones existen usuarios que, ya sea por habilidad o experiencia, tienen un mejor manejo del negocio, estas características no aparecen en ningún organigrama, sin embargo, debemos identificar a estos usuarios y entrevistarnos con ellos.

Es muy probable y altamente recomendable que nuestro proyecto comience con un área acotada del negocio; sin embargo, a la hora de seleccionar nuestros entrevistados, es una buena práctica incluir representantes de las otras áreas de la empresa para así tener una visión global del negocio, que nos permita asegurar la escalabilidad de nuestro data warehouse.

Es importante también comenzar por usuarios de nivel ejecutivo para obtener una visión de alto nivel del negocio que nos permita entender las estrategias globales de la empresa. A modo de ejemplo, podríamos utilizar algún modelo general para clasificar la estrategia competitiva de la empresa (ver tablas 2 y 3).

	Liderazgo en costos	Diferenciación	Concentración
Diferenciación del producto.	Bajo (principalmente por precios)	Alta(principalmente por exclusividad)	Baja a alta(precio o exclusividad)
Segmentación del mercado.	Bajo (Mercado masivo)	Alta(varios segmentos de mercado)	Baja(uno o pocos segmentos)
Habilidades Distintivas	Fabricación y administración de materiales	Investigación y desarrollo, ventas y marketing	Cualquier tipo de habilidad distintiva

Tabla 2: Estrategias genéricas competitivas [Hil+,01]

De igual manera es importante involucrarse con los objetivos a largo y mediano plazo de la organización y esto implica conocer la misión y visión de la empresa.

	Existentes	Nuevos
Existentes	Penetración en el Mercado	Desarrollo de productos
Nuevos	Desarrollo del mercado	Proliferación de productos

Tabla 3: Estrategias competitivas [Hil+,01]

Con un entendimiento global de las políticas de la empresa y cómo ésta se relaciona con el mercado debe ser nuestro primer objetivo que nos servirá como marco de trabajo para entender y direccionar mejor los requerimientos de los usuarios.

En los niveles medios entenderemos cómo estas estrategias globales se concretan en tácticas de negocio, en este nivel obtendremos una visión más concreta del negocio.

Otras consideraciones adicionales son:

- Conseguir compromiso de parte de los entrevistados, esto generalmente se consigue si la organización refleja su interés por el proyecto a sus subordinados.
- Programar las entrevistas, tratando de no hacer coincidir dos entrevistas en el mismo día, esto principalmente asociado a dos factores, el primero es el desgaste que producen este tipo de entrevistas y que nos puede llevar a omitir alguna característica, y segundo a que es útil madurar la

información resultado de cada entrevista ya que nos puede servir para la próxima.

- Considerar al menos dos personas (no más de tres) para cada entrevista, uno encargado de conducir la entrevista y el otro encargado de tomar nota. El uso de grabadoras puede ser muchas veces intimidante para el entrevistado, y si se hace indispensable utilizar, esto debe ser previo acuerdo con el entrevistado.

Ejemplo de Temario/Cuestionario de entrevista:

Introducción:

- Explicar objetivos generales del proyecto
- Explicar los objetivos de la entrevista

Objetivos de Negocio y similares:

- Describa su trabajo y como se relaciona con el resto de la empresa
- Describa cuales son sus responsabilidades
- ¿Cuáles son los objetivos de su organización?
- ¿Cuáles son los objetivos de esta área en específico?
- ¿Cuál es su objetivo principal?
- ¿Cómo mide actualmente su desempeño?
- ¿Utiliza indicadores? ¿Cuáles son sus indicadores de éxito?
- ¿Cómo sabe que lo esta haciendo bien?
- ¿Qué tan frecuentemente evalúa lo anterior?
- A su modo de ver ¿Cuáles son los principales problemas que enfrenta hoy en día su empresa?
- ¿Cuáles son los principales problemas que enfrenta hoy su área e negocio?
- ¿Cómo identifica estos problemas?
- ¿Tiene alguna capacidad de anticipación sobre estos problemas? ¿Cómo funciona?

Requerimientos

- ¿Qué tipo de análisis realiza frecuentemente? ¿Qué información usa para estos análisis?
- ¿Cómo obtiene esta información? ¿Cómo la procesa?
- ¿Qué información utiliza usted actualmente para tomar sus decisiones?
- ¿Cómo usa esta información en sus decisiones?
- ¿Qué problemas tiene actualmente para conseguir la información que necesita?

- ¿Existe información con la cual le gustaría contar y que actualmente no posea?
- ¿Qué capacidades de análisis le gustaría tener?
- ¿Existen cuellos de botella específicos para obtener información?
- ¿Qué necesidades tiene de información histórica?
- ¿Con que profundidad?
- ¿Qué posibilidades ve de optimizar su área en base a optimizar la información que recibe?
- ¿Qué reportes usa actualmente? ¿Cuál son los datos importantes en estos reportes?
- ¿Cómo se traduce esto financieramente?

2.3 Análisis de las fuentes de Información.

En paralelo a las entrevistas con los usuarios del negocio, debemos validar los requerimientos que capturemos contra las fuentes de datos de la empresa, es decir, validar que la información descrita en las conversaciones exista en las fuentes de datos. Para esto debemos realizar una auditoria a los datos y necesitaremos la cooperación de los ingenieros de sistemas de la empresa junto con los DBA o con la o las personas encargadas de mantener los sistemas informáticos operativos. Con estos, al igual que en la etapa anterior, planificaremos un conjunto de entrevistas que nos permitan entender los datos, las definiciones de los campos, la granularidad, los volúmenes, la distribución, etc., todo esto con el objetivo de verificar la factibilidad de satisfacer las necesidades de los usuarios recolectadas en la etapa anterior.

Dependiendo del tamaño de la empresa es posible que no exista un experto en sistemas local; es decir, si la totalidad de las actuales soluciones informáticas de la empresa están en modalidad de arriendo o son administradas externamente, es muy probable que nuestra mejor aproximación a un experto en sistemas sea algún usuario avezado, con experiencia en las aplicaciones pero sin demasiados conocimientos técnicos. En estos casos debemos en lo posible contrastar la información proveniente de este usuario con información proveniente del proveedor de las aplicaciones. Así podremos definir cuáles serán nuestras reales posibilidades de interacción con los sistemas fuente.

Ejemplo de Temario/Cuestionario de entrevista:

Introducción:

- Explicar objetivos generales del proyecto
- Explicar los objetivos de la entrevista

Auditoria de Datos

- Conversar acerca las fuentes de datos principales de los sistemas de la empresa.
- ¿Cómo se distribuye actualmente la información?
- ¿Cuáles son las aplicaciones que actualmente los usuarios utilizan más?
- ¿Cuál es el alcance de esas aplicaciones?
- ¿Cuáles son las tareas que se realizan más frecuentemente?
- ¿Qué tipo de análisis de datos se realizan actualmente?
- ¿Qué requerimientos de análisis realizan los usuarios comúnmente?
- ¿Cómo se realizan esos análisis?
- ¿Qué sabe de DataWarehousing?
- ¿Cómo ve que encaja un data warehouse en su empresa?
- ¿Existen problemas conocidos en la consistencia de los datos?
- ¿Existen datos Históricos? ¿Qué antigüedad? Enfocar orientado al tema.

Este proceso de entrevistas y análisis de las fuentes de datos debe continuar hasta comprobar que se encuentran disponibles los datos para soportar los requerimientos de los usuarios.

Debemos conocer las fuentes de datos no al detalle de un experto del sistema, pero sí lo suficiente para sentirnos cómodos direccionando los requerimientos de los usuarios.

2.4 Análisis y Validación de Requerimientos, desde la perspectiva de las PYMEs.

Las PYMEs son un segmento empresarial complejo y diverso; sin embargo, reúnen características comunes interesantes. Como mencionábamos en la introducción de este trabajo, los empresarios del sector PYME son renuentes a la utilización de nuevas tecnologías de información, estudios establecen que no más de un 35% de las PYMEs utiliza algún software en sus procesos de administración y producción [URL 13]. El Gobierno ha realizado innumerables esfuerzos con el objetivo de cambiar esta realidad, esfuerzos que van desde la implementación de tecnologías obligatorias, como bono electrónico, factura electrónica, etc. de manera de forzar a los empresarios por medio de la "urgencia" a invertir en tecnología, hasta la implementación de franquicias tributarias para fomentar las inversiones TI.

Como mencionábamos también en la introducción, los costos de las tecnologías de información son intimidantes y, generalmente, los compromisos económicos a corto plazo a los que deben responder las PYMEs, alejan aún más la posibilidad de éstas de realizar inversiones a largo plazo, cuyo retorno, peor aún, subestiman [Rob, 03].

Lo cierto es que, en nuestra economía de libre mercado y globalizada, más temprano que tarde, la brecha tecnológica que se está generando conectará un duro golpe a aquellas organizaciones que se resistan a modernizar sus procesos de negocios.

2.4.1 Difusión.

Bajo este contexto consideramos que es clave, antes de continuar con nuestro proyecto y previo a la definición de los objetivos y alcances del mismo, asegurar el compromiso organizacional y por sobre todo la comprensión de lo necesario e impactante que puede resultar la optimización exitosa de los procesos de toma de decisiones. Debemos, entonces, imprimir un sentido de urgencia a nuestro proyecto, de manera de transformarlo en parte del plan estratégico de la organización.

Para conseguir esto debemos tomarnos el tiempo que sea necesario en explicar y publicitar las características y ventajas de estos sistemas, ejemplarizando todas las posibles aplicaciones de las cuales podría sacar ventaja la organización. Debemos utilizar todo el conocimiento que hemos adquirido durante nuestras entrevistas, para dejar en evidencia el positivo impacto del sistema.

Concretamente, debemos programar reuniones de “difusión” con los usuarios del sistema y con los niveles gerenciales.

2.4.2 Análisis de factibilidad.

Otro aspecto importante de analizar en este momento, tiene relación con la factibilidad del proyecto; es decir, ya hemos capturado los requerimientos de los usuarios y hemos conocido los sistemas de información de la organización. Debemos entonces determinar qué grado de relación existe entre estos dos factores, es muy probable que la información manejada por los usuarios finales no provenga directamente del sistema de información operacional, si no que sea el resultado de cruces de información proveniente de planillas de cálculo e información del sistema, también es probable que algunos usuarios utilicen complejas macros para transformar la información proveniente desde el sistema

a información válida dentro del contexto actual del negocio. Esto último sucede frecuentemente en las PYMEs y se produce básicamente debido al distanciamiento del sistema de información con la realidad actual del negocio, generando abundante trabajo operativo.

Sin embargo debemos evitar generar la impresión de estar realizando algún tipo de auditoria ya que esto no es nuestro objetivo y generará irremediablemente una reacción defensiva de parte de los usuarios.

Muchos de estos problemas son solucionables. Debemos determinar y acordar caminos de solución efectivos en el corto plazo, de modo de asegurarnos de contar con fuentes de información consistentes con la naturaleza actual del negocio.

2.5 Definición de metas y objetivos.

Una vez recolectada toda la información resultante de las entrevistas, debemos tener una imagen clara de los aspectos globales de la empresa y los aspectos específicos del área en la que nos estamos abocando. A través de las entrevistas debemos haber identificado cuáles son las necesidades que tenemos que resolver y visualizar cuáles son los problemas a los que nos enfrentaremos.

Con toda esta información debemos definir formalmente el alcance de nuestro proyecto y cuáles serán nuestros objetivos, también es un buen momento para definir los plazos de nuestro proyecto, esto debemos formalizarlo a través de una carta Gantt que especifique plazos y holguras para cada una de las etapas del proyecto, es importante que haya un consenso con nuestro equipo de desarrollo en cuanto a los plazos, de manera de no caer en plazos demasiado optimistas o quizás poco realistas.

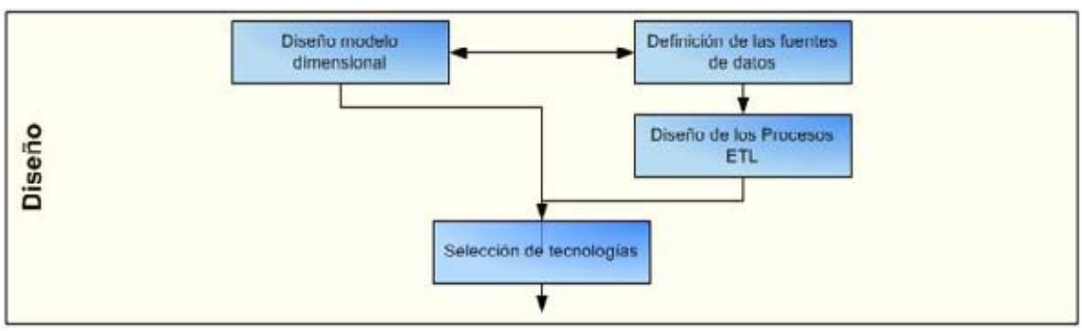
3. Diseño.

La etapa de diseño comprende la definición formal de la estructura lógica que involucra un sistema data warehouse, esto incluye los modelos de datos y a la definición lógica de todos aquellos procesos que alimentan y mantienen al sistema data warehouse.

3.1 Objetivos.

Al finalizar esta etapa debemos haber definido:

- EL modelo de datos que soportará el data warehouse (modelo de datos dimensional).
- Las fuentes de datos que alimentarán el data warehouse.
- El diseño de los procesos de carga de datos desde las fuentes al data warehouse, esto incluye.
    - Diseño del proceso de carga inicial.
    - Diseño del proceso de carga periódica.
    - Definición de la periodicidad de las cargas.



3.2 Diseño del modelo dimensional (estructura de datos).

Habitualmente en los diferentes sistemas operacionales encontramos la información organizada en grandes y complejos modelos de datos relacionales. Adicionalmente los usuarios manejan información en planillas de datos, archivos de texto, etc.

Los sistemas operacionales diseñados sobre modelos de datos relacionales son sistemas fuertemente orientados a soportar el día a día de la empresa y no diseñados para soportar la gestión. La generación de informes de gestión sobre estos sistemas generalmente es rígida y consume una gran cantidad de recursos.

La estructura lógica de un data warehouse debe ser entendible al usuario final, de manera que éste visualice fácilmente la relación entre los diferentes componentes del modelo.

Es por esto que para la construcción de un data warehouse se utiliza el modelo de datos dimensional cuya estructura es mucho más intuitiva, facilita la interacción con el usuario final y permite simplificar el diseño e implementación de las aplicaciones cliente [Kim+, 98].

El modelado dimensional es [Kim+, 02] una técnica de diseño lógico que busca organizar los datos en un marco estándar, que sea intuitivo y de acceso rápido. Por ejemplo, normalmente definimos un punto en el espacio por la intersección de sus coordenadas X, Y, Z. Si le asignamos valores a esas coordenadas, digamos X representa clientes, Y representa productos y Z representa el tiempo, luego consideramos la siguiente combinación X = clientes = Supermercados Y = productos = leche, Z = tiempo = enero-2004, obtendremos un punto en el espacio el cual podríamos definir como la cantidad de leche vendida.

A la estructura descrita anteriormente se le llama comúnmente en la literatura cubo. En la figura 4 podemos ver una representación gráfica del ejemplo anterior. A diferencia de un cubo geométrico, los cubos producto del modelado dimensional pueden tener de 1 a n ejes.

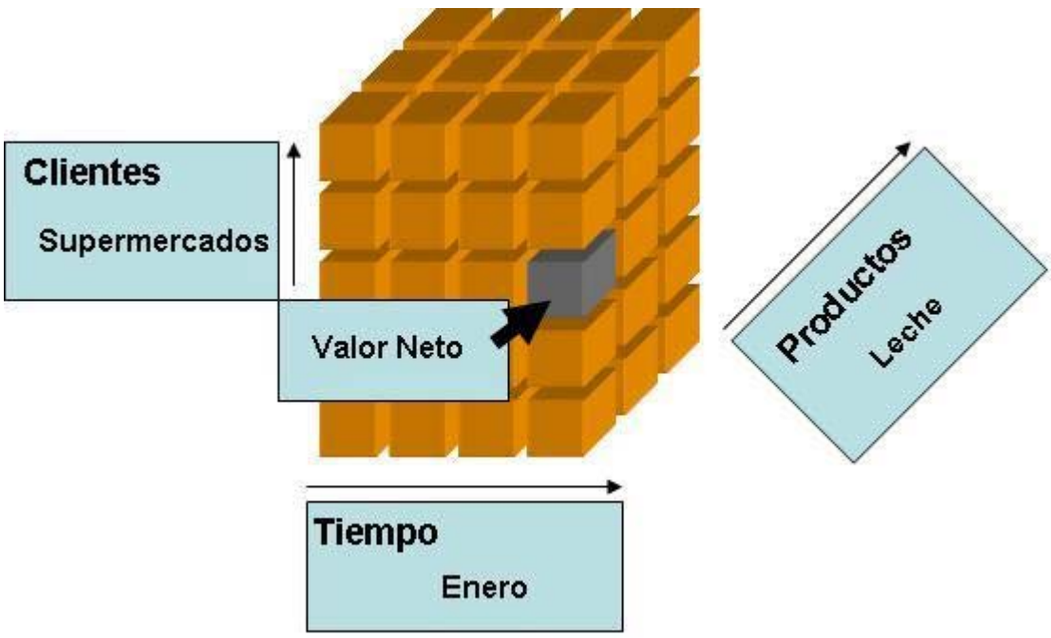


Figura 4: Ejemplo de Cubo en un modelo dimensional.

Hay que aclarar que la información requerida por un área de negocios o un modelo de datos relacional pueden equivaler a múltiples modelos dimensionales.

El modelar la información bajo este esquema supone una serie de ventajas a la hora de construir un data warehouse, algunas de estas ventajas son:

- El modelo dimensional define un marco estándar, que permite la construcción de aplicaciones finales flexibles y más fácilmente, esto porque los desarrolladores de herramientas pueden realizar fuertes supuestos acerca de la estructura de los datos.

- Este marco estándar soporta cambios inesperados en el comportamiento de los usuarios, esto gracias a la simetría del modelo, ya que podemos considerar cada dimensión como un punto de entrada simétrico en la tabla de hechos.
- El modelo es extensible, se pueden agregar o alterar las dimensiones, agregar nuevas métricas a la tabla de hechos, con la ventaja que las aplicaciones de usuario no tendrán que ser reprogramadas.

El diseño de un data warehouse implica un gran esfuerzo, tanto del equipo desarrollador del proyecto, como de parte de la empresa patrocinante. Dada esta realidad, pocas veces se inicia un proyecto de Data Warehousing que pretenda integrar todas las áreas del negocio en este tipo de sistemas, habitualmente estos proyectos se enfocan a áreas puntuales del negocio. Así, si tenemos la intención de atravesar toda la empresa con nuestro data warehouse, lo recomendable es dividir nuestro proyecto en sub-proyectos orientados a áreas específicas del negocio. También es posible que deseemos integrar sólo algunas áreas del negocio y dejar las otras para futuras oportunidades.

3.2.1 Arquitectura Data Warehouse.

Un data mart es “un subconjunto lógico de un data warehouse completo” [Kim+, 98]. Un data mart representa un proyecto de tamaño ejecutable. Un data warehouse se genera a partir de la unión de múltiples data marts sobre una lógica común.

Comúnmente consideramos a un data mart como una versión restringida, a un proceso o área de negocio específico, de un data warehouse.

Cuando comenzamos la construcción de múltiples data marts debemos tener especial cuidado en no construirlos de forma aislada, por que de otra

manera al final del proyecto tendremos un conjunto de islas con información pero que no se comunican entre ellas.

Nuestro proyecto se suma al enfoque presentado por Kimball [Kim+, 98] que describe una arquitectura de comunicación entre los diferentes data marts basada en ***Dimensiones conformadas y definición conformada de hechos (métricas)***.

Este enfoque plantea que en la fase de diseño debemos definir un conjunto de dimensiones conformadas para todos los data marts, así como estandarizar las definiciones de las métricas de las diferentes tablas de hecho.

El conjunto de estándares y definiciones resultantes será entonces lo que Kimball llama **Arquitectura de Bus (Data Warehouse Bus Architecture)** [Kim+, 98]**.**

Una dimensión conformada es una dimensión que significa lo mismo para todas las posibles tablas de hecho en las que participa. Por ejemplo, una dimensión conformada cliente es una tabla de clientes con una clave primaria independiente y un conjunto de atributos bien definidos que describen al cliente, esta dimensión debe contener la mejor y más completa información sobre cada cliente que podamos obtener del sistema operacional. De esta manera publicaremos esta dimensión para el uso tanto del sistema operacional como del data warehouse.

Al igual que una dimensión conformada, una métrica estandarizada corresponde a un concepto uniforme a través de todo el sistema.

.Entre las ventajas de utilizar dimensiones conformadas podemos señalar:

- Una sola tabla dimensión puede ser usada contra múltiples tablas de hechos en la misma base de datos.
- Los contenidos son consistentes donde sea que se utilice la dimensión.

- Existe una interpretación consistente de los atributos de la dimensión dentro y a través de múltiples data marts.

3.2.2 Técnicas de Modelado Dimensional.

Cada modelo dimensional está compuesto de una tabla con una clave primaria compuesta llamada *"tabla de hechos"*, y un conjunto de tablas más pequeñas llamadas *"tablas dimensiones"*. Cada tabla dimensión tiene una clave primaria simple que corresponde exactamente a uno de los componentes de la clave compuesta en la tabla de hechos (ver figura 5). Esta estructura similar a una estrella, es comúnmente llamada *"star join"*.

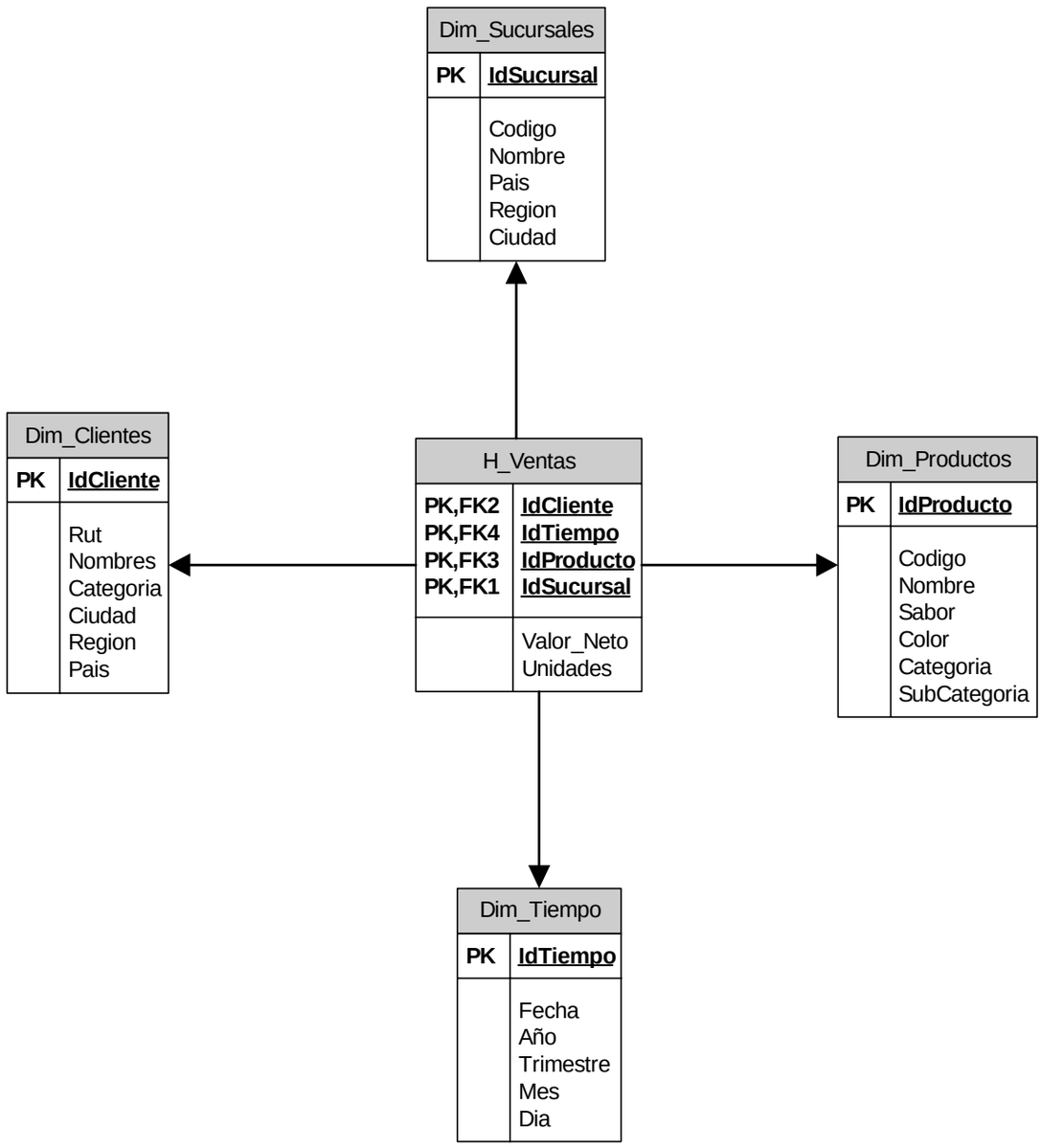


Figura 5: Ejemplo de esquema estrella.

La tabla de hechos, debido a que tiene una clave primaria compuesta, formada a partir de dos o más claves foráneas siempre expresa una relación de muchos a muchos.

Las tablas de hechos, también, deben incluir uno o más *"hechos numéricos"* que son el resultado de la combinación de las claves que definen cada registro, en la figura 4 los hechos son valor neto y unidades.

Los hechos, además de numéricos, deben ser aditivos, esto es, debido a que en muy pocas oportunidades consultaremos por un solo registro de la tabla de hechos, lo común es traer cientos o quizás miles de registros a la vez, y en estos casos la única cosa útil por hacer es sumarlos [Kim+,98].

Las tablas de dimensión, por el contrario, comúnmente contienen información descriptiva textual. Los atributos de las dimensiones son el origen de los filtros más interesantes en las consultas al data warehouse.

La tarea de trasformar los modelos relacionales en modelos dimensiónales consiste básicamente en tres pasos:

1. Dividir el modelo relacional en sub modelos que abarquen los diferentes procesos de negocio, para modelar cada uno de estos casos separadamente.
2. Seleccionar todas aquellas relaciones muchos a muchos existentes en el modelo relacional, que sean numéricas y aditivas, y designarlas como tablas de hechos.
3. Des normalizar todas las tablas restantes, hasta transformarlas en tablas planas con una clave primaria simple, que se conectará directamente con la tabla de hechos, éstas serán las dimensiones. En caso de que una misma dimensión se conecte a más de una tabla de hechos, representamos esta dimensión en ambos esquemas, y nos referiremos a ella como conformada. [Kim+,98].

Una de las principales ventajas que presenta el modelado dimensional tiene relación con que existe un conjunto de consideraciones formales para manejar situaciones comunes del modelado. Entre las situaciones más comunes e importantes, se tienen [Kim+,98]:

Dimensiones que cambian lentamente (Slowly changing dimensions, SCD), una dimensión aparentemente constante como “clientes” o “productos” en realidad evoluciona lentamente y asincrónicamente. La situación considerada aquí es que la clave de producción (de clientes) no cambia, sin embargo sí cambia alguno de los atributos textuales (Dirección etc.). Para este tipo de situaciones existen tres aproximaciones de solución que dependerán principalmente de la naturaleza de la dimensión y de los requerimientos:

- Sobrescribir el registro en la dimensión con los nuevos valores, y por consiguiente perder el histórico (del atributo).
- Crear un nuevo registro usando un nuevo valor para la clave sustituta.
- Crear un nuevo campo en la dimensión para almacenar el valor antiguo.

Esquemas de productos heterogéneos, este es el caso, por ejemplo, de los servicios financieros, debido a la naturaleza heterogénea de sus productos. Un banco puede ofrecer, cuentas corrientes, cuentas de ahorro, créditos hipotecarios, tarjetas de crédito etc. En este ambiente hay usualmente dos perspectivas de las actividades de los diferentes productos que son difíciles de representar en una sola tabla de hechos. La primera perspectiva es la perspectiva global en la cual necesitamos navegar por todas las cuentas de todos los tipos al mismo tiempo, existiendo un conjunto de hechos comunes. La segunda perspectiva desea ver los diferentes productos individualmente, cada uno de los cuales tendrá un conjunto de hechos especiales. Estos hechos

especiales no pueden ser agregados a la tabla de hechos principal, debido a que si lo hiciéramos deberíamos añadir hechos especiales por cada línea de productos, terminando con una cantidad inmanejable de hechos en la tabla principal. La solución a este problema consiste en crear una tabla de hechos personalizada para incluir aquello que está limitado exclusivamente a una línea de productos (ver figura 6) y extender tanto la dimensión productos, como la tabla de hechos principal para describir todos aquellos atributos y hechos especiales que tienen sentido para el determinado producto.

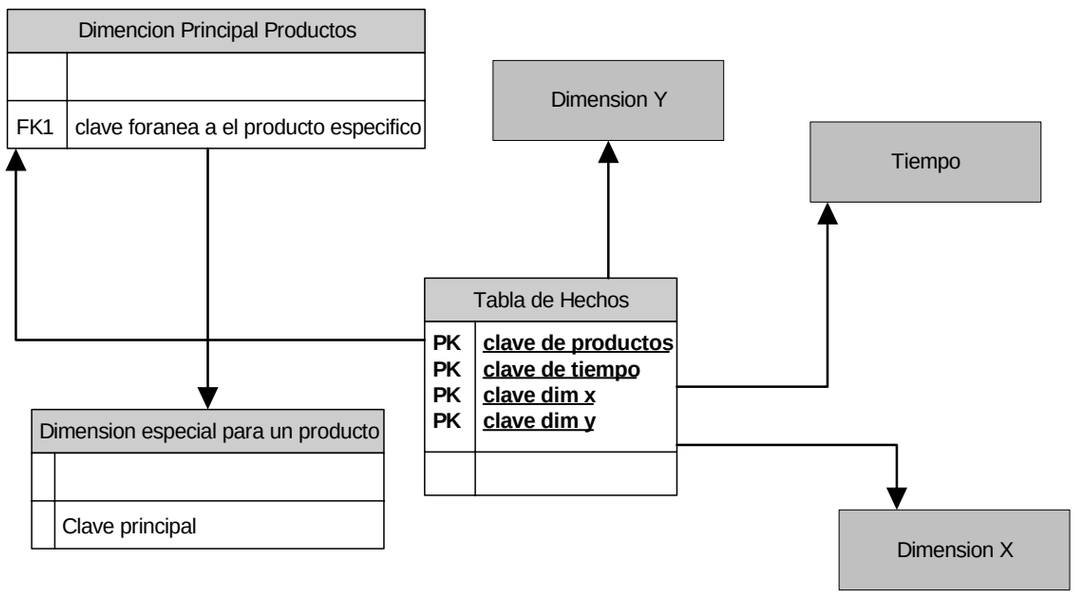


Figura 6: Productos heterogéneos.

Esquema de fotografías versus esquema transacciones, Virtualmente todos los data marts necesitarán dos versiones separadas de los datos: una versión transaccional y una versión que sea una fotografía periódica. El esquema transaccional representará el nivel más básico de nuestro sistema operacional, es decir, las transacciones individuales. Un registro en la tabla de hechos para una transacción individual generalmente contendrá un sólo hecho, el cual será el valor de la transacción. El esquema de fotografías almacenará información

periódica y el sistema entonces debe crear los registros necesarios periódicamente en una nueva tabla de hechos, calculando los valores del periodo a partir de la información de las transacciones.

Tablas de hechos "sin hechos", existen situaciones en las cuales podemos realizar un diseño correcto y sin embargo no encontrar ningún hecho que agregar a la tabla de hechos. Estas tablas de hechos son generalmente útiles a la hora de detectar eventos y coberturas. Un buen ejemplo de tabla de hechos sin hechos y que es útil para la detección de eventos puede ser un sistema de control de asistencia de alumnos, que registra cada asistencia de un alumno por día, en este caso no existirán hechos numéricos, simplemente cada registro en la tabla de hechos representará una asistencia. Estas tablas de hechos también permiten determinar cobertura, por ejemplo, de una promoción sobre un determinado producto, en este caso una clásica tabla de hechos nos permitiría con certeza saber cuánto vendimos de un producto bajo una promoción dada, pero, sin embargo, no podríamos contestar preguntas sobre situaciones que no sucedieron, como qué productos que estaban en promoción no se vendieron. Una tabla de hechos para la cobertura soluciona este problema, pues registrará cada producto que esté en promoción en un determinado tiempo.

Además de las situaciones comunes señaladas anteriormente existen un conjunto de buenas prácticas y recomendaciones que es necesario tener en cuenta cada vez que se utiliza el modelo dimensional [Kim, 01].

- **No colocar atributos de tipo texto en las tablas de hecho si se piensa utilizarlos como punto de entrada de una clave o para las funciones de agrupación.** Crear un modelo dimensional básicamente comienza por identificar las medidas numéricas entregadas por el sistema de origen, éstas van a la tabla de hechos. Luego identificar los

atributos textuales descriptivos dentro del contexto de las medidas, esto va a las dimensiones. Finalmente revisar caso a caso la naturaleza de todos los códigos e ítems seudo numéricos sobrantes, ubicándolos en la tabla de hechos si son más como una medida, o en las dimensiones si son más como la descripción textual de algo. Pero por ningún motivo debe dejarse texto puro en la tabla de hechos, especialmente campos de comentarios, sino tomar estos atributos y ubicarlos en las dimensiones.

- **No limitar el largo de los atributos textuales en una dimensión, para ahorrar espacio.** Si consideramos que virtualmente en todos los data warehouse, las dimensiones son geométricamente más pequeñas que la tabla de hechos, entonces ¿importa realmente tener una dimensión de 100MB cuando la tabla de hechos es 100 veces más grande? Para diseñar un data warehouse de “fácil uso” debemos poner a disposición toda la información descriptiva posible en cada dimensión, recordando siempre que los atributos textuales de las dimensiones son el punto de entrada del usuario.
- **No dividir las jerarquías y los niveles jerárquicos en múltiples dimensiones.** Una jerarquía es una serie en cascada de relaciones muchos-a-uno. Muchos productos pertenecen a una sola rama, muchas ramas pertenecen a una sola categoría y así. Si las dimensiones están expresadas al grano más fino (ej. Producto), entonces, todos los niveles más altos de la jerarquía pueden ser expresados como valores únicos en un registro Producto. Los usuarios entienden las jerarquías y nuestro trabajo es presentarlas de la manera más natural y eficiente. Una jerarquía pertenece a una sola tabla de dimensión física. No es recomendable dividir una jerarquía, generando un conjunto progresivo de pequeñas sub-dimensiones (conocidas como snowflakes).

- **Retardar enfrentarse con las dimensiones que cambian lentamente (SCD).** Demasiados data warehouse son diseñados para regularmente sobre escribir las dimensiones más importantes, como Productos, Clientes etc. desde las fuentes de origen correspondientes. Esto va contra el juramento básico de un data warehouse: El data warehouse representará la historia con precisión, incluso si las correspondientes fuentes de datos no lo hacen. Las SCD son un elemento esencial del diseño de un data warehouse.
- **No utilizar claves "inteligentes" para enlazar una tabla dimensión a una tabla de hechos.** Cuando diseñamos la clave primaria de la tabla dimensión, ésta necesariamente debe conectarse a la tabla de hechos. Es contraproducente declarar una clave primaria compuesta en la tabla de dimensión y luego utilizar ésta como la base del enlace físico con la tabla de hechos. Se debe remplazar las claves "inteligentes" por claves numéricas secuenciales simples.
- **No agregar dimensiones a una tabla de hechos antes de haber definido el grano.** Todo diseño dimensional debe comenzar por las medidas numéricas. Segundo, especificar el grano y el significado de esas medidas. Tercero, rodear estas medidas con dimensiones válidas dentro del contexto del grano. Mantenerse dentro del contexto del grano es fundamental para el diseño del modelo dimensional.
- **No establecer que un determinado modelo dimensional corresponde a un determinado reporte.** Un modelo dimensional no tiene nada que ver con un reporte específico. Un modelo dimensional es el modelo de un proceso de mediciones. Una medida numérica es una sólida realidad física. Las medidas conforman la base de la tabla de hechos. Las dimensiones apropiadas para una tabla de hechos son el

contexto físico que describe las circunstancias de la medida. Un modelo dimensional es independiente de cómo el usuario final escoja definir un reporte.

- **No mezclar granos diferentes en la misma tabla de hechos.** Un error serio en el diseño dimensional es agregar hechos "extras" a la tabla de hechos, como registros que describan los totales de un periodo específico de tiempo. Aunque estos registros pueden parecer simplificar algunas aplicaciones, pueden causar estragos en los resultados de agregaciones automáticas que contarán estos registros de nivel más avanzado, produciendo resultados incorrectos.
- **No dejar el nivel más bajo de datos (atómico) en modelos E/R.** Este nivel debe ser el cimiento físico del modelo dimensional.
- **No evitar las agregaciones en nuevas tablas de hechos, ni reducir las dimensiones cuando se enfrenten problemas de rendimiento, ni resolver los problemas de rendimiento agregando más hardware.** Las agregaciones son la mejor alternativa costo-efectiva para mejorar los tiempos de las consultas. La adición de hardware paralelo, de costo elevado, debe ser parte de un plan balanceado que incluya también la creación de agregaciones, creación de múltiples índices, aumento real del tamaño de la memoria, y aumento de la velocidad de la CPU.
- **Conformar los hechos a través de todas las tablas de hechos.** Si tenemos una medida llamada *Ingresos* en dos o más data marts alimentados de fuentes distintas, entonces debemos tener especial cuidado en asegurarnos que la definición técnica de ambas medidas sea exactamente la misma. Deseamos que sea posible operar sobre ambas medidas libremente en nuestras aplicaciones.

- **Conformar las dimensiones a través de todas las tablas de hechos**. Este es el punto más importante dentro de nuestro conjunto de técnicas de modelado. Si dos tablas de hechos tienen una misma dimensión, entonces estas dimensiones deben ser idénticas, o se debe ser muy cuidadoso si se escoge un subconjunto de una o de otra, asegurando que no se generen valores inválidos. La existencia de dimensiones conformadas a través de las tablas de hechos nos permite navegar a través de diferentes fuentes de datos indistintamente, ya que las entradas y los encabezados de las columnas tendrán igual significado. Las dimensiones conformadas son la clave para construir data warehouse distribuidos que permitan la aparición de inesperados nuevos datos, y que permitan además la integración de tecnologías heterogéneas.

3.2.3 Granularidad de la información.

La definición del grano del modelo debe ser realizada al inicio del modelado dimensional, una vez que las fuentes de datos están claras.

Cuando hablamos de granularidad, nos referimos al nivel de detalle de la información que extraeremos. Por ejemplo, para el caso de las facturas de ventas, podemos extraer, (desde el nivel más grueso al nivel más fino) el total del día, el total de la factura, o el total por ítem de la factura.

En general en los niveles más altos de gestión difícilmente estarán interesados en el detalle de alguna factura en específico; sin embargo, debemos escoger el nivel atómico de la información disponible, es decir, definir el grano al nivel más fino que soporte la lógica del negocio y nuestras fuentes de datos; esto es debido que a medida que el grano es más grueso menor es el

número de atributos que tienen validez. Para el caso de las facturas es claro que si extraemos solamente los totales de cada factura, perderemos la referencia a los productos que incluye la factura; más aún, si tomamos los totales diarios perdemos la referencia al cliente de cada factura. Esto merma considerablemente las capacidades de nuestro modelo en cuanto principalmente a flexibilidad y escalabilidad.

Mientras más fino sea el grano que escojamos, mayor será la información que podemos extraer con seguridad de las fuentes de datos, es decir, mayor será el número de dimensiones que tendrán lógica dentro del contexto del grano, es por esto que los datos atómicos son perfectos a la hora de generar un modelo dimensional.

Si esto no es posible tarde o temprano tendremos que enfrentar diversas dificultades, asociadas a lo señalado anteriormente.

Al definir el grano a nivel atómico podremos:

- Navegar a través de los diferentes niveles de granularidad. Para que esto sea posible debemos generar agregaciones.
- Obtener una relación 1-1 entre las métricas de las tablas de hechos y los datos del sistema de origen.
- Tener una lógica común entre las diferentes tablas de hechos.

3.3 Diseño Procesos ETL (Extracción, Transformación y Carga de datos).

Hemos realizado ya un análisis exhaustivo de nuestras fuentes de datos, y hemos definido el conjunto de dimensiones y tablas de hecho que formarán nuestro data warehouse, es entonces el momento de estructurar formalmente los procesos que nos permitirán poblar nuestro data warehouse desde las fuentes de datos.

Definiremos, entonces, procesos que nos permitan mapear los datos desde los sistemas fuentes correspondientes hacia el data warehouse, estos procesos son llamados procesos ETL.

Debemos, en esta etapa, definir dos procesos ETL, el primero definirá el proceso de extracción, transformación y carga inicial, es decir, el proceso con el cual poblaremos inicialmente, desde las fuentes de datos, nuestro sistema, este proceso se ejecutará una sola vez.

El segundo proceso a definir será el proceso que se encargará de agregar periódicamente la información nueva desde las fuentes de datos a nuestro data warehouse, ambos procesos son críticos para el éxito de nuestro proyecto, y generalmente consumen el mayor porcentaje de tiempo del proyecto.

Definimos tres etapas en cada proceso ETL:

- Extracción: Esta etapa consiste en recolectar los datos de las diferentes fuentes operacionales.

- Transformación: Una vez extraídos los datos, en esta etapa realizamos todas las tareas relacionadas con la calidad de los datos, esto puede consistir en eliminar datos incorrectos, trasformar datos de manera que sean válidos para nuestro data warehouse, limpiar datos etc.
- Carga: Etapa final del proceso ETL, en la cual ya contamos con datos consistentes y listos para la carga al data warehouse.

Para realizar este proceso debemos generalmente definir un área de datos intermedia “Data Staging Area (DSA)” [Kim, 1998] a la que llevaremos los datos resultantes de la extracción, para realizar un conjunto de transformaciones previamente definidas y finalmente cargarlos en nuestro sistema.

La figura 7 describe gráficamente las etapas lógicas y físicas de un proceso ETL.

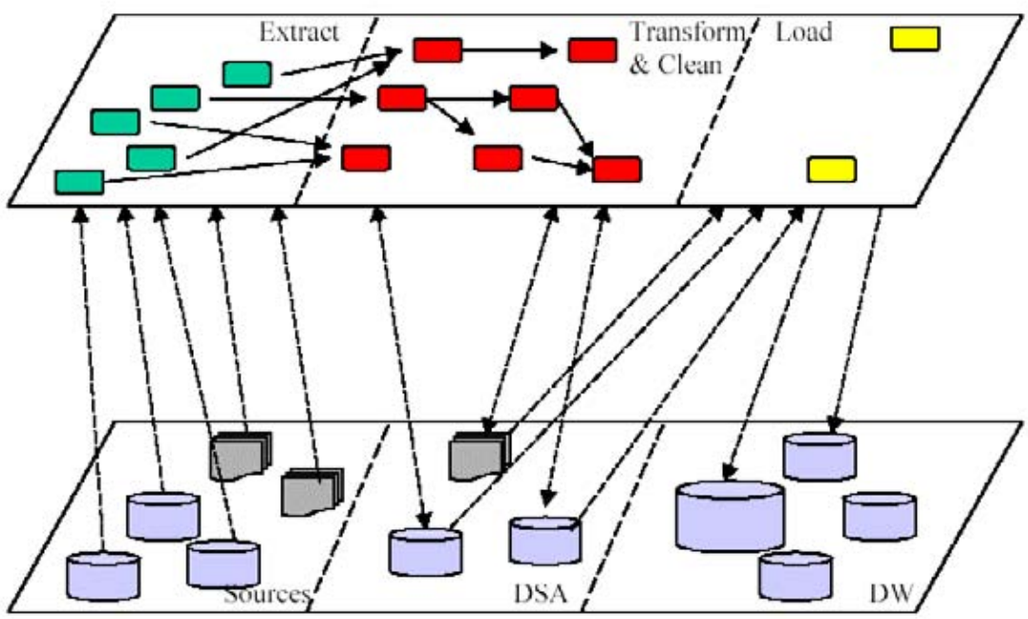


Figura 7: Proceso ETL [Vas+, 02].

Las características técnicas de esta área de datos intermedia (DSA) dependerán directamente de la naturaleza de los sistemas, por ejemplo:

- Si tanto los sistemas operacionales de donde extraeremos los datos, así como el sistema que estamos desarrollando, dependen tecnológicamente del mismo proveedor y existen o se pueden definir enlaces de alto rendimiento entre ambos sistemas, es posible realizar el proceso completo ETL de manera directa ente ambos sistemas, prescindiendo de la creación de la DSA.

- Si considerando los procesos ETL como procesos secuenciales que generalmente consisten en ordenar, seguido del paso secuencial de los datos a través de una o dos tablas, notaremos que es innecesario la utilización de una DBMS relacional para el área de datos intermedia y dada la simpleza de este paradigma de procesamiento, bastaría con un procesamiento secuencial de archivos planos.

- Si, por otra parte, consideramos que si bien el proceso ETL es un proceso secuencial relativamente simple desde el punto de vista lógico, existen una serie de ventajas al utilizar un DBMS relacional para la zona de datos intermedia, entre éstas: conexiones y consultas son de más alto nivel, integridad referencial, persistencia etc.

Para asegurar la integridad referencial en nuestro data warehouse debemos comenzar por procesar las dimensiones y luego las tablas de hechos, debemos revisar que nuestras dimensiones sean consistentes con nuestras fuentes de datos y, si no es así, debemos realizar las operaciones necesarias para conservar la consistencia. La manera más simple de realizar estas tareas es a través del procesamiento secuencial de los datos; esto es, recorrer tanto el

sistema origen como el sistema de destino y comparar uno a uno en busca de diferencias. Para esto seleccionamos y ordenamos el conjunto de datos de los sistemas fuente utilizando alguna clave de producción y lo mismo hacemos con cada dimensión (sólo que en este caso la clave es sólo un atributo más) y comenzamos a comparar entre ambos sistemas realizando las acciones pertinentes (SCD).

La siguiente tarea corresponde al limpiado y transformación de los datos; esto es, por ejemplo, la corrección ortográfica de los atributos o la verificación de que determinado atributo pertenece a una lista de valores válidos para ese atributo. Otro ejemplo común es realizar divisiones de algunos atributos para almacenarlos en múltiples campos de la dimensión, por ejemplo en el sistema fuente podríamos encontrar el atributo “dirección” en el cual se especifique la dirección de un cliente, pero en nuestra dimensión dividimos ese atributo en múltiples atributos (Departamento, calle, número, etc.).

Una vez procesadas las dimensiones se procesarán las tablas de hechos, éstas las encontraremos en los sistemas operacionales con sus claves de producción y no con las claves auto numéricas que nosotros hemos utilizado; debemos entonces realizar el reemplazo de las claves de producción por nuestras claves numéricas.

Si existen resúmenes debemos realizar las tareas necesarias de refresco, para regenerar nuevos resúmenes válidos.

3.3.1 ETL basados en Herramientas vs. ETL de código escritos manualmente.

Otra decisión a tomar en esta etapa tiene que ver con las tecnologías que se utilizarán para implementar estos procesos ETL, hoy en día existe un conjunto sofisticado de herramientas comerciales diseñadas para realizar estos procesos de manera gráfica y con funciones pre-programadas que ayudan a simplificar nuestro trabajo. Sin embargo, son herramientas de alto costo y en el mundo Open Source existen herramientas que permiten realizar algunas operaciones sobre determinadas fuentes de datos pero están más orientadas a procesos de migración de datos que a procesos ETL.

La decisión que tomemos debemos basarla en los costos versus los beneficios que presenta una alternativa sobre la otra. A continuación se señalan las principales ventajas de las herramientas ETL, y las ventajas de las herramientas desarrolladas a medida del proyecto [Nis, 2003].

ETL basados en Herramientas:

- Simples, más rápidas y bajo costo de desarrollo, el costo de una herramienta se compensará en proyectos grandes y sofisticados.
- Muchas herramientas ETL incluyen repositorios de metadata que pueden sincronizar metadata desde los sistemas fuentes, bases de datos o herramientas de BI (Inteligencia de negocios)
- La mayoría de las herramientas ETL fuerzan una consistencia en la metodología de manejar la metadata, forzando a los desarrolladores a seguirla.
- Diagramas de flujo y documentación pueden ser generados automáticamente, a partir de la metadata.

- Cuentan con conectores para diferentes fuentes de datos, lo cual es una ventaja si se está trabajando con fuentes de datos heterogéneas.
- La mayoría de las herramientas ETL entregan excelente rendimiento con grandes volúmenes de datos.

ETL de código escrito manualmente:

- Existen herramientas para verificar código automáticamente, por ejemplo junit [URL 2] es una herramienta para realizar pruebas automáticas sobre código java (nunit .net framework). La verificación automática de código mejora significativamente nuestra producción.
- La técnica de Orientación a Objeto ayuda a realizar todas las transformaciones consistentes, con reportes de errores, validaciones y actualizaciones de metadata y es posible que las herramientas ETL no ofrezcan este nivel de control sobre los procesos.
- Se puede manejar de manera directa la metadata, lo que nos permite realizar modificaciones de manera más flexible.
- Un análisis previo a un sistema ETL rápidamente nos enfocará al procesamiento de archivos, no en procedimientos almacenados, el procesamiento basado en archivos es más simple de codificar, más fácil de probar, y más entendible.
- Una herramienta ETL nos limita a las habilidades de un proveedor determinado, y al lenguaje que nos provea, en cambio una herramienta desarrollada internamente puede ser desarrollada en lenguajes comunes y bien conocidos.

- Proveen de flexibilidad ilimitada, se puede hacer literalmente lo que uno quiera.

A partir de las características señaladas podemos establecer una tabla comparativa (ver tabla 4).

	ETL de código escrito manualmente	ETL basados en Herramientas
Costo de licenciamiento	Alto	Nulo
Rapidez de desarrollo	Bajo	Proporcional al proyecto
Flexibilidad	Dependiente de la herramienta	Ilimitada
Rendimiento	Alto	Medio

Tabla 4: Comparación ETL codificadas y herramientas comerciales ETL

Consideramos que la decisión entre una herramienta comercial y una herramienta desarrollada a medida dependerá fundamentalmente del tamaño, cantidad de recursos y complejidad del proyecto, debiendo optarse por aquella alternativa que se acerque más a nuestras necesidades.

3.3.2 ETL como Workflow.

Dadas las características de los procesos ETL, adherimos a la definición de este proceso como un proceso Workflow [Bon+,01], [Bou+,99]. A partir de esto, se define el proceso ETL como un flujo de tareas que se inicia con la extracción de los datos y que termina con la carga de los datos definitivos en el data warehouse. El definir el proceso ETL como Workflow presenta una serie de ventajas, muchas de ellas heredadas del concepto de workflow:

- Mejorar el control sobre las excepciones o casos especiales. Por ejemplo si existe una etapa que se encargue de analizar direcciones de clientes de manera que transforma el dato dirección de la fuente origen en múltiples atributos en una dimensión (calle, sector, numero, etc.) de forma automática, podemos también, definir una etapa para los casos

especiales que no podamos procesar automáticamente, de manera de asignárselos a un usuario específico, que realice esta tarea manualmente.

- Los procesos son más flexibles a los cambios en el tiempo. Un workflow tiene la característica de ser modificable y escalable en el tiempo
- Permitir calendarización de las tareas, esto es, permitir la programación en el tiempo de las diferentes etapas del proceso.
- El proceso workflow genera información de gestión que estará disponible para los administradores del sistema.
- Permitir la definición a nivel de metadata del proceso ETL.
- El proceso será más comprensible al estar definido dentro de un marco teórico
- Permite la integración de diferentes herramientas, esto es, podemos utilizar las ventajas de una u otra herramienta dependiendo la etapa en la que nos encontremos.

Las herramientas comerciales de ETL más sofisticadas consideran muchas de estas características, permitiendo a los usuarios definir dinámicamente sus procesos ETL, por ejemplo, el “Data Transformacion Service” (Servicio de Transformación de Datos, ver figura 8) de Microsoft permite automatizar los procesos de extracción de datos desde fuentes heterogéneas, también, permite automatizar las transformaciones, permitiendo al usuario programar estas tareas en el tiempo, y como Microsoft señala: *“Los desarrolladores pueden acceder y manipular las operaciones de la extracción de datos multi-etapa que les permiten interactuar en múltiples puntos durante el proceso de transformación”*. [URL 3] Este enfoque “multi-etapa”, representa un workflow.

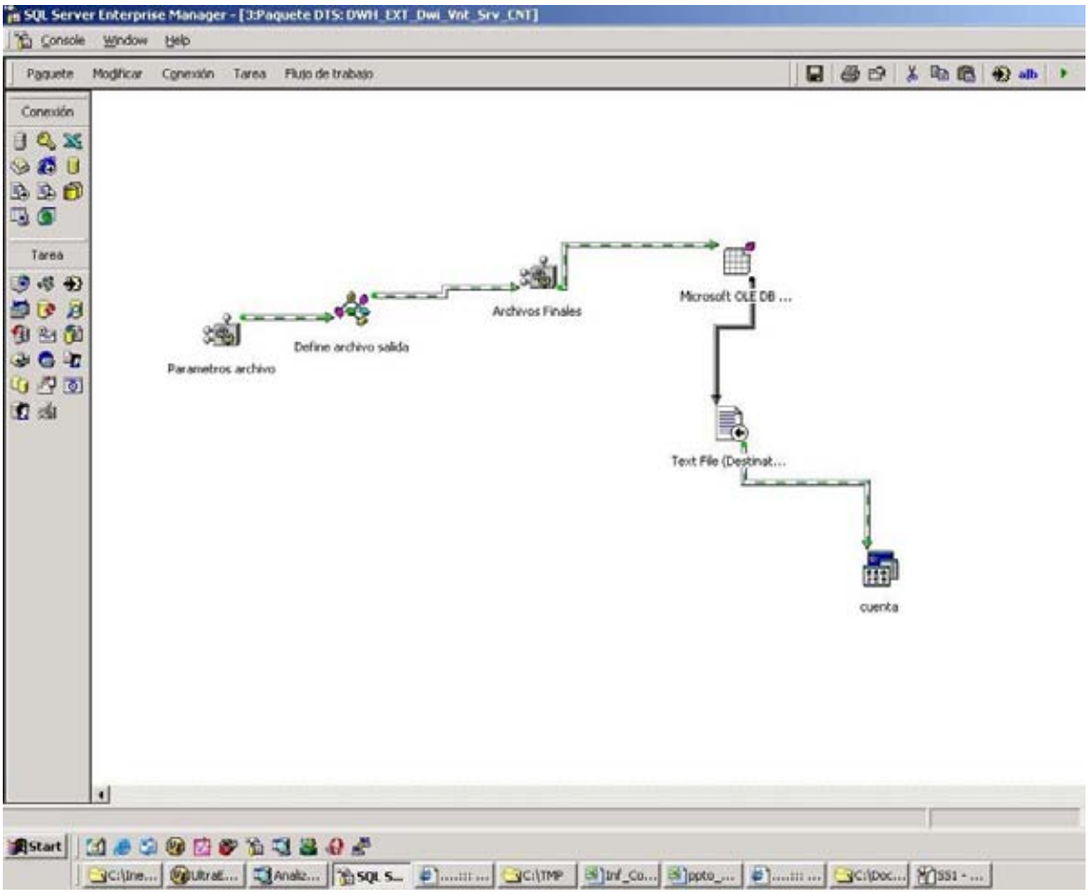


Figura 8: Data Transformation System.

100% automático definido por los desarrolladores durante la implementación, en el cual, en cada una de las etapas se realizan diversas tareas con el objetivo de conseguir los datos deseados para el data warehouse.

Sin embargo, la posibilidad de integrar de manera nativa herramientas de workflow con un conjunto de herramientas de extracción, carga y transformación es aún más flexible.

Si bien, en teoría, debiéramos esperar que todos nuestros procesos ETL fueran 100% automáticos, en la práctica y dado el enfoque de este trabajo (el cual está orientado más bien a empresas en las cuales la integración de sus fuentes de información no es completa y, más aún, es posible que existan algunas islas de información externas a los sistemas), notaremos que es práctico y a veces fundamental que nuestros procesos ETL interactúen con

usuarios específicos y para esto nada mejor que enmarcar esta interacción dentro de un workflow, que nos permitirá monitorear esta interacción.

La gran mayoría de los proveedores de herramientas ETL está enfocado a las grandes empresas y por ende, grandes procesos ETL, que en su mayor parte contarán con sistemas fuentes sólidos y operacionalmente integrados, los cuales darán un punto de partida firme a todo el proceso ETL. En base a esto, estas herramientas encapsulan decenas de utilidades comunes en este tipo de procesos, permitiendo a los desarrolladores definir sus flujos visualmente y a partir de parámetros, reduciendo la generación de código drásticamente.

3.3.3 Periodicidad de las cargas.

La ejecución periódica de los procesos ETL permite mantener el sistema data warehouse actualizado; sin embargo, este no es un proceso continuo, los sistemas fuente están en constante cambio, cambios que se propagan al data warehouse, sólo con la ejecución de un proceso ETL. En rigor, el data warehouse nunca contiene la información actualizada al nivel del sistema operacional.

Debemos asegurarnos que la información que contenga el data warehouse sea oportuna. Esto dependerá directamente de la periodicidad con la que ejecutemos los procesos de carga. El factor básico que dicta la periodicidad es el negocio, debemos responder, ¿Qué grado de actualización de información requieren las consultas? Si un usuario necesita consultar reportes de ventas, por ejemplo, del día, es fundamental proveerle de la información de día; si, por el contrario, solicita informes en base a la información de la semana pasada, podemos programar cargas semanales.

Es probable que debamos realizar algunos procesos de carga con una periodicidad menor que otros, y esto no será problema, siempre y cuando los usuarios cuenten con la información que necesitan en el momento oportuno.

3.3.4 Especificación de los ETL.

El cómo deben formalizarse los procesos ETL, es un tema poco abordado en la literatura, no existe estándar alguno que nos indique cómo debe ser esta especificación; sin embargo, dada la naturaleza crítica de los procesos ETL de refresco, sobre los cuales recae la responsabilidad de transportar los datos correctos al data warehouse a lo largo del tiempo, debemos realizar una especificación que nos permita definir de manera completa y simple cada uno de los procesos ETL que se desarrollen.

Una representación gráfica, como la sugerida en [Vas+,02] satisface las necesidades planteadas anteriormente, en ese documento los autores proponen una arquitectura gráfica para representar los diferentes escenarios ETL.

Para clarificar las posibilidades de esta representación grafica, a continuación presentamos un ejemplo. Primero describiremos un escenario global que representaremos en un “gráfico general”, y posteriormente tomaremos una sección específica del problema para representarlo á través de un “gráfico de arquitectura”. La notación utilizada para el gráfico general se presenta en la figura 9.

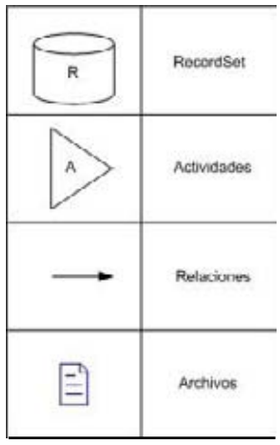


Figura 9: Notación gráfico general

El ejemplo considera dos bases de datos de origen S1, S2, un data warehouse central DW, y un área de datos intermedia (DSA) donde se producirán todas las transformaciones necesarias. El proceso considera la propagación de los datos desde una tabla PARTSUPP (PKEY, DATE, QTY, COST) del sistema S1, y de otra tabla PARTSUPP (PKEY,QTY,COST) del sistema S2, hacia el data warehouse. La tabla DW.PARTSUPP (PKEY, SUPPKEY, DATE, QTY, COST) almacena información de la cantidad disponible (QTY) y el costo de una parte (PKEY) por proveedor (SUPPKEY). Para este ejemplo los sistemas S1 y S2 corresponden a los dos proveedores de los datos para el data warehouse. Asumimos que el proveedor S1 es americano y el proveedor S2 es europeo, por lo tanto los datos del sistema S2 deberán ser convertidos a formato americano. Una vez poblada la tabla PARTSUPP se generarán resúmenes para poblar dos data marts V1 y V2. Todos los atributos con excepción de la fecha (DATE) son instancias de tipo Integer. V1 almacenará el costo mínimo por cada parte por día y V2 almacenará el costo promedio mensual de cada parte. Los atributos de fecha son del tipo US_DATE para S1 y EU_DATE para S2.

1. Primero transferimos las tablas S1.PARTSUPP y S2.PARTSUPP a DS.PS_NEW1 y DS.PS_NEW2 en el DSA, utilizando una conexión OLEDB.
2. En el DSA mantenemos copias locales de las tablas recientemente transferidas, así como copias de las penúltimas tablas trasferidas en DS.PS_OLD1, DS.PS_OLD2, de modo que controlando las diferencias entre la última y la penúltima, reconoceremos los nuevos registros agregados. Estos nuevos registros los almacenaremos en DS.PS1 y DS.PS2 respectivamente.
3. Luego se inician dos flujos, uno para los registros de DS.PS1 y otro para los de DS.PS2, a través de las actividades Add_SPK1 y Add_SPK2 y con el objetivo de reconocer de qué sistema proviene cada registro, agregamos el atributo SUPPKEY, asignando los valores 1 y 2 respectivamente, dependiendo de la fuente.
4. En ambos flujos creamos una clave de reemplazo para las claves primarias, ya sabemos que no es aconsejable utilizar las claves de producción como claves primarias dentro de un data warehouse. Para realizar esto utilizamos una tabla que llamamos LOOKUP_PS (PRODKEY, SOURCE, SKEY), esta tabla nos ayudará a mapear las claves de producción. Estas tareas son realizadas por las actividades SK1 y SK2.
5. Para los datos provenientes de S1, debemos aplicar una función de conversión para el atributo costo de manera de convertir sus valores de euros a dólares, de igual manera debemos convertir el atributo fecha desde el formato europeo al formato americano.
6. Paralelamente, para los datos provenientes de S2, (a) verificamos por los valores de costo no nulos, mediante la actividad NotNULL, (b) mediante

la actividad AddDate generamos el atributo date utilizando la fecha del sistema, (c) seleccionamos sólo aquellos registros para los cuales el atributo QTY es mayor a 0, en la actividad CheckQTY.

7. Ambos flujos son consolidados a través de la actividad U en la tabla DW.PARTSUPP.
8. Para poblar el data mart V1 simplemente agrupamos (Group-by) por día y PKEY, calculando el valor mínimo por parte, (actividad Aggregate1).
9. Para V2 agrupamos por mes y calculamos el promedio, actividad Aggregate 2.
10. Cada una de las funciones aplicadas a los registros genera información sobre los registros aceptados y rechazados en archivos log.

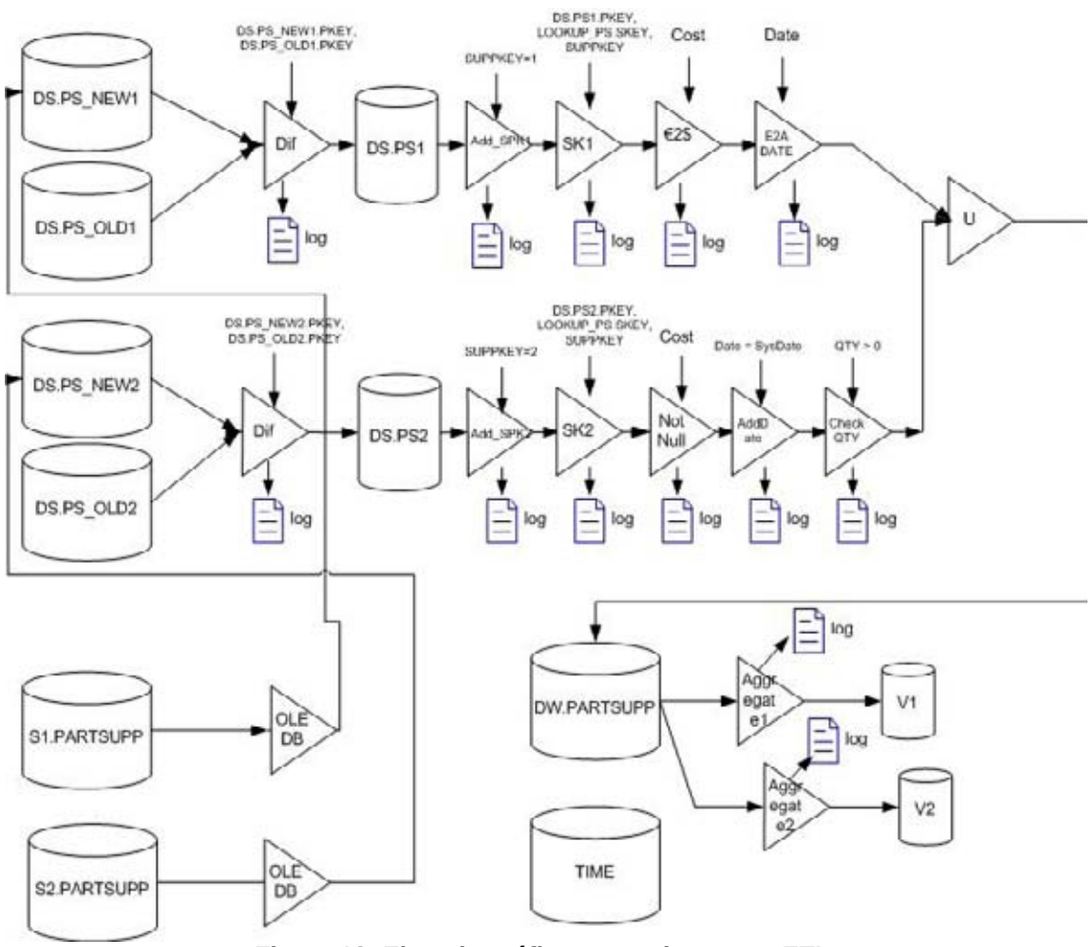


Figura 10: Ejemplo gráfico general proceso ETL.

En la figura 10, damos una definición formal de las actividades y tablas que participan en el escenario del proceso ETL de ejemplo. La estructura completa del proceso ETL, que involucra las actividades, los registros y las funciones pueden modelarse a través de un *“gráfico de arquitectura”*.

Cada *“gráfico de arquitectura”* de un proceso ETL estará compuesto de nodos y aristas. Los tipos de datos, funciones, constantes, atributos, actividades, tablas, constituyen los nodos. En la figura 11 se señala la notación utilizada en los gráficos de arquitectura.

A continuación señalamos el *“grafico de arquitectura”* para dos de las actividades del escenario ETL, las actividades AddSPK1 y SK1, para ambas actividades señalamos los atributos de entrada, IN, y los atributos de salida OUT además de los parámetros requeridos por la actividad. En la figura 12 observamos que la entidad DS.PS1 es entrada para la actividad AddSPK1 la cual además recibe un parámetro a través de la función Add_const1 (encargada de determinar el origen de los datos).

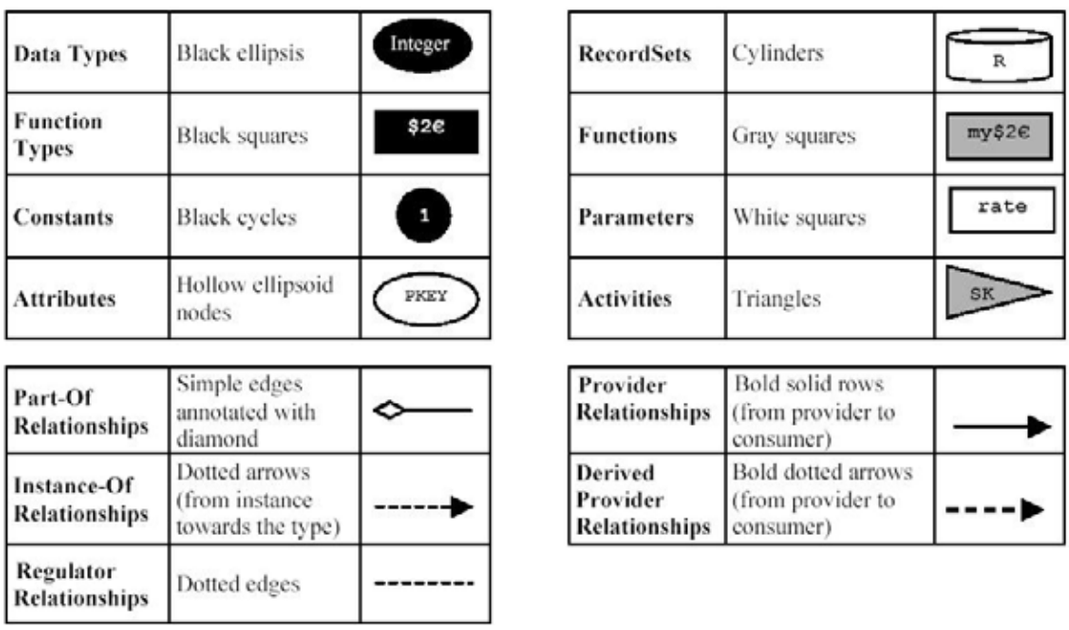


Figura 11: Notación para gráficos de arquitectura [Vas+,02].

El diagrama también nos permite determinar que el resultado de la función Add_const es asignado al atributo SUPKEY en la actividad AddSPK1.

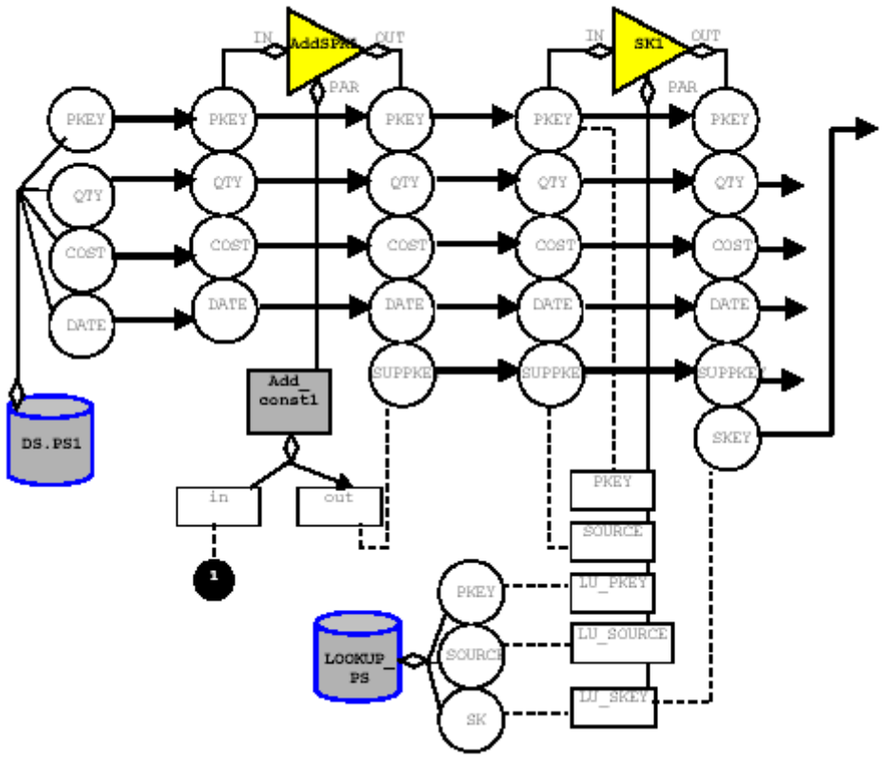


Figura 12: Diagrama de Arquitectura

Proponemos el uso de una notación gráfica como la descrita, con el objetivo de formalizar el proceso de diseño lógico de los diversos escenarios ETL y expresarlos de manera gráfica.

La definición de la periodicidad de cada proceso es externa a la notación antes presentada.

Con la especificación formal de los procesos ETL se finaliza la etapa de diseño. En adelante, con la información recopilada hasta este momento procederemos a la construcción física de nuestro sistema de data warehouse.

3.4 Selección de tecnologías habilitantes.

Realizado el diseño del sistema, es un buen momento para escoger las herramientas de software que nos permitirán realizar la implementación.

Si bien la metodología propuesta es absolutamente independiente de herramientas de software específicas, proponemos un conjunto de herramientas integrables que satisfagan las necesidades específicas de este tipo de sistemas y cuyo costo de licenciamiento sea cero.

Uno de los objetivos de este trabajo es proponer una solución costo efectiva para empresas de tamaño medio y creemos que el conjunto de herramientas propuestas en adelante valida ese requerimiento.

La elección de herramientas comprendió varios niveles para cada uno de los niveles de la implementación (ver tabla 5, figura 13):

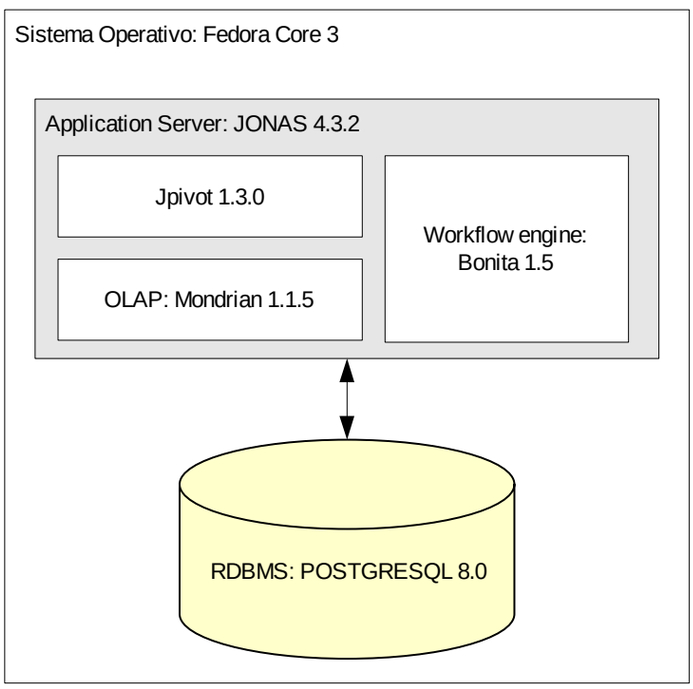


Figura 13: Arquitectura herramientas seleccionadas.

Sistema Operativo	**Linux Fedora Core 3**
RDBMS	**Postgresql 8.0**
Servidor OLAP	**Mondrian 1.1.5**
Presentación OLAP	**JPivot 1.3.0**

Motor de Workflow	**Bonita 1.5**
Servidor de Aplicación J2EE	**Jonas 4.3.2**
IDE de Programación	**Eclipse 3.0**

Tabla 5: Herramientas seleccionadas

A continuación describiremos los motivos y las principales características de cada una de las herramientas seleccionadas.

3.4.1 Sistema Operativo.

Linux es un sistema operativo muy popular a nivel de servidores y no es nuestro objetivo ahondar aun más en sus ya conocidas características, las principales razones por las que se escogió Linux Fedora Core 3 son:

- Es una distribución Open Source
- La comunidad de usuarios es considerable
- Es integrable con el resto de las herramientas seleccionadas.

3.4.2 Bases de Datos.

La selección del RDBMS comprendió básicamente dos alternativas: Postgresql y Mysql, ambas bases de datos open source ampliamente utilizadas. Tres fueron las características fundamentales que inclinaron nuestra decisión hacia PostgreSql.

1. PostgreSql implementa integridad referencial.
2. Postgresql provee de un potente lenguaje de programación interno como PL/pgSQL.
3. Si bien no existe una implementación de vistas materializadas, es muy probable que en el corto plazo exista una.

A continuación describimos las características más importantes de PostgreSql:

- DBMS Objeto-Relacional

PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.

- Altamente Extensible

PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.

- Soporte SQL Comprensivo

PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

- Integridad Referencial

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

- API Flexible

La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

- Lenguajes Procedurales

PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

- MVCC

MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Si alguna vez ha usado algún DBMS con capacidades SQL, tal como MySQL o Access, probablemente habrá notado que hay ocasiones en las que una lectura tiene que esperar para acceder a información de la base de datos. La espera está provocada por usuarios que están escribiendo en la base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros.

Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

- Cliente/Servidor

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

- Write Ahead Logging (WAL)

La característica de PostgreSQL conocida como Write Ahead Logging incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser enormemente beneficioso en el caso de que existiese una caída,

ya que cualesquiera cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato que fue previamente registrado. Una vez el sistema ha quedado restaurado, el usuario puede continuar trabajando desde el punto en que quedó cuando se cayó la base de datos.

3.4.3 Servidor OLAP.

Mondrian es quizás el primer servidor OLAP open source serio y estable, que por el momento consta con una pequeña (en crecimiento) pero activa comunidad.

Mondrian es un servidor OLAP escrito en java, que permite interactuar con grandes conjuntos de datos almacenados en bases de datos SQL sin utilizar SQL. Mondrian es un proyecto alojado en sourceforge.net desde el año 2001 y cuyo primer “release” fue en febrero del 2003, actualmente se encuentra disponible su versión 1.1.5 liberada el 7 de abril del 2005.

La arquitectura del servidor Mondrian consta de cuatro capas, desde el punto de vista del usuario, éstas son:

• Capa de presentación

• Capa de cálculos

• Capa de agregación

• Capa de datos

La capa de presentación establece la manera en la que el usuario ve la información generada por el servidor, así como también establece de qué manera el usuario puede interactuar con el servidor, ya sea generando una nueva consulta o modificando su consulta inicial. Existen diversas maneras de presentar información dimensional al usuario, las más comunes incluyen gráficos de diferentes tipos y el uso de tablas pívot.

La capa de cálculos procesa y valida las consultas hechas al servidor, estas consultas son especificadas en el lenguaje de consultas, desarrollado por Microsoft, MDX (acrónimo de **M**ulti**d**imensional E**x**pressions) que permite la definición y manipulación de objetos y datos multidimensionales. MDX es similar en muchos aspectos a SQL. Los conceptos claves para comprender la estructura de una consulta MDX se encuentran disponibles en la ayuda en línea de Microsoft [URL 16]. Para mejorar el rendimiento, las consultas MDX son evaluadas escalonadamente por la capa de cálculo y los requerimientos de datos son enviados desde esta capa a la capa de agregación. Un *transformador de consultas* permite a las aplicaciones manipular consultas MDX existentes, en lugar de forzarlas a generar una nueva consulta MDX desde cero. En esta capa, además, se maneja la metadata que permite al servidor OLAP encontrar las correspondencias entre el modelo dimensional lógico y las tablas residentes en la base de datos relacional.

La capa de agregaciones maneja resultados pre-calculados en memoria, la capa de cálculo consulta por una determinada información a la capa de agregación, esta información es buscada en el cache del servidor y en caso de no existir la solicitud es enviada a la capa de datos. Mondrian maneja las agregaciones en memoria y no de manera persistente.

La capa de datos es el RDBMS que es responsable de enviar los datos para la capa de agregación, Mondrian utiliza un RDBMS como repositorio de datos (ROLAP) en lugar de un repositorio optimizado para la manipulación de datos multidimensionales. Los desarrolladores de Mondrian consideraron innecesario (desde la perspectiva costo/beneficio) desarrollar un nuevo sistema de almacenaje, cuando el RDBMS provee de uno. Bajo esta perspectiva, Mondrian propone que el RDBMS se haga cargo de manejar todos los datos persistentes, y Mondrian se limitará a generar agregaciones en memoria.

Adicionalmente si el RDBMS soporta vistas materializadas o alguna tecnología similar éstas serán implícitamente utilizadas por Mondrian. Esto es un punto crítico debido a que uno de los principales requerimientos de un servidor OLAP es que sus tiempos de respuesta sean mínimos. Para lograr satisfacer este requerimiento es fundamental la configuración adecuada del RDBMS. Mondrian provee de un nivel de debug, con el cual podemos ver todas las sentencias SQL que el servidor ejecuta contra la base de datos y con esta información podemos optimizar los planes de ejecución de estas consultas mediante la generación de índices, o modificando los parámetros de configuración de la base de datos.

La definición del modelo de datos se hace a través de un esquema que permite la definición a nivel lógico de la estructura de una base de datos multidimensional. En este esquema podemos definir cubos, dimensiones, jerarquías, niveles y todo lo necesario para la construcción de un modelo dimensional (incluso se permite la definición de modelos snowflake). Este esquema no es más que un archivo XML, incluso existe un plugin para Eclipse que permite generarlo gráficamente. Otra de las principales características que hacen de Mondrian una herramienta interesante es que soporta tanto Java OLAP (JOLAP) como XMLA (XML for Analysis). JOLAP es una especificación que pretende satisfacer la necesidad de una API nativa de java que soporte la creación, almacenaje, accesos y mantención de datos y metadatos en servidores OLAP y bases de datos multidimensionales. Por otra parte XMLA es una especificación inicialmente patrocinada por Hyperion y Microsoft Corp. que define un conjunto de interfaces de mensajes XML que utilizan el protocolo SOAP para definir interacción entre aplicaciones clientes y un proveedor de datos analíticos (OLAP) sobre Internet.

Mondrian ha sido testeado utilizando los principales motores de bases de datos, tanto comerciales como open-source, entre ellas PostgreSql por supuesto.

3.4.4 Presentación Olap.

Para la capa de presentación se utilizó una biblioteca de etiquetas JSP llamada JPivot, que permite presentar datos dimensionales a través de tablas pívot, además permite la interacción del usuario con el servidor, implementando navegaciones típicas como slice (cortar un cubo por uno de sus ejes), dice (rotar un cubo), drill down (descender en una jerarquía de un cubo) y roll up (ascender por una jerarquía) (ver figura 14).

Figura 14: Representación utilizando JPivot.

JPivot se integra perfectamente con Mondrian; de hecho, JPivot es distribuido con Mondrian y viceversa.

JPivot cumple con el requerimiento de permitir al usuario interactuar con cubos de datos de manera sencilla e intuitiva, además implementa capacidades de exportación tanto en formato excel como pdf.

3.4.5 Motor de Workflow.

Para la implementación de los procesos ETL se analizaron diferentes implementaciones de workflow. El primer filtro que se estableció fue que la herramienta debía estar implementada en java, esto con el objetivo de mantener la homogeneidad con el resto de las aplicaciones.

Existen numerosas aplicaciones de workflow implementadas en java [URL 5], entre las de más alta difusión en la Web se encuentran WfMOpen [URL 6] y Bonita Workflow [URL 7] ambos implementados bajo la especificación J2EE de Sun.

Para nuestra aplicación optamos por Bonita Workflow principalmente basados en las buenas experiencias del gobierno francés [URL8], y considerando que es patrocinado por ObjectWeb [URL 9], y el INRIA (Instituto Nacional de Investigación en Informática y Automatización, Francia) [URL 10], y además porque satisfacía adecuadamente los requerimientos de nuestras aplicaciones.

Bonita es un sistema flexible de Workflow acorde con la especificación de la WfMC (Workflow Management Coalition) [URL 11], basado en el modelo de Workflow propuesto por el equipo ECCO, que incorpora la anticipación de actividades para hacer su mecanismo de ejecución más flexible. Bonita es Open Source y se distribuye bajo la licencia LGPL.

Bonita provee:

- Un conjunto de herramientas gráficas para realizar definición, control y monitoreo de procesos.
- 100% ambiente Web, con Web Services que encapsulan y publican los métodos de negocio.
- Un motor de Workflow de tercera generación basado en el modelo de anticipación de actividades.

Bonita es un sistema J2EE acorde con la especificación 2.0 de Enterprise Java Beans (EJB) [URL15] ejecutable sobre cualquier servidor de aplicaciones que implemente el estándar J2EE de acuerdo a la especificación 2.x de EJB.

3.4.6 J2EE, Servidor de Aplicaciones.

Bonita requiere de un servidor de aplicaciones J2EE, de la misma manera que JPívot requiere de un contenedor Web, para ejecutar sus etiquetas JSP.

Un servidor de aplicaciones es una implementación por parte de un proveedor de la especificación J2EE, por tanto cumple con la arquitectura J2EE presentada en la figura 15.

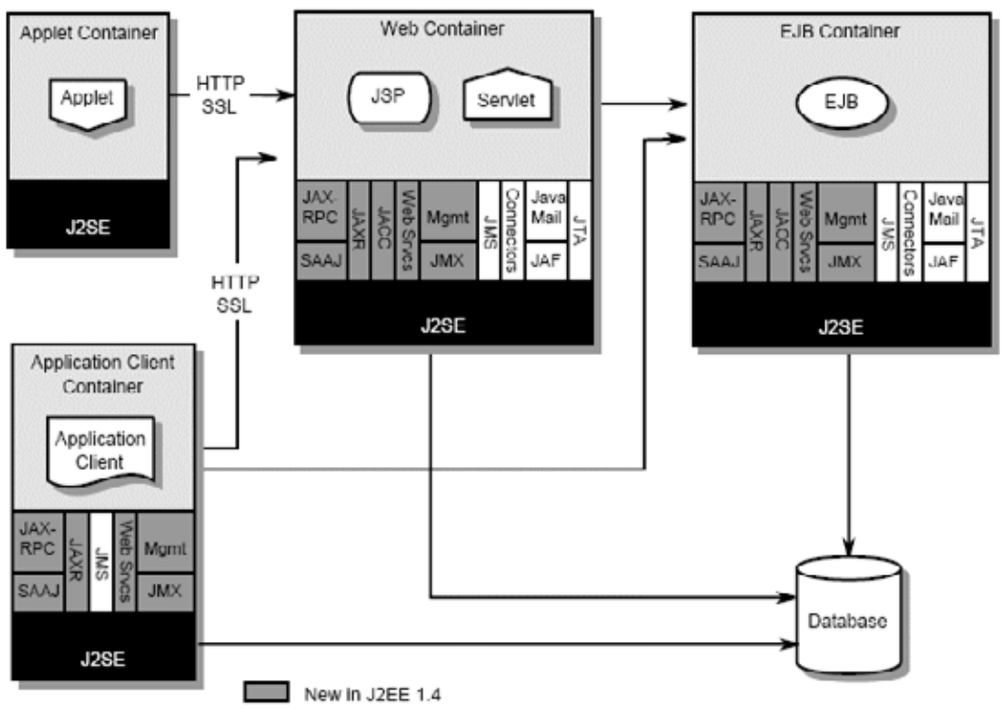


Figura 15: Arquitectura J2EE. [URL14].

Los contenedores, denotados por rectángulos, son ambientes de ejecución J2EE que proporcionan los servicios necesarios a los componentes de aplicaciones, representados en la parte superior de los rectángulos. Cada

tipo de componente de aplicación se ejecuta en un tipo de contenedor específico.

Cada componente de aplicación es una unidad de software autosuficiente, que se ensambla en una aplicación J2EE con sus clases y archivos relacionados y que posee la capacidad de comunicarse con otros componentes. Una aplicación J2EE puede estar compuesta por uno o más componentes.

Existen fundamentalmente 4 tipos de componentes:

1. Aplicaciones Clientes (componentes ejecutados en el cliente).
2. Applets.
3. Java Servlet y Java Server Pages (JSP), componentes Web que se ejecutan en el servidor.
4. Enterprise JavaBeans (EJB) componentes de negocio ejecutados en el servidor.

En el mundo open source, dos son las implementaciones más populares del estándar J2EE, JBoss y Jonas. JBoss es quizás la implementación open-source más utilizada con una comunidad y reconocimiento inmenso; sin embargo, no cuenta con la certificación de Sun, debido principalmente a razones políticas. Jonas, por otro lado, es resultado del esfuerzo de ObjectWeb [URL 9] y se encuentra certificado en su versión 4.3.4.

El servidor de aplicaciones seleccionado fue Jonas básicamente porque Bonita Workflow recomienda su utilización, aun cuando en teoría ambos servidores debieran satisfacer las necesidades.

A continuación señalaremos brevemente las principales características de Jonas.

Jonas implementa:

- EJB, contenedores para aplicaciones J2EE, tanto Web, como cliente.

- Web Service
- Clustering
- Escalabilidad
- Interoperabilidad
- Transacciones Distribuidas
- Soporte de seguridad
- Administración basada en JMX
- Integración con JMS
- J2EECA

Jonas puede ser utilizado en y con:

- SSOO, Windows, Linux, Solaris, AIX, HP-UX, Novell, etc.
- Múltiples máquinas virtuales de java
- Bases de Datos como: Oracle, PostGreSQL, MySQL, SQL Server, DB2, ObjectStore, Informix, Internase, etc.

Arquitectura JonAS.

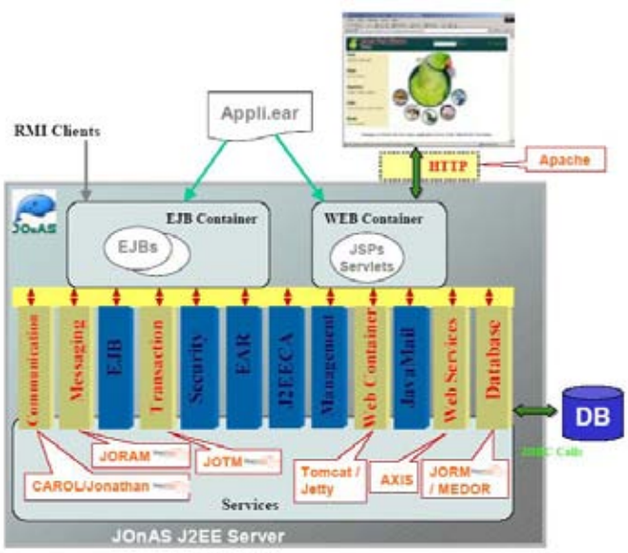


Figura 16: Arquitectura JOnAS.

La arquitectura de Jonas se basa en los servicios. Los componentes principales de la Arquitectura de JOnAS (ver figura 15) son los siguientes servicios:

- Contenedor EJB.
- JTM (Java Transaction Manager).
- Un servicio JDBC para soportar la administración de las conexiones a las bases de datos.
- Un servicio de Mensajería, JOnAS se integra con JORAM que es la implementación Open Source de JMS.
- Servicio de Seguridad.
- Servicio de Administración.
- Un *Connector Resource Service.*
- Un Servicio Contenedor Web para ejecutar Servlet y JSP, actualmente Tomcat o Jetty.
- Un servicio EAR, para instalar aplicaciones J2EE empaquetadas como archivos EAR.
- Servicio de Mail.
- Servicio de Comunicación.

3.4.7 IDE de Programación.

Para la implementación de los ETL se escogió el lenguaje de programación java y con el objetivo de acelerar nuestros tiempos de desarrollo nos propusimos utilizar una herramienta IDE que tuviese habilidades de compilación ejecución y debug de aplicaciones, entre otras. Eclipse [URL 12] es un IDE distribuido bajo la licencia CPL (Common Public License) (no es open source) extensible a través de plugins y que permite el desarrollo de

aplicaciones java. La potencia de Eclipse radica en su flexibilidad y extensibilidad, para el caso específico del proyecto utilizamos un plugin llamado Lomboz desarrollado por ObjectWeb que permite el desarrollo de aplicaciones J2EE utilizando Eclipse (Fig. 16)

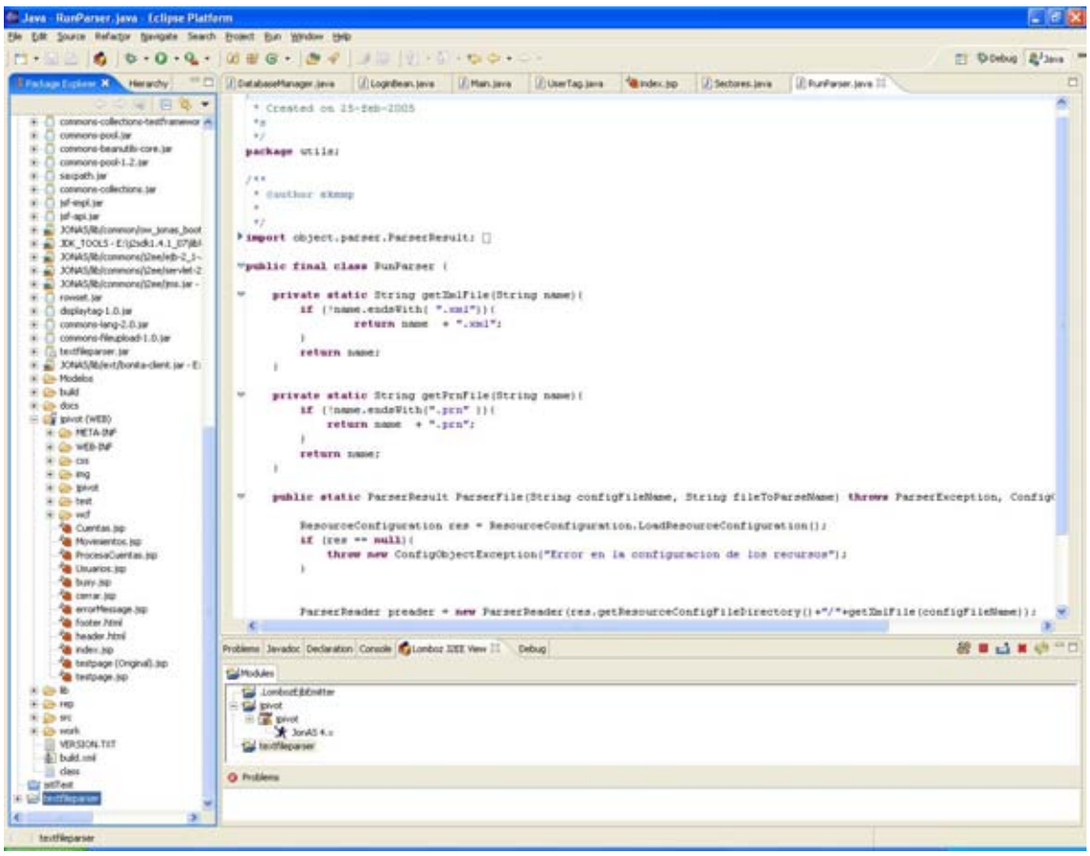
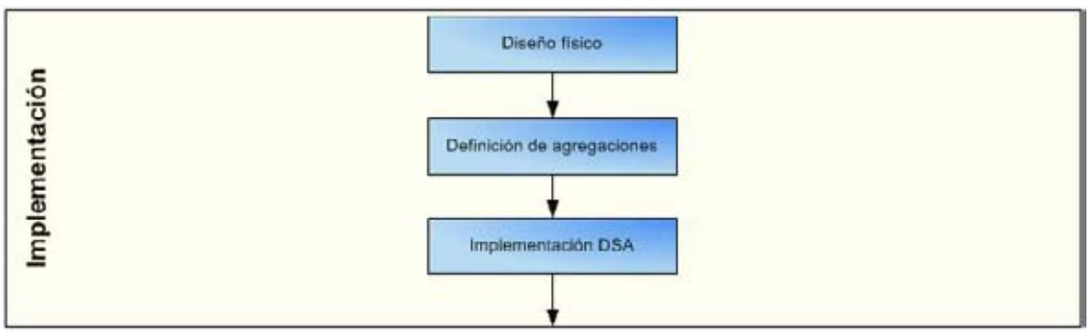


Figura 17: IDE Eclipse.

4. Implementación.

4.1 Objetivos.

Con los requerimientos capturados, los modelos de datos generados y los procesos de carga definidos debemos implementar el sistema final que estará disponible para los usuarios. El principal objetivo de esta etapa es obtener como resultado un sistema funcional acorde a las especificaciones definidas en el diseño y el análisis de requerimientos.



4.2 Diseño físico.

En esta etapa nos proponemos definir algunos requerimientos y consideraciones con miras a plasmar nuestro modelo lógico en la base de datos física. Los detalles de la implementación variarán a través de las diferentes tecnologías que evolucionan rápidamente en el mundo de la informática. Debido a esto, pretendemos dar un vistazo de manera general al tema.

Los detalles relacionados con el diseño físico y la implementación de la base de datos están fuertemente ligados a las características del proyecto, esto es, el modelo de datos lógico, el RDBMS, los volúmenes de datos, las herramientas de acceso, etc. Sin embargo, podemos definir algunos patrones aconsejables de seguir una vez emprendida esta tarea.

4.2.1 Definir un estándar para los nombres de los objetos de la base de datos.

Esencialmente, existen tres componentes básicos dentro del nombre de cualquier objeto en una base de datos [Kim+, 98]:

- Palabras Primarias: describen a que se refiere el elemento, responden a la pregunta ¿qué es?, algunos ejemplos son productos, clientes, ciudad etc.
- Palabras de clase: clasifican los objetos en grupos mayores, responden a la pregunta ¿qué tipo de objeto es?, algunos ejemplos son, ID, fecha, nombre, descripción etc.
- Cuantificadores: son opcionales y pueden ayudar a definir a la palabra primaria y a la palabra de clase, algunos ejemplos son: comienzo, primero, segundo, etc.

Los nombres de los objetos de la base de datos generalmente seguirán una sintaxis por ejemplo Cuantificador _Clase _Primaria

Es recomendable también definir los nombres de los objetos físicos de manera descriptiva evitando abreviaciones. En rigor, podemos utilizar un conjunto pequeño de abreviaciones simples; por ejemplo desc, obviamente se refiere a descripción, pero dmk no queda claro que se refiera por ejemplo a departamento de marketing, la meta es encontrar un balance entre ser demasiado descriptivo (valor_neto_cuentas_canceladas_por_cliente) y demasiado vago (valor_neto), hay que considerar que es muy posible que estos nombres sean los encabezados de las columnas de muchos reportes para diferentes aplicaciones. Es muy importante que se documente el estándar de nombres definido.

4.2.2 Desarrollo del modelo de datos físico.

Tomando como referencia el modelo lógico creado con anterioridad, la idea es en lo posible generar una copia de ese modelo lógico en la base de datos física. Adicionalmente, el modelo contendrá tablas anexas que corresponderán al DSA y a tablas diseñadas para la mantención, pero la principal diferencia entre el modelo físico y el modelo lógico tiene que ver con el nivel de detalle, en el modelo físico es necesario especificar con precisión los tipos de datos y algunos otros parámetros que variarán dependiendo del RDBMS (tamaño de las tablas, parámetros de almacenamiento, particionamiento, técnicas de indexación, etc.).

Es recomendable utilizar herramientas gráficas para el modelado, éstas nos ayudarán entre otras cosas a verificar que la convención de nombres definida se esté utilizando, generar el diccionario de datos, integrar más

fácilmente el data warehouse con el resto de los sistemas de la organización, etc.

Con respecto a la selección de los tipos de datos es importante escoger para las columnas que serán claves aquellos tipos de datos más eficientes. En general el tipo de dato más eficiente es el entero, pero esto puede variar dependiendo del RDBMS.

Determine previamente qué columnas permitirán valores nulos y organice físicamente las columnas que no permiten nulos primero que las que sí permiten valores nulos, esto para ahorrar espacio en disco.

Declare todas las claves primarias y foráneas explícitamente, sólo si estamos seguros que nuestro proceso de carga es limpio y robusto, podremos evitar esto.

4.2.3 Desarrollo del plan de indexación.

Cada RDBMS implementa diferentes algoritmos de indexación, diseñados para diferentes tipos de columnas. El algoritmo indexación más común es el **B-Tree** [URL 17] (Balanced o Bayer - Tree), este tipo de índice es ideal para columnas de alta cardinalidad, por ejemplo, columnas claves auto numéricas, B-Tree organiza la información en un árbol balanceado (Fig. 17), por lo que encontrar el valor de una determinada fila involucra comparar el valor con cada rama, si el valor es mayor subimos un nivel, si es menor descendemos un nivel, si es igual, hemos encontrado el valor.

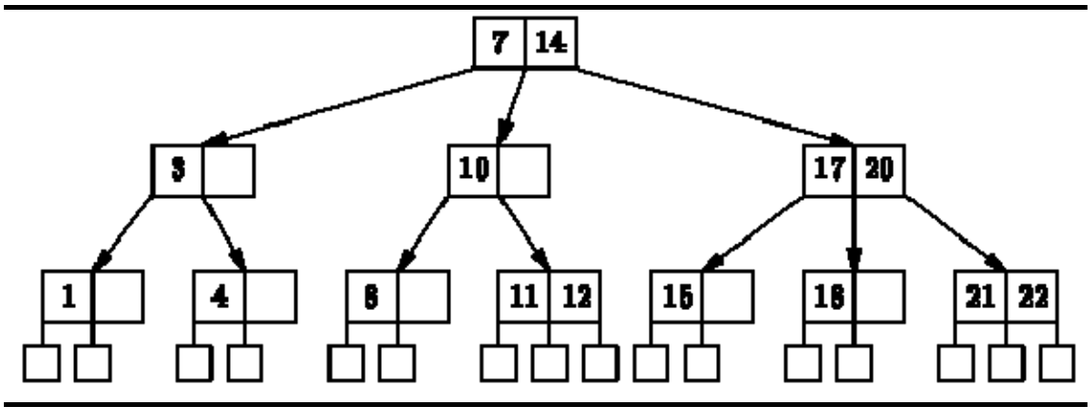


Fig. 17: Estructura de un B-tree.

Existen otros algoritmos de indexación útiles para columnas con otras características, como los índices de mapa de bit o **Bitmapped Index**, estos son útiles para columnas de baja cardinalidad, como por ejemplo una columna género (M o F). Otro tipo común de índices es el **R-Tree** similar al B-Tree pero aplicable sólo a un conjunto de filas, dada una condición; es decir, podemos indexar sólo un segmento de la tabla. Un ejemplo clásico de la utilización de este tipo de índices se presenta en una tabla con una columna fecha, utilizando un índice R-Tree podríamos indexar la columna fecha sólo para el periodo de tiempo correspondiente a los últimos x meses, los que sabemos serán consultados más frecuentemente.

Muchos RDBMS implementan estos algoritmos y otros más de características propietarias, razón por la cual es fundamental conocer las bondades de los sistemas de indexación del RDBMS seleccionado.

La mayoría de los RDBMS generan por omisión un índice B-Tree en la clave primaria. El orden con el cual se definen las columnas es importante, por ejemplo un índice B-Tree generado para las columnas (a, b) será utilizado toda vez que consultemos por (a, b) o por (a) pero no será utilizado si consultamos por (b).

Dado que en la tabla de hechos la mayoría de las consultas tendrán como principal punto de entrada el tiempo no es una mala idea definir en la estructura de la tabla la clave IdTiempo primero.

La creación de otros índices en la tabla de hechos (además del generado por omisión en la clave primaria) dependerá de cada situación.

Para el caso de las tablas de dimensión éstas deberán tener una clave primaria simple y por ende un índice generado en esa clave. Es recomendable generar índices de mapa de bit en aquellas columnas frecuentemente utilizadas en las condiciones de filtro y que sean de baja cardinalidad (esto siempre y cuando el RDBMS soporte este tipo de índices). De igual manera, para dimensiones grandes, si conocemos un conjunto de atributos que eventualmente serán consultados frecuentemente, podemos crear un índice compuesto de estas columnas.

Finalmente, debemos tener en cuenta que algunos RDBMS no reconstruyen los índices después de una operación de actualización (insert o update), esto nos obligará a regenerarlos luego de cada proceso de carga, de manera de mantenerlos actualizados.

4.3 Agregaciones.

Uno de los requerimientos básicos de un data warehouse tiene que ver con su rendimiento, es decir la capacidad de responder las consultas en tiempos razonables y aceptables. Idealmente, nuestro diseño nos llevará a construir un conjunto pequeño de tablas dimensiones y otro conjunto de grandes tablas de hechos, las cuales podrían superar los millones de registros conforme pase el tiempo, dependiendo del tamaño de nuestro negocio.

Para asegurar que los tiempos de respuesta de nuestro data warehouse sean aceptables podemos considerar variadas alternativas, que dependerán directamente de las herramientas de las que dispongamos a la hora de implementar nuestro proyecto.

La solución ideal es implementar un conjunto de resúmenes por cada tabla de hechos; esto es, precalcular algunos valores agregados y almacenarlos en la base de datos para incrementar el rendimiento, de manera que cada vez que se ejecute una consulta sobre el data warehouse éste seleccionará adecuadamente de dónde extraer sus resultados, eligiendo los resúmenes correctos cada vez que sea posible.

Esta tarea en específico la deberá realizar el DBMS el cual conocerá de la existencia de estas agregaciones a través de la especificación de algún tipo de metadata.

Existen principalmente dos técnicas, ampliamente aceptadas, para la implementación de resúmenes, la primera consiste en almacenar estos registros precalculados en nuestra tabla de hechos y dimensiones original, esto implica que debemos de alguna manera introducir un nuevo campo “NIVEL” en cada una de las tablas afectadas con el objetivo de distinguir los registros

resumidos de los originales, y no cometer errores en los cálculos. La segunda alternativa, más recomendada, consiste en seleccionar cada resumen y ubicarlo en nuevas tablas de hechos, que ahora se rodearán de nuevas tablas dimensionales más pequeñas que contendrán sólo aquellos atributos que tengan sentido para cada nivel de resumen. Ambas técnicas producen el mismo número extra de registros, la diferencia es sólo dónde los almacenamos. [Kim, 95].

Las principales características de un buen conjunto de resúmenes son: [Kim, 96]:

- Proveer de dramáticas mejoras en los tiempos de respuestas para todas las posibles categorías que un usuario pueda seleccionar.
- Agregar una cantidad razonable de datos extra, razonable a los ojos del DBA.
- Ser completamente transparente al usuario final y a los desarrolladores de aplicaciones finales.
- Entregar beneficios a todos los usuarios del data warehouse, independiente de las herramientas de consulta que estos utilicen.
- Impactar con el más bajo costo posible los procesos de extracción transformación y carga.
- No generar demasiadas nuevas responsabilidades sobre el DBA, por lo que la metadata debe ser lo más reducida y fácil de mantener posible.

Existen varios proveedores de bases de datos que en la actualidad permiten la implementación de estas metodologías de agregación. Oracle, por ejemplo, define **Vistas Materializadas** que permiten definir vistas persistentes que mejoran el rendimiento de las consultas[URL 4], del mismo modo SQL

Server 2000 a través de sus **Vistas Indexadas** permite realizar esta misma tarea [Del, 01].

Es necesario que esta implementación sea transparente al usuario e incluso debe ser transparente a las aplicaciones finales, es por esto que no sirve definir agregaciones y esperar que el usuario seleccione la agregación desde un formulario por ejemplo, esto sería un error que sólo llevaría a la confusión de nuestros usuarios finales. Tampoco es una alternativa programar nuestras aplicaciones finales de manera que sean ellas las que seleccionen las agregaciones. Esto último, además de generar una complicación adicional a su desarrollo, limitaría las posibilidades de nuestro DBA de modificar o definir nuevas agregaciones según lo estime conveniente, ya que por cada modificación al conjunto de agregaciones deberemos hacer las modificaciones necesarias en nuestras aplicaciones finales, con el costo que esto significa.

Lamentablemente a la fecha de realización de este proyecto no existen aún DBMS Open Source que incluyan en su distribución alguna de estas características para la gestión de agregaciones; sin embargo, para el caso de PostgreSQL existe un primer esfuerzo en esta dirección, a través de la implementación de vistas materializadas desde código PL/pgSQL, presentadas en el articulo “Materialized Views in PostgreSQL” [Gar, 04]. Además, para incrementar el rendimiento debemos realizar los afinamientos (tunning) necesarios al DBMS, alterando su configuración y agregando índices en los lugares adecuados. Existen también algunas aplicaciones que realizan resúmenes en memoria que permiten mejorar considerablemente los tiempos de respuesta.

Por último, es importante aclarar que no es necesario ni efectivo crear agregaciones para cada una de las tablas del sistema. Debemos seleccionar cuidadosamente las tablas sobre las que será realmente efectiva la creación de

agregaciones, con este objetivo establecemos dos criterios de evaluación [Kim+, 98]:

- **Requisitos comúnmente accedidos.**

  Consiste en la determinación de todos aquellos atributos de cada dimensión que son frecuentemente accedidos, luego determinar qué combinaciones de estos atributos son accedidas más frecuentemente, estas serán candidatas para formar agregaciones.

- **Distribución estadística de los datos.**

  Debemos determinar el número de valores para los atributos que hemos establecido como candidatos, por ejemplo, al nivel más fino una tabla contiene 1,000.000 de registros y el atributo candidato pertenece al siguiente nivel en la jerarquía y contiene 500.000 valores distintos. Este atributo no es un buen candidato para generar agregaciones, debido a que estaremos manejando un cantidad similar de registros, Distinto es si el atributo candidato varía entre 50.000 valores, en este caso una agregación puede mejorar significativamente el rendimiento, ya que el sistema se enfrentará con un número considerablemente menor de registros.

La combinación costo-efectiva de hardware específico y un conjunto balanceado de agregaciones y afinamientos al DBMS deberán sustentar el rendimiento del data warehouse permitiéndonos entregar tiempos de respuesta aceptables.

4.4 Implementación del área de datos intermedia (DSA).

Esta es una etapa crítica dentro del proyecto, debemos planificar el proceso que poblará de datos periódicamente a nuestro sistema. Dado que en una etapa anterior hemos definido las características de los procesos ETL en documentos formales, ahora debemos implementar físicamente los procesos definidos en esos documentos. En este punto probablemente ya habremos definido si utilizaremos alguna herramienta ETL o si codificaremos el proceso entero. Cualquiera sea la elección, debemos considerar:

- Crear casos de prueba: es fundamental generar casos de prueba para cada uno de los casos posibles por cada ETL. Esto implica probar las cargas.
- Generación de archivos de log: es importante generar información de cada proceso ETL que permita rastrear el progreso y los estados por los que transita cada ETL, existen herramientas que permiten generar información de log en el código fuente de diferentes niveles (DEBUG, INFO, ERROR etc.), estas herramientas son extremadamente útiles para la detección de errores post implantación.
- Generar documentación acerca de los detalles de la implementación, esto incluye generar documentación para las aplicaciones codificadas (si las hay), documentar todos los parámetros utilizados, documentar el modelo de datos utilizado para el DSA (si es relacional).
- Generación de código bien documentado para las aplicaciones codificadas, siguiendo estándares para el nombramiento de variables métodos, clases etc. Existen innumerables herramientas que ayudan a facilitar la tarea de generación de documentación (javadoc, ndoc, etc).

- Es buena idea guardar los archivos de configuración de la base de datos, así como también copias de los scripts con los que se generó la base de datos.

Una vez cargados los datos en el DSA debemos transferirlos al data warehouse, esto se limita simplemente a un proceso de carga, ya que hemos realizado todas las transformaciones necesarias en el DSA. Si el RDBMS del DSA es el mismo que el del data warehouse es recomendable tener en cuenta los volúmenes de información cargados cada vez, las operaciones insert into pueden ser ineficientes, así como también hay que tener en mente el tamaño de los segmentos de rollback y de log ya que el proceso puede fallar por cualquiera de estos dos motivos. Muchos RDBMS proveen de aplicaciones para realizar cargas masivas entre bases de datos, si este es el caso del RDBMS escogido, es una buena idea utilizarlos.

5. Implantación.

Esta etapa considera todos aquellos procesos que involucran la implantación formal, la mantención y el crecimiento del sistema data warehouse.

5.1 Objetivos.

- Definir la política de educación y capacitación de los usuarios.
- Definir las estrategias de soporte para los usuarios.
- Preservar el rendimiento del sistema.
- Mantener el foco en los usuarios.
- Detectar nuevas posibilidades de crecimiento, generando proyectos con miras al crecimiento del sistema.

5.2 Capacitaciones.

“Una estrategia de educación robusta para los usuarios finales del negocio es un prerrequisito para el éxito de un data warehouse” [KIM+,98], La sofisticación del sistema o lo interesante de la información incluida en él, pueden no tener valor sin la apropiada educación para los usuarios.

Desde una perspectiva técnica podemos dividir en tres aspectos básicos el sistema data warehouse: los datos contenidos, las aplicaciones finales y las herramientas de acceso a datos. Los usuarios, sin embargo, no comprenden los límites entre estos componentes, y ante sus ojos visualizan el data warehouse como un sólo elemento. Debemos mantener esa perspectiva en el usuario, básicamente por un sentido de transparencia y simplicidad.

La capacitación de los usuarios no debe sólo centrarse en el entrenamiento de cómo utilizar un conjunto de aplicaciones finales, si no que, debe incluir de igual manera, un entrenamiento en los contenidos que publica el data warehouse.

La educación sobre los contenidos debe incluir las estructuras, jerarquías y definiciones del sistema. Los diagramas dimensionales diseñados durante el capitulo 3 son muy útiles para ayudar a la comprensión de los usuarios.

El objetivo es que los usuarios estén conscientes de la información que está disponible en el data warehouse, así como de comprender cómo está organizada la información dentro del sistema. Un usuario que comprende estos aspectos será capaz de encontrar la información que requiere navegando a través de las diferentes estructuras lógicas del sistema.

La educación de los usuarios sobre la utilización de las herramientas finales debe introducirlos sobre las características, objetivos, modos de uso, opciones etc. de la herramienta.

Todos los procesos de educación deben ocurrir una vez el sistema está completamente operativo, la idea es que una vez capacitados los usuarios éstos puedan comenzar inmediatamente a utilizar el sistema.

5.3 Mantención y Crecimiento.

5.3.1 Focalización en los usuarios.

Para asegurar el éxito de la implantación de nuestro sistema, debemos programar sesiones de capacitación para los usuarios del sistema, y posterior a esto, mantenernos lo más cercano posible a ellos, un factor de éxito, que no debemos subestimar, tiene relación con la acogida que dan los usuarios al nuevo sistema, las noticias corren rápido dentro de las empresas y un usuario insatisfecho puede generar publicidad negativa, por el contrario un usuario satisfecho alentará a sus compañero acerca de las bondades del nuevo sistema.

La metodología descrita propone el desarrollo evolutivo de este tipo de sistemas. Bajo esta perspectiva, nuestra primera implementación nunca comprenderá la integración total de la empresa, el crecimiento del sistema es natural y señal inequívoca de que las cosas andan bien; en rigor, nuestro sistema nunca dejará de crecer, debido a la naturaleza cambiante de toda organización. Considerando lo anterior, proponemos la integración gradual de aquellas áreas del negocio faltantes de manera oportunista, es decir, no forzar la marcha por nuestro propio ímpetu de hacer todo lo antes posible, debemos esperar que las evidentes ventajas que proporcionará el sistema sobre aquellas áreas integradas hagan reaccionar a la organización y a los usuarios respecto la utilidad de agregar nuevas áreas.

Cada vez que deseemos extender nuestro sistema a nuevas áreas del negocio debemos reiniciar el proceso desde la captura de requerimientos de los nuevos usuarios hasta la implementación física.

Bajo este contexto toma importancia la arquitectura de bus del data warehouse, que incluye las dimensiones y hechos conformados, que se transforman en los cimientos de nuestro sistema data warehouse.

5.3.2 Focalización en el sistema.

Por último debemos preocuparnos de la mantención física del sistema, eso implica básicamente:

- Asegurar la consistencia de la información cargada. Los procesos ETL son críticos y no hay lugar para errores, los usuarios demandarán por la información más reciente, y es nuestra tarea ponerla a su disposición.
- Asegurar la calidad de los tiempos de respuesta, debemos realizar las tareas que sean necesarias en cuanto a optimización del RDBMS, optimización de consultas y optimización de herramientas, para mantener tiempos de respuesta aceptables conforme vaya aumentando el volumen de datos.
- Es necesario mantenerse al tanto de los posibles cambios que puedan sufrir los sistemas fuente, lamentablemente es probable que nuestros procesos ETL cambien tan frecuentemente como surjan modificaciones en los sistemas fuente.

Parte 2: Desarrollo de un Datamart para CCK.

1. Introducción.

Con el objetivo de validar la metodología propuesta en la primera parte, nos propusimos seleccionar una empresa valdiviana para estudiar sus procesos de negocios y sistemas de información de manera de proponer el desarrollo de una primera etapa de un sistema data warehouse, bajo el concepto de desarrollo evolutivo propuesto en la metodología. Este último concepto es fundamental, con miras a asegurar la escalabilidad de la solución desarrollada. Los parámetros que utilizamos para realizar esta selección fueron:

- Existencia de información histórica.
- Existencia de un sistema informático a nivel operacional.
- Disponibilidad e interés de la empresa.

Como resultado de este proceso de selección se escogió a la Compañía Cervecera Kunstmann, en adelante CCK, ubicada en el Km 5 camino a Niebla. Esta empresa satisfizo los requerimientos establecidos.

Una de las condiciones establecidas por CCK tuvo que ver con la confidencialidad de la información compartida, motivo por el cual, en adelante nos enfocaremos en aspectos globales del desarrollo del proyecto omitiendo referenciar cualquier información que pueda ser considerada inapropiada por nuestro cliente.

2. Análisis de Requerimientos.

Seleccionada la empresa, procedimos a programar un calendario de entrevistas con los potenciales clientes. En el área administrativa y de gestión de la empresa el número de empleados es de 6 personas, se programaron 3 entrevistas, una con el Jefe de Administración y Finanzas, otra con el Jefe de Contabilidad y por último una con el Encargado de Ventas, Activo Fijo, y Existencias, este último quizás el usuario con mayor conocimiento a nivel operativo de los procesos.

Uno de los objetivos principales de este proceso de entrevistas fue establecer el alcance del proyecto de manera de no plantearse un objetivo demasiado ambicioso o demasiado simplista.

2.1 Análisis de las características del negocio.

Con el objetivo de comprender las características generales de la empresa CCK desarrollamos un cuestionario guía (ver tabla 6), similar al presentado en el capitulo 2.2 dividido en dos áreas principales, objetivos del negocio y requerimientos, la primera orientada a determinar las características y políticas macro de la empresa, y la segunda orientada a determinar requerimientos más concretos del usuario con respecto a sus funciones.

Objetivos de Negocio y similares
• Describa su trabajo y como se relaciona con el resto de la empresa • Describa cuales son sus responsabilidades • ¿Cuáles son los objetivos de su organización? • ¿Cuáles son los objetivos de esta área en específico? • ¿Cuál es su objetivo principal? • ¿Cómo mide actualmente su desempeño? • ¿Utiliza indicadores? ¿Cuáles son sus indicadores de éxito? • ¿Cómo sabe que lo esta haciendo bien? • ¿Qué tan frecuentemente evalúa lo anterior? • A su modo de ver ¿Cuáles son los principales problemas que enfrenta hoy en día su empresa? • ¿Cuáles son los principales problemas que enfrenta hoy su área e negocio? • ¿Cómo identifica estos problemas? • ¿Tiene alguna capacidad de anticipación sobre estos problemas? ¿Cómo funciona?
Requerimientos
• ¿Qué tipo de análisis realiza frecuentemente? ¿Qué información usa para estos análisis? • ¿Cómo obtiene esta información? ¿Cómo la procesa? • ¿Qué información utiliza usted actualmente para tomar sus decisiones? • ¿Cómo usa esta información en sus decisiones? • ¿Qué problemas tiene actualmente para conseguir la información que necesita? • ¿Existe información con la cual le gustaría contar y que actualmente no posea? • ¿Qué capacidades de análisis le gustaría tener? • ¿Existen cuellos de botella específicos para obtener información? • ¿Qué necesidades tiene de información histórica? • ¿Con qué profundidad? • ¿Qué posibilidades ve de optimizar su área en base a optimizar la información que recibe? • ¿Qué reportes usa actualmente? ¿Cuáles son los datos importantes en estos reportes?

• ¿Cómo se traduce esto financieramente?

Tabla 6: Cuestionario guía entrevistas

Previo al inicio de cualquier entrevista, le explicamos al entrevistado la idea del proyecto a grandes rasgos y los objetivos de la entrevista.

El resultado de las entrevistas nos permitió determinar características de CCK en los niveles planteados (globales y específicos). Entre las características globales más importantes podemos señalar:

- CCK apuesta a mantener y mejorar la calidad de sus productos al largo plazo buscando diferenciación con el resto del mercado.
- CCK se presenta como una empresa productora de cerveza y cuyo canal de distribución es CCU para todo el país con excepción de la provincia de Valdivia, dentro de la cual continúan vendiendo directamente a sus clientes finales.
- CCK apuesta al largo plazo en su alianza con CCU, siendo éste su principal cliente.
- CCK conserva como valor fundamental su marca, de reconocido prestido a nivel nacional.

Con respecto a los requisitos específicos determinamos que el principal problema al que se enfrenaban los usuarios de CCK tiene relación con las características hostiles de su actual sistema informático de nivel operacional.

CCK cuenta con un sistema ERP llamado INFORMAT que utiliza en modalidad de arriendo, este sistema concentra toda la información operativa que genera CCK en los niveles financiero-contables e información de ventas, la versión del sistema que poseen se ejecuta completamente en modo consola y permite la generación de un número limitado de reportes definidos por la

aplicación los cuales son presentados al usuario en tres modalidades: en pantalla, en impresora, o a través de un archivo plano de texto (ver figura 18).

CERVECERA KUNTSMANN — Venta por Rubros : Septiembre 2004 — PAGINA : 1 — PGM : [VEF.036]

Rubro : 0001 - SUPERMERCADO
Vendedor : 24.001 - ANITA JARAMILLO ALVAREZ

Producto (Descripcion)	Unidades Fisicas ----Sep 2004----	--Venta- Sep 2004
60.10.01.101 - CAJA ALE 1/24	.00	,796
60.10.01.120 - CAJA ALE MIEL 1/24	.00	,200
60.10.01.201 - CAJA BOCK 1/24	.00	,912
60.10.02.001 - DISPLAY LAGER 6/4	.00	,038
60.10.02.101 - DISPLAY ALE 6/4	.00	,988
60.10.02.201 - DISPLAY BOCK 6/4	.00	,692
60.20.02.001 - DISPLAY LATA PREMIUM 6/4	.00	,229
60.99.01.001 - FLETES CAJA LAGER	.00	,611
60.99.01.002 - FLETES CJA TB.ALE	.00	,032
60.99.01.003 - FLETES CAJA BOCK	.00	,387
60.99.01.004 - FLETES L PREMUIM	.00	,303
60.99.01.008 - FLETE CAJA ALE MIEL	.00	,546
*** Total Vendedor : ANITA JARAMILLO ALVAREZ		,734

Figura 18: Ejemplo reporte archivo plano sistema INFORMAT.

Estos archivos planos son difícilmente manipulables utilizando planillas de cálculo lo que genera una gran cantidad de trabajo operativo, los usuarios se ven forzados a generar los reportes deseados cruzando información obtenida de múltiples informes y para esto utilizan gran parte de su tiempo, produciendo esto un costo directo a la empresa en HH.

Existen dos conjuntos de informes constantes que son entregados periódicamente al directorio, los primeros informes incluyen información de los estados financieros, balance, estado de resultados etc., y los segundos incluyen información de ventas, clasificada por diferentes criterios.

Con respecto a la información contable podemos señalar que CCK contabiliza mensualmente. Específicamente existen segmentos de la información que son contabilizados en tiempo real, (al momento de generarse el movimiento), y existen otros movimientos, por ejemplo los movimientos producto de las ventas, que son centralizados sólo una vez, al cierre del periodo contable.

Con respecto a los ingresos por venta de cerveza y licores, también es importante señalar que, para efectos contables, CCK detalla el precio final de sus productos en “Flete” y el costo del producto, esto es un esfuerzo por

sincerar los costos que el cliente paga por la cerveza versus los costos que paga por su distribución, fundamentalmente debido a que ambos ítems están sujetos a impuestos diferentes, esto significa que por cada producto que contiene alcohol CCK genera un producto igual pero cuyo precio corresponde al "Flete"

La información de ventas se produce por los documentos que genera CCK, ya sea facturas, boletas o notas de crédito. Cada transacción incluye información del cliente, vendedor, bodega que provee, fecha, condición de pago y el total en pesos del documento, además de los productos transados, para los cuales se especifica el código del producto, el número de unidades físicas, el precio unitario, el valor neto. Para todos los informes de gestión se utiliza solamente el valor neto, esto es antes de aplicar impuesto.

En la actualidad, como mencionamos anteriormente, el principal cliente de CCK es CCU, esta última adquiere sobre el 50% de los productos que CCK fabrica. Bajo esta perspectiva, CCK posee un número pequeño de clientes, CCU y el conjunto de clientes existentes dentro de la provincia. CCK agrupa estos clientes por segmentos: Supermercados, Hogar, Inmediato y Mayorista. Adicionalmente, CCK utiliza información acerca de los rubros de sus clientes. Cada cliente está asociado a un vendedor que señala que éste es el enlace entre el cliente y la empresa.

Para los productos CCK realiza dos clasificaciones principales, la primera llamada internamente "MIX" dice relación con el tipo de embase que se utiliza en cada producto, estos son básicamente: Botella, Kegs y Lata; sin embargo CCK fabrica otros productos fuera de estas categorías, como son los licores y los souvenires y estos no figuran en los informes de gestión dado que se consideran sus ingresos como marginales.

Tampoco son considerados actualmente en los informes de gestión los ingresos percibidos por el concepto de “Flete” que mencionamos anteriormente. La segunda clasificación utilizada para agrupar los productos, es llamada internamente “PINTA” y se refiere al tipo de producto (Lager, Bock, etc.). Nuevamente, CCK utiliza sólo aquellas categorías de impacto, sin considerar ni clasificar los productos como licores y souvenir.

Existe una última clasificación importante de mencionar con respecto a los productos, y es una clasificación que discrimina los productos terminados con las materias primas, esto debido a que las materias primas igual son ingresadas como productos, ya que ocasionalmente son vendidas directamente, principalmente a CCU, cuando esta última sufre de algún problema puntual de abastecimiento. Para todo efecto, en los informes de gestión sólo se incluyen los productos terminados.

Con anterioridad a la alianza con CCU la información de vendedores era importante, ya que existía un número importante a lo largo del país, pero en la actualidad esto se reduce a un vendedor directo para la provincia y a la oficina de CCK que trata directamente con CCU.

Con respecto a los requerimientos de los usuarios estos se centraban básicamente en lograr generar los reportes e informes descritos con anterioridad de manera más simple, rápida y flexible, características que según ellos les permitirían dedicar más tiempo a generar ventajas competitivas y a analizar la información con mayor profundidad. También surgió la inquietud de ver la posibilidad de definir algunos indicadores financieros.

2.2 Análisis de las fuentes de Información.

Como mencionamos anteriormente, el sistema informático que actualmente soporta los procesos de CCK está contenido en un ERP llamado INFORMAT, este sistema se encuentra físicamente en Santiago y es accedido a través de una conexión dedicada.

Quizás en este punto se nos presentó el principal desafío debido básicamente a los siguientes factores:

- No existe en CCK un experto formal en la utilización del INFORMAT, en general cada usuario conoce los módulos en los que le corresponde trabajar, sin embargo sí existía un usuario con mayor experiencia en el sistema que conocía parte importante de los módulos.
- Existe un pobre canal de comunicación entre CCK e INFORMAT, lamentablemente con el paso del tiempo el vínculo entre CCK e INFORMAT ha desaparecido prácticamente, lo que dificultó la posibilidad de resolver dudas rápidamente.
- Dado lo mencionado en el punto anterior, existe una gran cantidad de información en el sistema que ya no es utilizada; por ejemplo, la clasificación de productos “MIX”, mencionadas en el punto anterior, no existe en el maestro de productos de INFORMAT; sin embargo, existen al menos dos clasificaciones que no tienen validez actualmente para la empresa, esto es debe, creemos, a que los parámetros iniciales bajo los cuales se configuró el sistema han ido cambiando desde la perspectiva del negocio, pero no desde la perspectiva del sistema. Existe aún un caso más complejo que dice relación con el plan de cuentas que está configurado en el INFORMAT, bajo el cual INFORMAT genera balances y estados de resultados que no se condicen con la realidad actual de

CCK. CCK ha realizado modificaciones al plan de cuentas de hecho, pero no las ha traducido al sistema.

Todos los factores mencionados anteriormente, y fundamentalmente el último, supusieron un grado adicional de dificultad al proceso de determinación de las fuentes de datos, decidimos entonces medidas en varias direcciones:

A pesar de estas diferencias, que se solucionaron parcialmente (como se explicará en el siguiente punto), existía en INFORMAT la mayor parte de la información necesaria, y al grano más fino, esto permitía suponer que era factible la implementación de los requerimientos definidos por los usuarios, pero faltaba por determinar de dónde y cómo se extraería la información. Se manejaron dos alternativas:

- Solicitar el desarrollo de una aplicación por parte de INFORMAT, que permita a otra aplicación, desarrollada dentro del proyecto, realizar un conjunto de consultas predefinidas y extraer la información resultado de esas consultas.
- Utilizar como fuente de datos directa los archivos en formato plano, esto implica el desarrollo de aplicaciones que sean capaces de capturar la información contenida en estos archivos.

Por razones de costo y de simplicidad para CCK y considerando la falta de comunicación con INFORMAT, se escogió la segunda alternativa, que de cierta manera presentaba un desafío mayor desde el punto de vista de la implementación pero aseguraba un grado de independencia para nuestro proyecto con respecto a terceros.

2.3 Análisis y Validación de requerimientos.

Conocidas las características del negocio de CCK, los requerimientos de los usuarios y las características de los sistemas de información de CCK nos propusimos definir qué tareas eran necesarias de realizar previo al inicio del diseño e implementación del sistema. Y considerando lo crítico de los procesos ETL recomendamos la actualización en lo posible de la información contenida en el sistema INFORMAT de manera de que ésta fuera válida desde la perspectiva del negocio, estas modificaciones comprendían básicamente dos tareas principales:

- Remplazar todos aquellos parámetros obsoletos por parámetros utilizados actualmente en el sistema, esta tarea es completamente realizable al interior de CCK ya que INFORMAT permite la actualización de los parámetros por parte del usuario.
- Redefinir el plan de cuentas en el INFORMAT, esto significaba solicitar a INFORMAT que realizara los cambios de configuración necesarios ya que se requería la definición de nuevos asientos contables y esas opciones no estaban disponibles a nivel de usuario.

De los dos requerimientos descritos se logró que el primero se realizara, a pesar de significar una importante carga operativa, sin embargo el segundo requerimiento fue imposible de conseguir ya que nunca se recibió una respuesta oportuna de parte de INFORMAT.

2.4 Definición de Metas y Objetivos.

Dada la realidad de de CCK en cuanto a requerimientos e información disponible definimos en conjunto con la empresa abarcar en esta primera etapa el área de ventas y la información contable, principalmente porque estimamos que la implantación de los procesos ETL sería costosa en términos de tiempo y abarcar alguna otra área, podría ser demasiado ambicioso.

3. Diseño

3.1 Modelo Lógico.

A partir del análisis realizado en la etapa anterior comenzamos a desarrollar el diseño lógico del sistema.

Se identificaron las siguientes dimensiones:

- Clientes
- Productos
- Tiempo
- Vendedores
- Documentos
- Cuentas

Se definieron dos tablas de hecho:

- Ventas
- Movimientos Contable

A partir de esto se generaron dos modelos estrella, el primero representado a los movimientos contables (ver figura 19) y el segundo a las ventas (ver figura. 20)

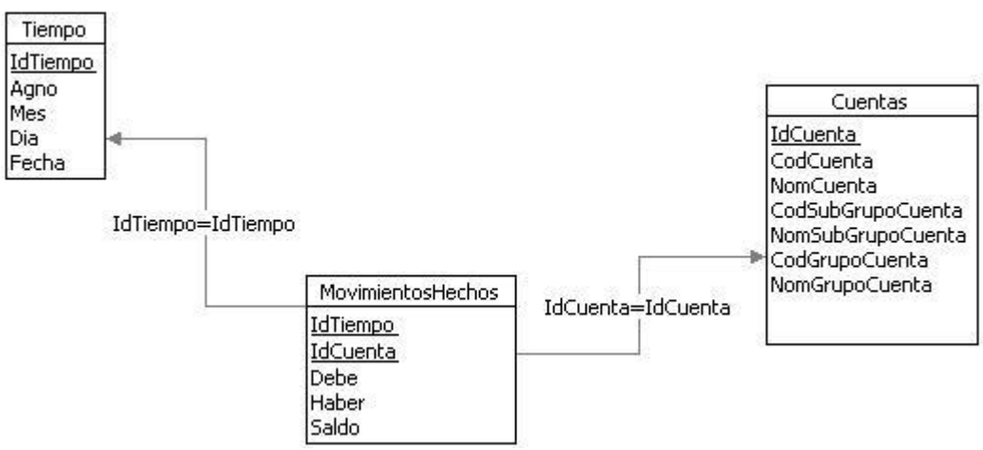


Figura 19: Modelo estrella Ventas.

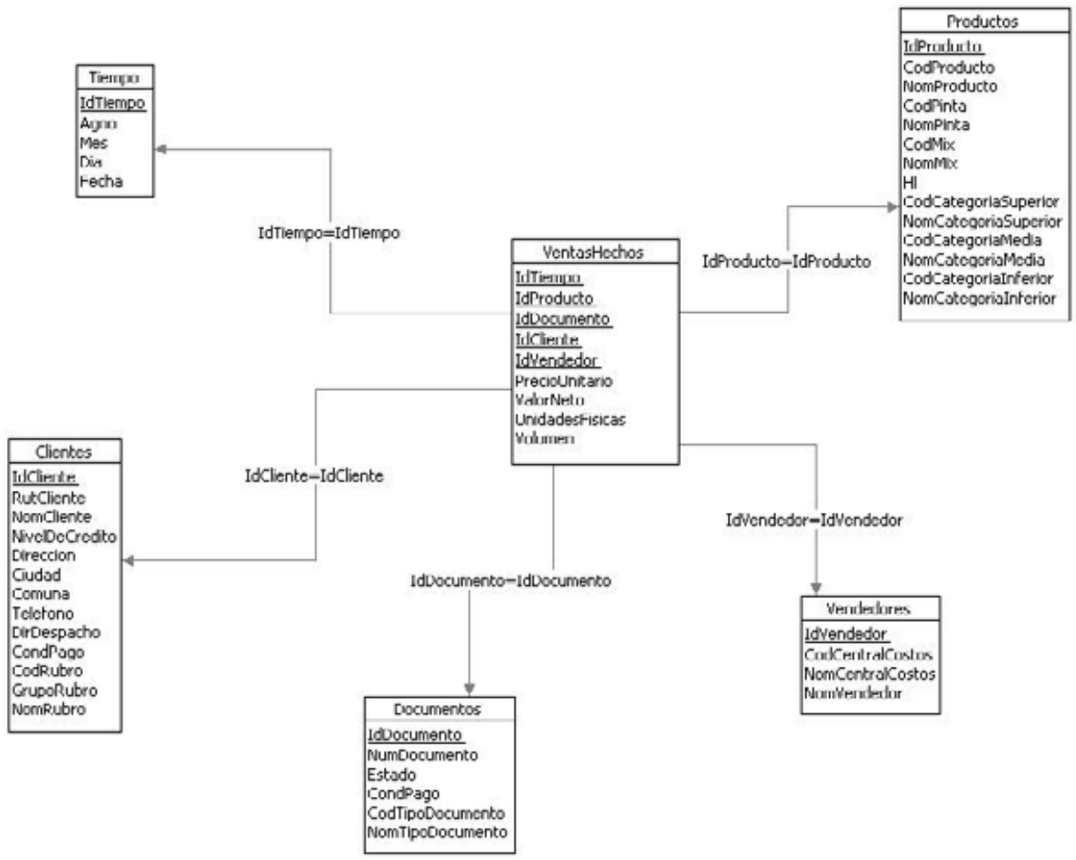


Figura 20: Modelo estrella Movimientos Contables.

A partir de ambos modelos se definieron las jerarquías (ver figura 21).

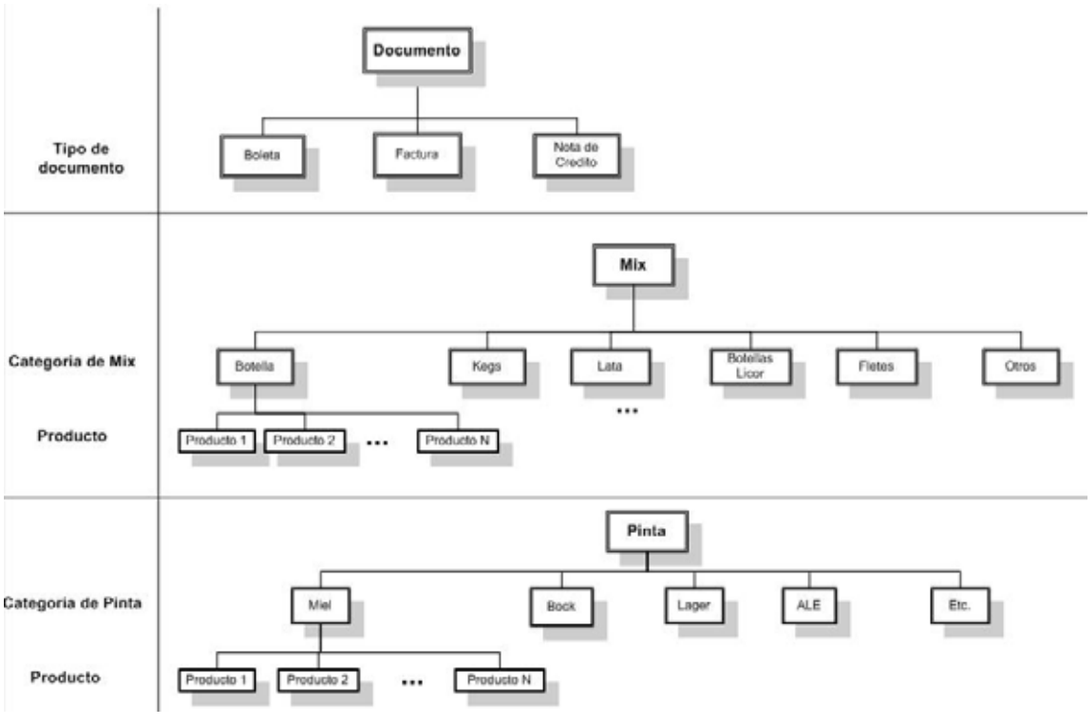


Figura 21: Ejemplo de Jerarquías.

3.1.2 Arquitectura del Data Warehouse.

Con los modelos presentados anteriormente, y teniendo en cuenta el resto de las áreas del negocio establecimos un conjunto de dimensiones como conformadas, declarándolas como parte de la ***arquitectura del data warehouse***, de esta manera aseguramos la escalabilidad del sistema con miras a la integración de otras áreas del negocio, pensando en una integración completa a largo plazo.

Las dimensiones declaradas conformadas son:

- Tiempo
- Productos
- Clientes
- Cuentas

Estas dimensiones extenderán su significado lógico a todo el sistema, por este motivo deben contener información del grano más fino dictado por su propia naturaleza. Para el caso de Clientes, este nivel será el cliente en sí, para el caso de Productos este nivel será el producto en sí, etc.

Cualquier modificación posterior a alguna de estas dimensiones se reflejará en todo el sistema.

En al tabla 7 consideramos la naturaleza de cada dimensión con miras a determinar cómo variarán en el tiempo.

Dimensión	Evolución
Tiempo	Sólo sufrirá cambios cada vez que sea necesario generar nuevos periodos de tiempo, inicialmente se considera cargar datos para al menos los próximos 4 años.
Cuentas	Prácticamente estática si se producen cambios estos reemplazaran a los datos actuales.
Clientes	Esta dimensión se espera cambie rápidamente, con la aparición gradual de nuevos clientes. Estos será agregados conforme generen movimientos.

Productos	Cambiará muy lentamente, CCK no esta avocada a la generación constante de nuevos productos. Cada nuevo producto será agregado al sistema cada vez que genere un movimiento.
Vendedores	Cambiará muy lentamente y es probable que en el tiempo esta dimensión se detenga, si CCK decide distribuir sólo a través de CCU.
Documentos	Esta dimensión cambiará rápidamente pues contiene información de cada documento de venta transado, deberá ser recargada en cada proceso de carga.

Tabla 7: Evolución de las dimensiones.

3.1.3 Agregaciones.

La existencia o no de agregaciones tiene directa relación con los tiempos de respuesta y los volúmenes de datos que maneja el sistema, para el caso de CCK, los volúmenes de datos son relativamente pequeños a los ojos de un RDBMS por tanto no se consideró la implementación de agregaciones iniciales. Sin embargo, si con el paso del tiempo los volúmenes de datos crecieran a niveles que afectaran el rendimiento entonces, como parte de las tareas de mantención, se deberán generar las agregaciones necesarias.

3.2 Diseño de los Proceso ETL.

El diseño de los proceso ETL fue especificado de manera gráfica utilizando la técnica propuesta en la metodología. Las extracciones se iniciarían a partir de diferentes archivos de textos obtenidos del sistema INFORMAT los que serían analizados en busca de la información necesaria, la cual sería transferida a un área de datos intermedia DSA relacional, el modelo del DSA se presenta en la figura 22.

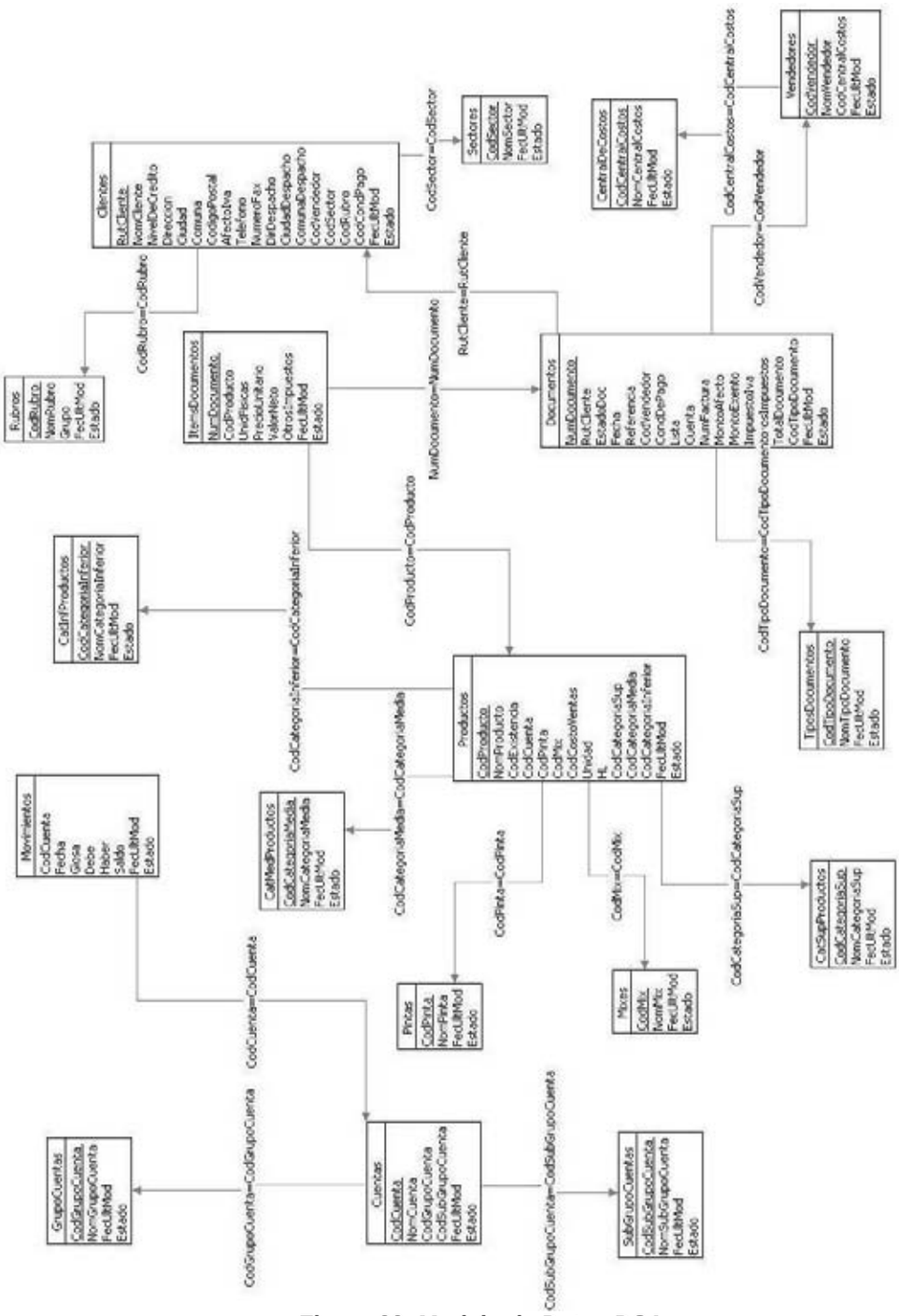


Figura 22: Modelo de Datos DSA.

Todos los procesos ETL siguen la lógica señalada en la figura 23.

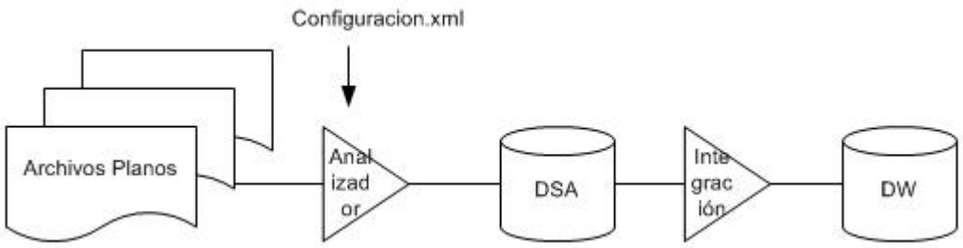


Figura 23: Procesos ETL desde un punto de vista global.

En esta figura describimos que los archivos Planos serán analizados a partir de un archivo de configuración XML, una vez analizados los datos rescatados serán almacenados en el DSA. Se escogió un DSA relacional para asegurar la integridad referencial de los datos. Del DSA los datos serán finalmente cargados en el data mart mediante una función de integración que se encargará de seleccionar la información adecuada desde el DSA.

A continuación presentamos el diagrama de arquitectura completo (ver figura 24) para la carga de la dimensión Cuentas a partir del archivo plano con la información del plan de cuentas.

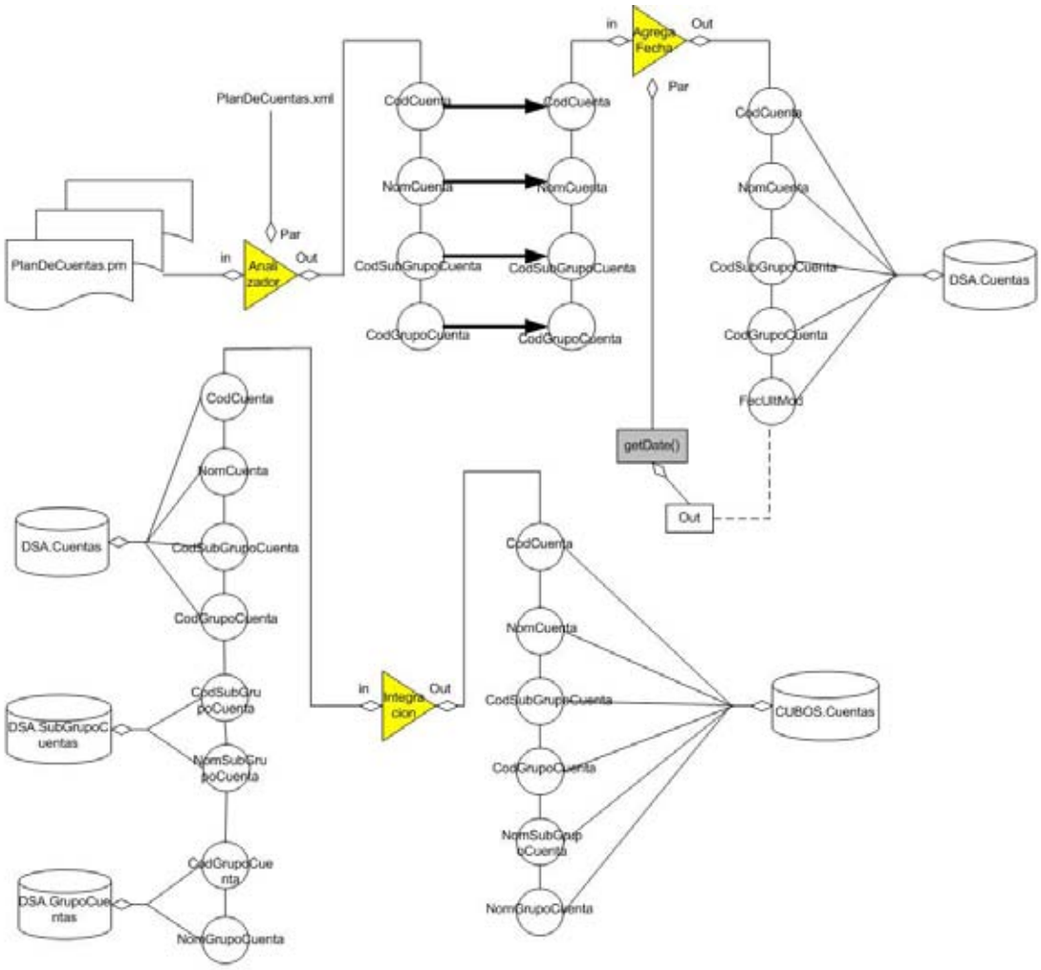


Figura 24: Diagrama de Arquitectura ETL carga cuentas.

Los demás diagramas de arquitectura son similares y se incluyen en el anexo C.

Dado lo especial de los procesos ETL que debemos construir se decidió codificar el proceso ETL completo, los archivos planos desde los cuales se extrajo la información estaban todos organizados de diferentes maneras, eso significa que se debió analizar cada archivo bajo diferentes reglas. En el próximo capítulo explicaremos con mayor detalle las características específicas de la implementación.

4. Implementación.

La implementación de un sistema que cumpla con las características establecidas en la etapa de diseño, es el objetivo de esta etapa. En el capitulo 4 de la primera parte (Implantación) de la metodología establecimos algunas buenas prácticas y consideraciones útiles a la hora de enfrentar esta tarea; sin embargo, en todo momento conservamos una perspectiva independiente de las herramientas de software.

4.1 Implementación física.

Escogidas las herramientas, comenzamos la implementación física del sistema transfiriendo los modelos de datos lógicos a la base de datos física. Para el nombramiento de los objetos de la base de datos utilizamos el estándar de nombres establecido en el anexo A, los modelos de datos físicos que incluyen los tipos de datos seleccionados están disponibles en el anexo B que incluye modelo de datos dimensional, el modelo de datos del DSA y adicionalmente un esquema que almacenará datos de usuarios, privilegios y preferencias. Además, se determinó la creación de índices adicionales en columnas que probablemente serán consultadas a menudo. Postgresql soporta cuatro tipos de índices: b-tree, r-tree, hash y gist. Con respecto a los rendimientos de estos índices es conocido que b-tree tiene un rendimiento superior a hash, en cuanto a gist éste es una implementación extensible, es decir permite al usuario la definición de índices personalizados, gist actúa como una plantilla sobre la cual se puede especificar un índice personalizado. Los índices b-tree mejoran su rendimiento si los datos se encuentran ordenados físicamente, esto se logra mediante el comando CLUSTER.

Se crearon índices b-tree en todas las claves primarias de cada dimensión así como índices compuestos sobre las claves primarias de las tablas de hechos, estableciendo como primer término del índice la columna IdTiempo, adicionalmente se crearon índices en campos de acceso frecuente como se señala en la tabla 8:

Tablas	**Índice**
Clientes	Idx_Rubro (Rubro)
Documentos	Idx_CodTipoDocumento (CodTipoDocumento)
Vendedores	Idx_CodCentralCostos

	(CodCentralCostos)
Productos	Idx_CodMix (Mix) Idx_CodPinta (Pinta)
Cuentas	Idx_CodSubGrupoCuenta (CodSubGrupoCuenta)
MovimientosHechos	Idx_IdTiempo (IdTiempo)
VentasHechos	Idx_IdTiempoIdProducto (IdTiempo, IdProducto)

Tabla 8: Índices creados excluyendo claves primarias.

PostgreSql requiere la ejecución de procesos de mantención periódicamente para su óptimo funcionamiento. Considerando esto, se programarán tres tareas de mantención que se ejecutarán diariamente (sólo si se han producido cargas)

- Análisis y recolección de basura, mediante el comando VACUUM ANALYZE (elimina físicamente los registros obsoletos y eliminados de la base de datos, y analiza cada tabla para la optimización).
- Mediante el comando CLUSTER se forzará la indexación física de los principales índices lógicos.
- Respaldo mediante el comando pg_dump.

Estas tareas deben ejecutarse de manera exclusiva, esto significa que durante el tiempo de duración de estas operaciones, el sistema se mantendrá no disponible.

El servidor OLAP requiere la especificación lógica de los esquemas y tablas que formarán los cubos de datos, así como las jerarquías y miembros de éstas que forman las dimensiones. Como mencionamos anteriormente en Mondrian esta definición se hace a través de un archivo de esquema en XML, parte de este archivo se muestra en la figura 25.

A partir de este esquema el servidor OLAP es capaz de traducir las consultas del usuario a SQL y ejecutarlas contra el RDBMS.

Para optimizar el rendimiento en producción del servidor se configurarán los niveles de información de log sólo a nivel de error.

Para Mondrian la cantidad de memoria disponible para la JVM (Java Virtual Machine) es importante, ya que es ahí donde se almacenan las agregaciones que optimizan el rendimiento del servidor. Con este objetivo se configuró la memoria de la JVM, estableciendo 512MB de memoria física y 768MB para los peak.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="kunstmann">
  <Dimension name="Tiempo">
    <Hierarchy hasAll="false" allMemberName="Todos" primaryKey="IdTiempo">
      <Table schema="CUBOS" name="Tiempo" />
      <Level column="Agno" name="Agno" type="Numeric" levelType="TimeYears" />
      <Level column="Mes" name="Mes" type="Numeric" levelType="TimeMonths" />
      <Level column="Dia" name="Dia" type="Numeric" levelType="TimeDays" />
    </Hierarchy>
  </Dimension>
  <Dimension foreignKey="IdCuenta" name="Cuenta">
    <Hierarchy hasAll="true" allMemberName="Todas las Cuentas" primaryKey="IdCuenta">
      <table schema="CUBOS" name="Cuentas" />
      <level column="CodGrupoCuenta" name="Grupo">
        <Property name="Nombre Cuenta" column="NomGrupoCuenta" />
      </level>
      <level column="CodSubGrupoCuenta" name="SubGrupo">
        <Property name="Nombre Cuenta" column="NomSubGrupoCuenta" />
      </level>
      <level column="CodCuenta" name="Cuenta">
        <Property name="Nombre Cuenta" column="NomCuenta" />
      </level>
    </Hierarchy>
  </Dimension>
  <Cube name="Balance">
    <table schema="CUBOS" name="MovimientosHechos" />
    <DimensionUsage name="Tiempo" source="Tiempo" foreignKey="IdTiempo" />
    <DimensionUsage name="Cuenta" source="Cuenta" foreignKey="IdCuenta" />
    <Measure name="Debe" column="Debe" aggregator="sum" formatString="#,#" />
    <Measure name="Haber" column="Haber" aggregator="sum" formatString="#,#" />
  </Cube>
```

Figura 25: Esquema XML utilizado por el servidor OLAP.

4.3 Implementación ETL.

Para la implementación de los procesos ETL nos propusimos la integración de de herramientas de workflow en el proceso y definimos un proceso genérico en Bonita (ver figura 26)

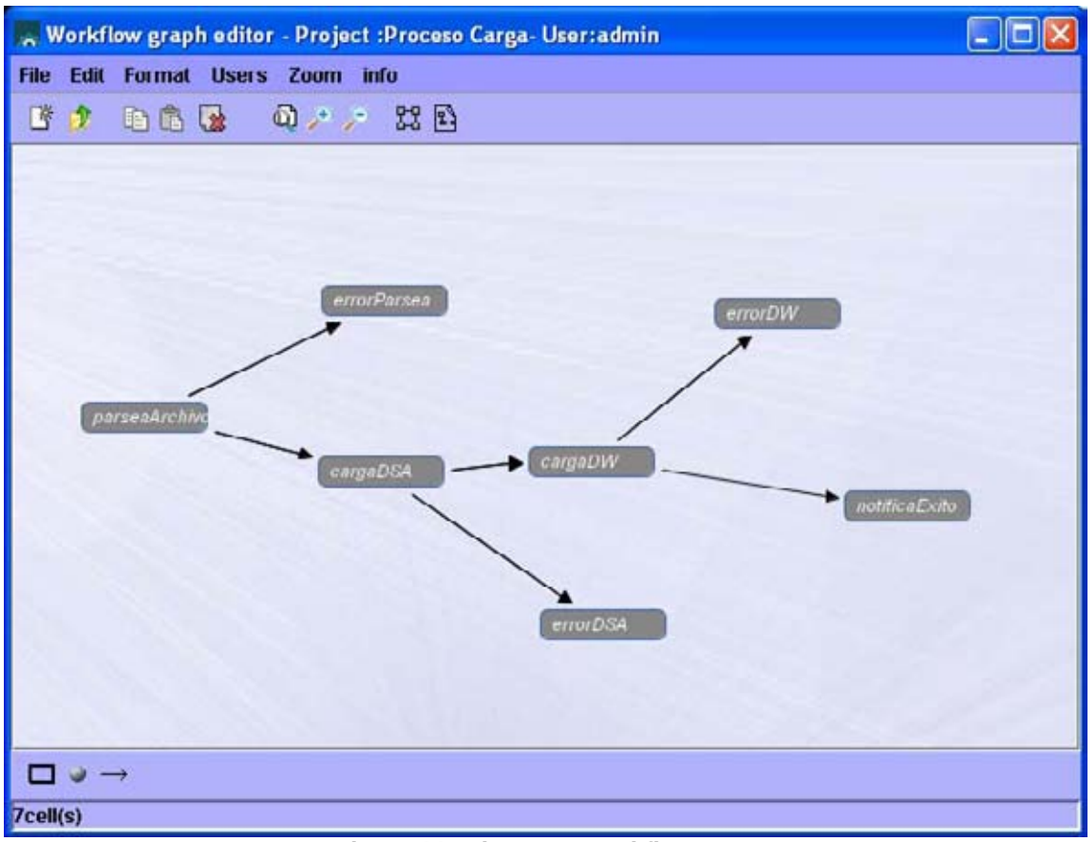


Figura 26: Diagrama workflow ETL.

En la etapa “parseaArchivo” el sistema deberá leer el archivo de entrada y reconocer la información válida que en él se incluye. El analizador de archivos debe lidiar con una complicación adicional que surge debido a que los formatos de los archivos varían, dependiendo del reporte. Como solución, se codificó una aplicación capaz de leer datos desde un archivo plano, según reglas establecidas en un archivo XML.

Este archivo de configuración permite la definición lógica de la estructura de la información que incluye el archivo plano específico. Para esto, el archivo de configuración permite múltiples parámetros que van desde la posición y largo

del dato esperado, hasta la especificación de expresiones regulares que permitan la detección de datos complejos.

Un ejemplo de un archivo descriptor XML se presenta en la figura 27

```
<parser>
  <header className="kunstmann.movimientos.HeaderMovimientos" endAt="Hasta">
    <line>
      <elem name="Desde" length="10" trim="true" startWith="Desde el" regex="\d{2}\/\d{2}\/\d{4}$" />
      <elem name="Hasta" length="10" trim="true" startWith="Hasta el" regex="\d{2}\/\d{2}\/\d{4}$" />
    </line>
  </header>
  <file className="kunstmann.movimientos.Movimientos" endAt="CodCuenta">
    <line>
      <elem name="CodCuenta" length="8" trim="true" startAt="0" regex="\d{2}\.\d{2}\.\d{2}$" />
      <elem name="SaldoInicial" length="17" trim="true" startAt="115" regex="" />
    </line>
    <line>
      <elem name="CodCuentaDos" length="8" trim="true" startWith="*** TOTAL CUENTA :" regex="\d{2}\.\d{2}\.\d{2}$" />
      <elem name="TotalDebe" length="16" startAt="80" trim="true" regex="" />
      <elem name="TotalHaber" length="17" startAt="97" trim="true" regex="" />
      <elem name="TotalSaldo" length="16" startAt="115" trim="true" regex="" />
    </line>
    <object className="kunstmann.movimientos.Movimiento" endAt="Fecha" methodName="addMovimiento">
      <line>
        <elem name="Fecha" length="5" trim="true" startAt="0" regex="\d{2}\-\d{2}$" />
        <elem name="Glosa" length="19" startAt="43" trim="true" regex="" />
        <elem name="Debe" length="16" startAt="80" trim="true" regex="" />
        <elem name="Haber" length="17" startAt="97" trim="true" regex="" />
        <elem name="Saldo" length="17" startAt="115" trim="true" regex="" />
      </line>
    </object>
  </file>
</parser>
```

Figura 27: Ejemplo archivo de configuración para el analizador.

El analizador genera arreglos de objetos con la información que ha capturado, la definición de las clases a las que pertenecerán esos objetos deben ser especificadas por el programador, como parte de la configuración.

Adicionalmente, el analizador tiene la capacidad de realizar verificaciones especiales definidas por el programador. Por ejemplo, para el caso de los balances, se verifican todos los totales por cuenta contra la suma de los valores capturados desde el archivo, esto con el objetivo de asegurar que la información está 100% correcta. En caso de detectarse un error, el flujo pasa a la etapa “errorParsea” donde el usuario es notificado del error y el modo de solucionarlo. Esto se logra gracias a que el sistema es capaz de diferenciar sobre un conjunto de errores posibles, ya sea, desde errores en el formato del archivo, hasta la necesidad de cargar otro archivo previamente (por ejemplo si aparece un nuevo cliente). Una vez corregido el error, el usuario deberá iniciar un nuevo proceso de carga.

Si la información es válida, ésta será almacenada en el DSA en la etapa “cargaDSA”. Este proceso consiste en almacenar los objetos generados por el analizador de manera persistente. Nuevamente si existe algún error el usuario es notificado en la etapa “errorDSA”.

La etapa “cargaDW” se encarga de transferir la información desde el DSA al data mart, y posteriormente vaciará el cache generado por Mondrian de manera de forzar al servidor OLAP a recargar nuevamente los datos (dado que éstos han cambiado) desde el RDBMS. Finalmente en la etapa “notificaExito” notificamos al usuario del éxito del proceso de carga y notificamos que el sistema se encuentra disponible incluyendo la nueva información recientemente cargada.

5. Implantación.

El plan de implantación final que se desarrolló para CCK incluía básicamente tres aspectos: capacitación, mantención y crecimiento, cada una de las cuales explicamos en adelante.

5.1 Capacitaciones.

Con el objetivo de educar a los usuarios con respecto a las características del nuevo sistema, proponemos la programación de un curso de capacitación en el cual participen todos los potenciales usuarios del sistema.

El curso deberá dividirse en dos sesiones. Una primera sesión tendrá el objetivo de educar a los usuarios con respecto a los contenidos que incluye el sistema y a cómo éstos se organizan. La segunda sesión estará orientada a capacitar a los usuarios en el uso de la herramienta para acceder a los datos, específicamente JPivot.

El proceso de capacitación contempla también durante las primeras semanas sesiones personalizadas con aquellos usuarios que requieran mayor atención.

5.2 Mantencion.

Las tareas de mantención del sistema tienen como objetivo conservar la calidad de la información publicada por el sistema, así como conservar los tiempos de respuesta dentro de un rango aceptable.

Para realizar estas tareas consideramos necesario la contratación de una persona con conocimientos en administración de bases de datos, de manera que se haga responsable de ejecutar todas las tareas de mantención que el RDBMS requiere. Además, conforme el volumen de los datos aumente, se deberán realizar tareas de afinamiento sobre los parámetros de la base de datos.

La definición de agregaciones es un punto que al largo plazo puede presentarse como necesario. Como mencionamos con anterioridad, PostgreSql no soporta agregaciones de manera nativa; sin embargo, esta realidad puede cambiar en el corto plazo (es probable que se incluya en las próximas versiones), y de no ser así, existen artículos que proponen soluciones alternativas.

Consideramos que los volúmenes de datos que CCK maneja no deberían en el mediano plazo ser problema para el RDBMS. Por ejemplo, para las tablas de hechos proyectamos linealmente los volúmenes presentados en la tabla 9.

Tabla	2004	2005	2006	2007
MovimientosHechos	35000	70000	105000	140000
VentasHechos	24000	48000	72000	96000

Tabla 9: Numero de registros esperados para las tablas de hechos por año.

Aún siendo en extremo pesimista, es muy poco probable superar los 500.000 registros para el año 2007 y aun si esto sucediera el volumen de datos es completamente manejable por el RDBMS con rendimientos aceptables.

5.3 Crecimiento.

Con miras a soportar el crecimiento mediante la integración de la información de otras áreas del negocio, proponemos a CCK conservar el vínculo con la Universidad Austral y específicamente con el Instituto de Informática, para que este último asesore a CCK en los procesos que involucren el crecimiento del sistema.

Conclusiones.

El valor de la información como soporte de la gestión de calidad no reside en la información por sí misma, sino en la capacidad de análisis que tengamos sobre ésta. La gestión de calidad surge de la capacidad de tomar las decisiones adecuadas basadas en información útil, oportuna y correcta.

Las fuentes de información que soportan las tareas operacionales de una empresa no soportan los procesos de toma de decisiones, pues no están diseñadas con ese objetivo. Las organizaciones deben introducir nuevas tecnologías de información que les permitan integrar sus fuentes de información operacionales en sistemas de información orientados a la gestión.

Las PYMEs no son distintas al resto de las organizaciones, generan un volumen de información operacional menor, pero su necesidad de gestión de calidad es igual; sin embargo, en general, las PYMEs desconocen o incluso a veces menosprecian el valor agregado que genera el manejo correcto de la información sobre sus procesos de toma de decisiones, y por lo tanto no realizan los esfuerzos necesarios para mejorar sus procesos de toma de decisiones mediante la introducción de nuevas tecnologías de información.

Durante nuestro trabajo observamos que los principales factores que impiden a las PYMEs mejorar la calidad de su gestión mediante la introducción de nuevas tecnologías de información son:

- **Confidencialidad de la información:** Las PYMEs son celosas de su información operacional. Para muchas, éste es su principal activo y no están dispuestas a compartirlo con asesores externos.
- **Visión estrecha:** En general los mandos medios y superiores en las PYMEs no visualizan el impacto que produce la mejora en la

calidad de la gestión producto de la introducción de nuevas tecnologías de información. Las PYMEs no consideran que la introducción de nuevas tecnologías de información sea crítica para el crecimiento de su empresa y son reacios a abordar proyectos de mediano a largo plazo.

- **Alto costo de la tecnología:** Sumado a lo anterior, los productos de software asociados a la gestión tienen un alto costo de licenciamiento, básicamente porque están enfocados a otro segmento de empresas.
- **Alto costo de asesorías:** Los proceso de análisis, diseño e implantación de sistemas de gestión requieren un avanzado nivel de asesoría para su éxito, cuyo costo es alto. Sin embargo, en el último tiempo el gobierno está desarrollando programas y franquicias tributarias para reducir estos costos.

De los puntos mencionados anteriormente, consideramos que los dos primeros tienen directa relación con las características y mentalidad de las PYMEs, mientras, que los últimos dos tienen relación con el mercado.

La metodología propuesta se enfoca en reducir los costos producto del mercado.

En primer lugar la metodología propuesta estable un marco teórico que permite desarrollar un proyecto de data warehouse en una PYME, aportando al conocimiento general sobre el tema, desde un punto de vista específico. El seguimiento de la metodología propuesta reduce el riesgo al fracaso y establece mecanismos formales para conseguir los resultados deseados.

En segundo término la metodología propone un conjunto de herramientas integrables, que eliminan el costo de licenciamiento y que proveen de las

funcionalidades necesarias para construir un sistema data warehouse en cada uno de sus niveles. Creemos que las herramientas propuestas son factibles de aplicar integradas o individualmente en una empresa PYME.

En específico también concluimos que la integración de herramientas de workflow en los procesos de extracción aporta a la transparencia y escalabilidad de estos mismos, reduciendo la necesidad del uso de herramientas comerciales.

Consideramos el resultado de nuestra experiencia en la empresa Compañía Cervecera Kunstmann S.A. como positivo pues se lograron los objetivos propuestos y además se logró crear un vínculo entre la Universidad Austral de Chile y la empresa, beneficiosa para ambas instituciones y que asegura al menos en el mediano plazo la continuación y creación de éste y otros proyectos. Creemos que la Universidad puede hacer grandes aportes al desarrollo tecnológico de las PYMEs acercando las tecnologías a éstas y motivándolas a innovar y crecer.

Por último consideramos que las PYMEs deben valorar en su justa medida la importancia de la gestión y el impacto de las tecnologías de información en ésta, de manera de realizar todos los esfuerzos necesarios para mejorar sus procesos de toma de decisiones, que en el mediano y largo plazo redundarán en beneficios directos para la organización.

Referencias.

[Ale, 04] Jaime Alée: “Uso de TIC en las empresas” Asociación Chilena de Empresas de Tecnologías de Información

[Nis, 2003] Gary Nissen: “Is Hand-Coded ETL the Way to Go? Absolutely yes, or absolutely no, depending...” by Gary Nissen edited by Ralph Kimball May 31, 2003. http://www.intelligententerprise.com/030531/609warehouse1_2.jhtml

[Kim, 95] Kimball: “The Aggregate Navigator (11/1995) How to optimize your data warehouse using aggregates without driving your end users crazy” http://www.dbmsmag.com/9511d05.html

[Kim, 96] Kimball: “Aggregate Navigation With (Almost) No Metadata” By Ralph Kimball DBMS Data Warehouse Supplement, August 1996 http://www.dbmsmag.com/9608d54.html

[Kim, 98] Kimball: “Is Data Staging Relational? Or does it have more to do with sequential processing?”.

[Kim+, 02] Ralph Kimball, Margy Ross: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. 2002.

[Bac+,01] Torben Bach, Christian S. Jansen 2001 “Multidimensional Database Technology”.

[Bon+,01] Bonifati, Angela; Casati, Fabio; Dayal, Umesh; Shan, Ming-Chien: “Warehousing Workflow Data: Challenges

and Opportunities", Politecnico di Milano, 2001.

[Bou+,99] Bouzeghoub, Mokrane; Fabret, Francoise; Matulovic-Broqué, Maja: "Modelling Data Warehouse refreshment process as a workflow application", Proceedings of the International Workshop on Design and Management of Data Warehouses, junio, 1999.

[Kim, 01] Kimball: "What Not to Do. Dimensional modelling mistakes to guard against".

[Gar, 04] Gardner 2004: "Materialized Views in PostgreSQL" http://jonathangardner.net/PostgreSQL/materialized_views/matviews.html

[Bit, 02] Bitam: "Acerca de Business Intelligence". http://www.bitam.com/spanish/AcercaDeBI.htm

[Vas+, 02] Modeling ETL Activities as Graphs. Panos Vassiliadis, Alkis Simitsis, Spiros Skiadopoulos

[Hil+, 01] Charles W. L. Hill / Gareth R. Jones: "Administración Estratégica" 2001, 3era edición.

[Rob, 03] Cynthia Robert 2003, gerente general de FUNDES Chile: "La brecha tecnológica de las PYMES".

[Nad, 04] Javier Nadar 2004: "Sistema de apoyo gerencial Universitario".

[Wol, 2002] Carmen Gloria Wolff: "La tecnología data warehouse".

[Del, 01] Kalen Delaney: "Inside Microsoft Sql Server 2000"

[URL 1] e-workflow - then workflow portal. www.e-workflow.org

[URL 2] JUnit.org.
www.junit.org

[URL 3] Data Transformation Services (DTS) in Microsoft SQL Server 2000.
http://msdn.microsoft.com/SQL/sqlwarehouse/DTS/default.aspx?pull=/library/en-us/dnsql2k/html/dts_overview.asp

[URL 4] Materialized View Concepts and Architecture
http://www.stanford.edu/dept/itss/docs/oracle/9i/server.920/a96567/repmview.htm

[URL 5] Open Source Workflow Engines Written in Java
http://www.manageability.org/blog/stuff/workflow_in_java/view

[URL 6] WfMOpen Workflow.
http://wfmopen.sourceforge.net/

[URL 7] Bonita Cooperative Workflow.
http://bonita.objectweb.org/

[URL 8] Bonita Success Stories.
http://bonita.objectweb.org/html/Integration/index.html

[URL 9] Object Web Open Source Middleware.
Http://www.objectweb.org

[URL 10] Institut national de recherche en informatique et en automatique.
http://www.inria.fr/

[URL 11] The Workflow Management Coalition
http://www.wfmc.org/

[URL 12] eclipse.org

http://www.eclipse.org

[URL 13] José Miguel Piquer, Universidad de Chile

http://www.emb.cl/gerencia/articulo.mv?sec=14&num=97

[URL 14] http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

[URL 15] http://java.sun.com/products/ejb/docs.html

[URL 16] msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/olapmultidimensional_expressions_overview.asp

[URL 17] http://publib.boulder.ibm.com/infocenter/ids9help/index.jsp?topic=/com.ibm.adref.doc/adrefmst229.htm

Anexos

Anexo A: Documento estándar de nombres objetos base de datos

Para el nombre de los objetos se utilizará el formato:

ClaseCuanticadorPrincipal

Se utilizarán las siguientes abreviaciones definidas en la tabla 1:

Abreviación	Palabra Abreviada
Cat	Categoría
Cod	Código
Fec	Fecha
Inf	Inferior
Mod	Modificación
Med	Media
Nom	Nombre
Sup	Superior
Ult	Ultima Modificación

Tabla1: Abreviaciones

Para el nombramiento de claves he índices se utilizaran el siguiente formato:

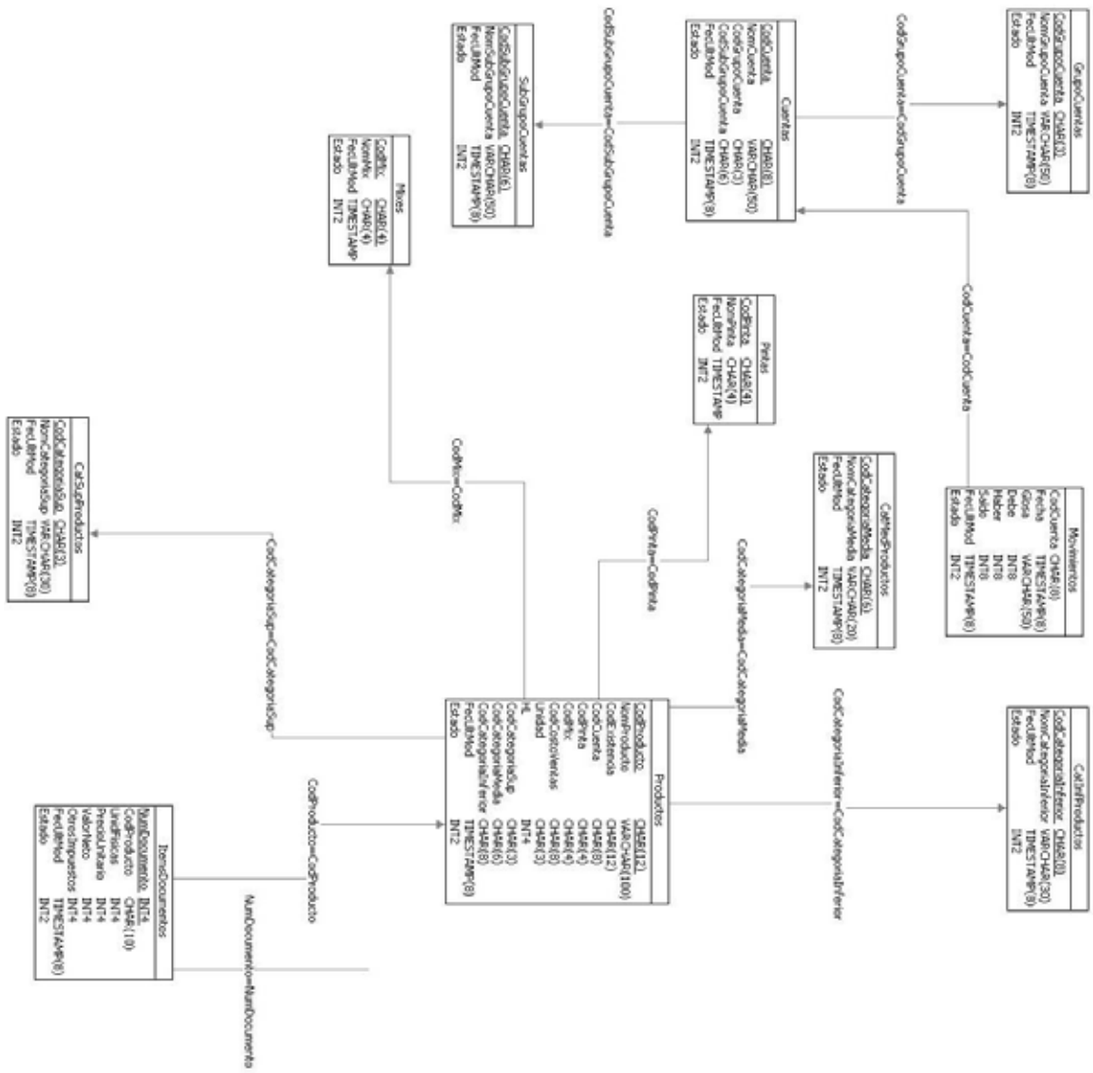
TipoClave_NombreObjetoAfectado1_NombreObjetoAfectado2 …

Se utilizaran las siguientes abreviaciones para los tipos de claves:

Abreviación	Palabra Abreviada
pk	Primary key
fk	Foreign Key
uk	Unique key
idx	Index

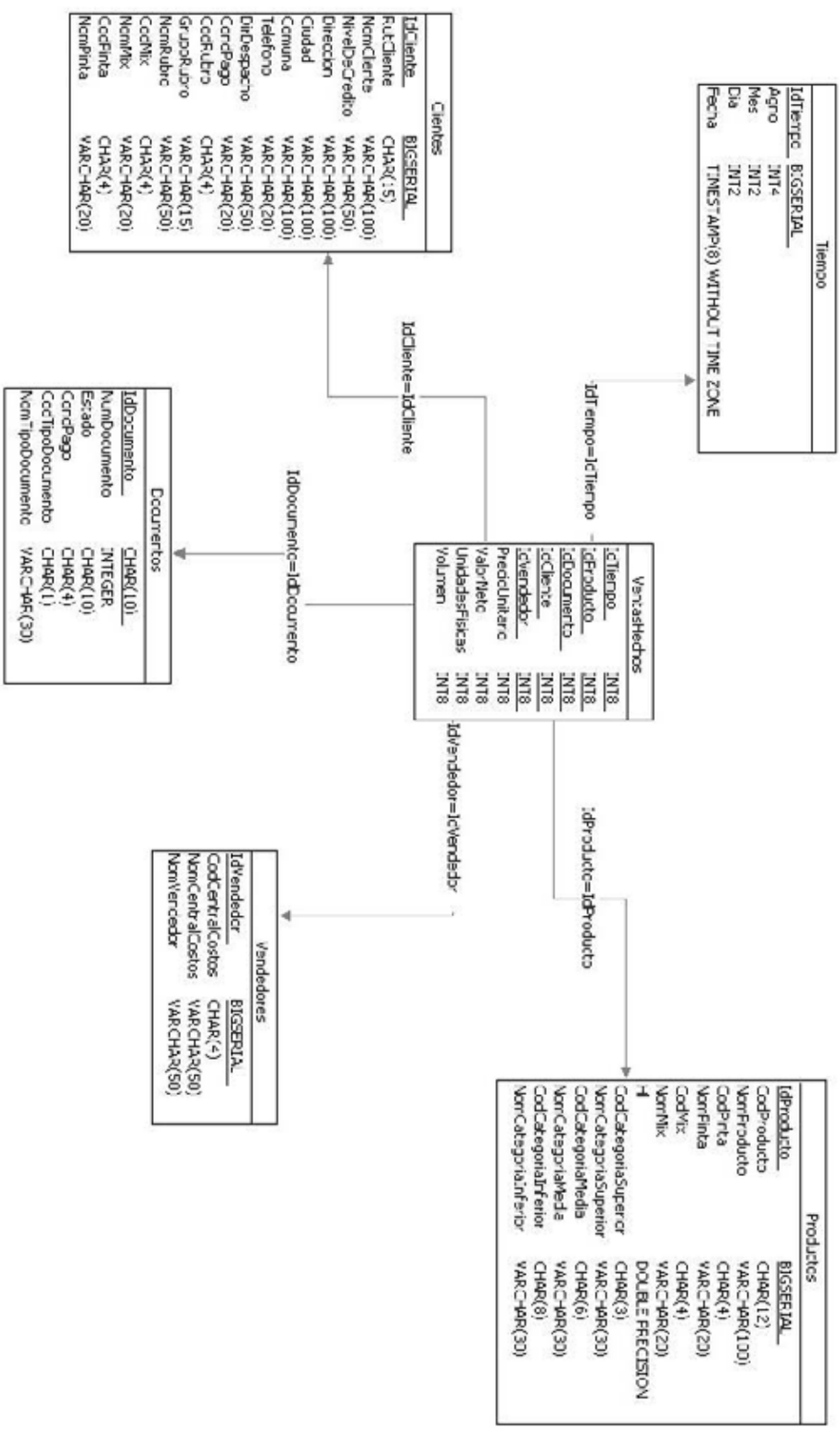
Anexo B: Modelos de Datos.

Modelo de datos DSA.

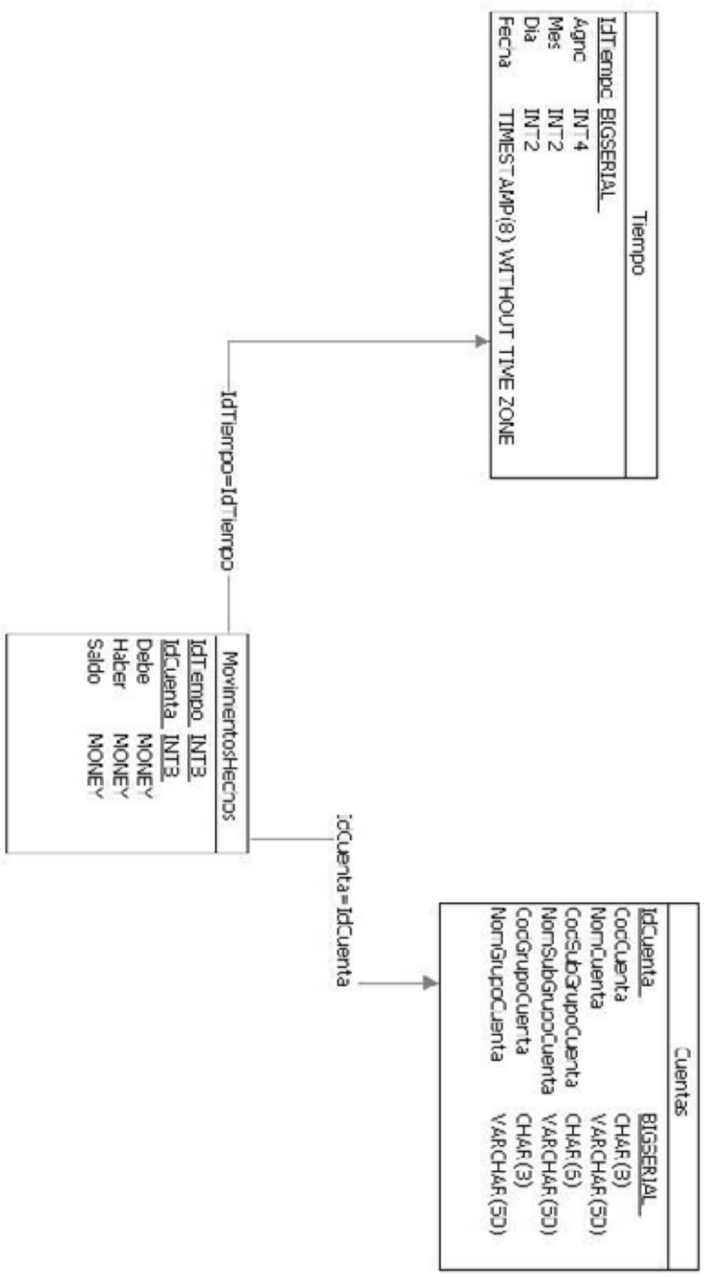


Documentos
NumDocumento INT4
RutCliente CHAR(15)
EstadoDoc CHAR(10)
Fecha TIMESTAMP(8)
Referencia INT4
CodVendedor CHAR(6)
CondDePago CHAR(4)
Lista VARCHAR(25)
Cuenta CHAR(8)
NumFactura INT4
MontoAfecto INT4
MontoExento INT4
ImpuestoIva INT4
OtrosImpuestos INT4
TotalDocumento INT4
CodTipoDocumento CHAR(1)
FecUltMod TIMESTAMP(8)
Estado INT2
Clientes
RutCliente CHAR(15)
NomCliente VARCHAR(100)
NivelDeCredito INT4
Direccion VARCHAR(100)
Ciudad VARCHAR(100)
Comuna VARCHAR(100)
CodigoPostal VARCHAR(10)
Afectoiva CHAR(2)
Telefono VARCHAR(15)
NumeroFax VARCHAR(15)
DirDespacho VARCHAR(100)
CiudadDespacho VARCHAR(100)
ComunaDespacho VARCHAR(100)
CodVendedor CHAR(4)
CodSector CHAR(4)
CodRubro CHAR(4)
CodCondPago CHAR(4)
FecUltMod TIMESTAMP(8)
Estado INT2
Vendedores
CodVendedor CHAR(6)
NomVendedor VARCHAR(30)
CodCentralCostos CHAR(4)
FecUltMod TIMESTAMP(8)
Estado INT2
CentralDeCostos
CodCentralCostos CHAR(4)
NomCentralCostos VARCHAR(30)
FecUltMod TIMESTAMP(8)
Estado INT2
Rubros
CodRubro CHAR(4)
NomRubro VARCHAR(30)
Grupo VARCHAR(15)
FecUltMod TIMESTAMP(8)
Estado INT2
TiposDocumentos
CodTipoDocumento CHAR(1)
NomTipoDocumento VARCHAR(30)
FecUltMod TIMESTAMP(8)
Estado INT2
Sectores
CodSector CHAR(4)
NomSector VARCHAR(30)
FecUltMod TIMESTAMP(8)
Estado INT2
NumDocumento=NumDocumento
CodVendedor=CodVendedor
CodCentralCostos=CodCentralCostos
CodTipoDocumento=CodTipoDocumento
RutCliente=RutCliente
CodRubro=CodRubro
CodSector=CodSector

Modelo de Datos Cubo Ventas.



Modelo de Datos Cubo Movimientos.

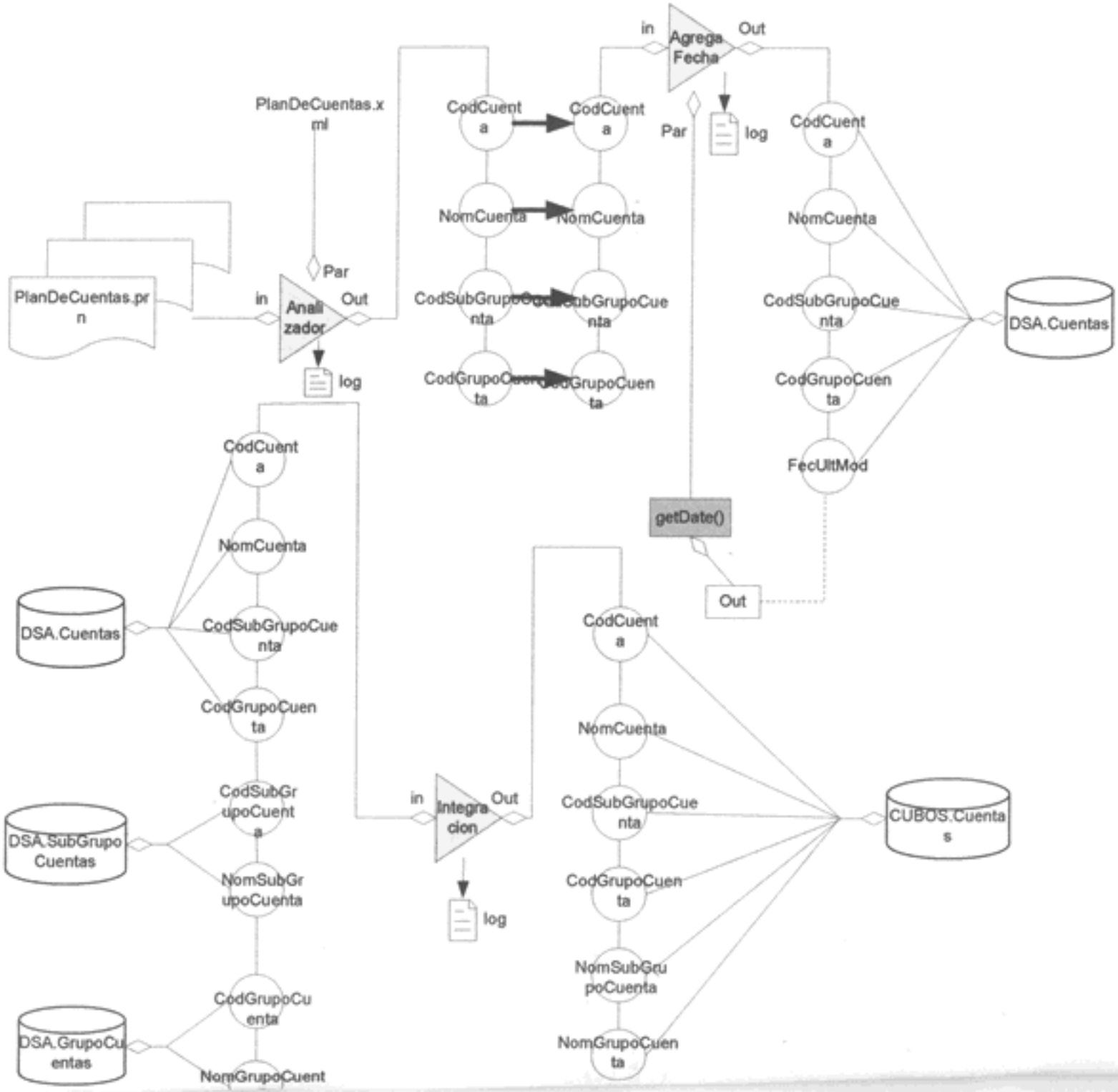


Anexo C: Diagramas de Arquitectura ETL.

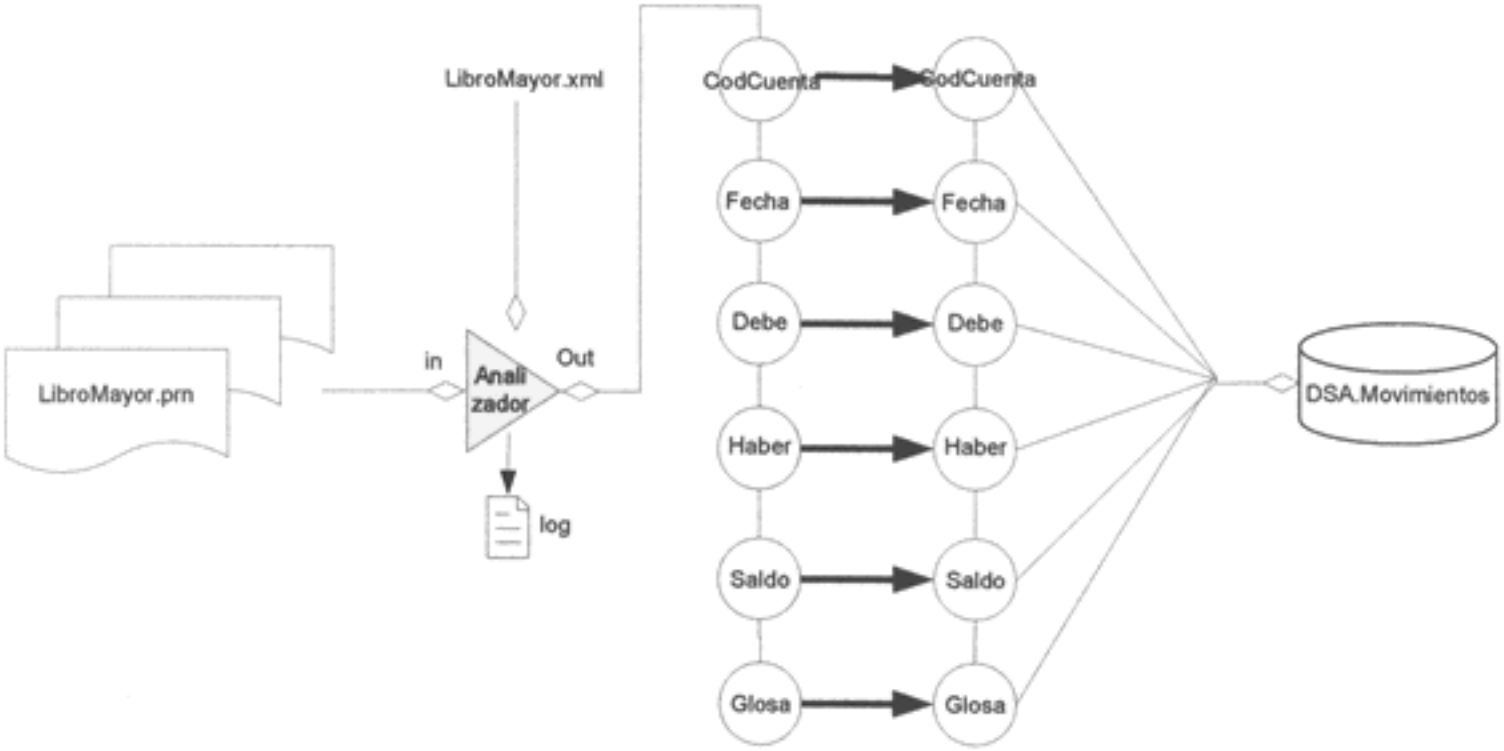
Tabla de diagramas de arquitectura.

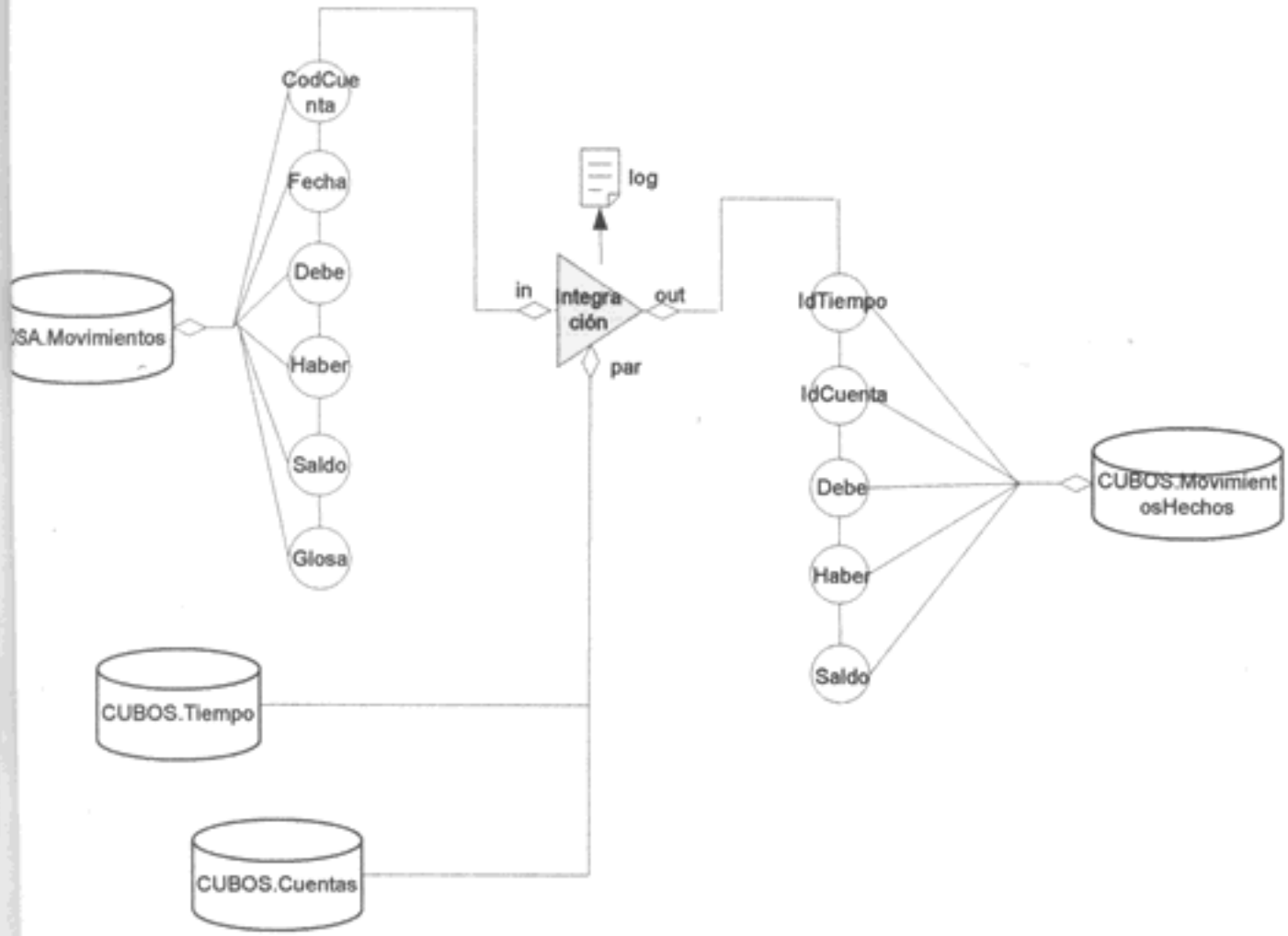
Nombre Diagrama	Origen	Destino	Descripción
ETLCuentas	PlanDeCuentas.prn	CUBOS.Cuentas	Carga la dimensión Cuentas utiliza las tablas Cuentas, SubGrupoCuentas y GrupoCuentas del DSA.
ETLMovimientos	LibroMayor.prn	CUBOS.MovimientosHechos	Carga la tabla de hehos MovimientosHechos, utiliza la tabla Movimientos del DSA y las Dimensiones Tiempo y Cuentas.
ETLProductos	Productos.prn	CUBOS.Productos	Carga la dimensión Productos, utiliza las tablas Productos, Pintas, Mixes, CatSubProductos, CatMedProductos, CatInfProductos del DSA
ETLClientes	Clientes.prn	CUBOS.Clientes	Carga dimensión Clientes, utiliza las tablas Clientes, Rubros y Sectores del DSA
ETLVentas	Libro de Documentos.prn	CUBOS.VentasHechos, CUBOS.Documentos.	Carga la dimensión Documentos y la tabla de hechos VentasHechos utiliza las tablas Documentos, ItemDocumento del DSA y las dimensiones Productos, Tiempo, Clientes y Vendedores
ETLVendedores	Vendedores.prn	CUBOS.Vendedores	Carga la dimensión Vendedores utilizando las tablas Vendedores y Central de Costos del DSA.

ETLCuentas


PlanDeCuentas.prn
PlanDeCuentas.xml
Par
in
Analizador
Out
log
CodCuenta
NomCuenta
CodSubGrupoCuenta
CodGrupoCuenta
in
Agrega Fecha
Out
Par
log
FecUltMod
getDate()
DSA.Cuentas
DSA.SubGrupoCuentas
DSA.GrupoCuentas
CodSubGrupoCuenta
NomSubGrupoCuenta
CodGrupoCuenta
NomGrupoCuenta
in
Integracion
Out
log
CUBOS.Cuentas

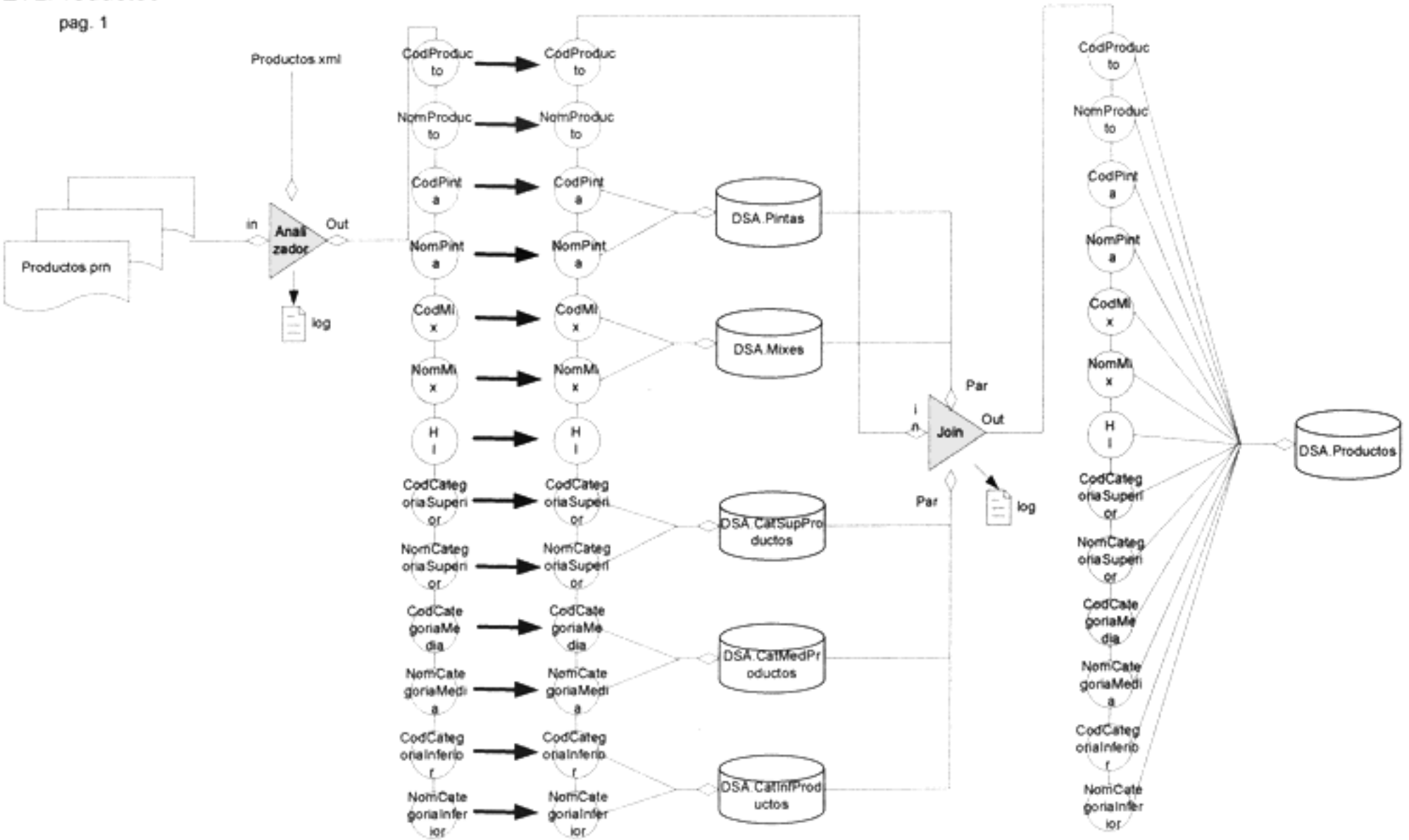
ETLMovimientos



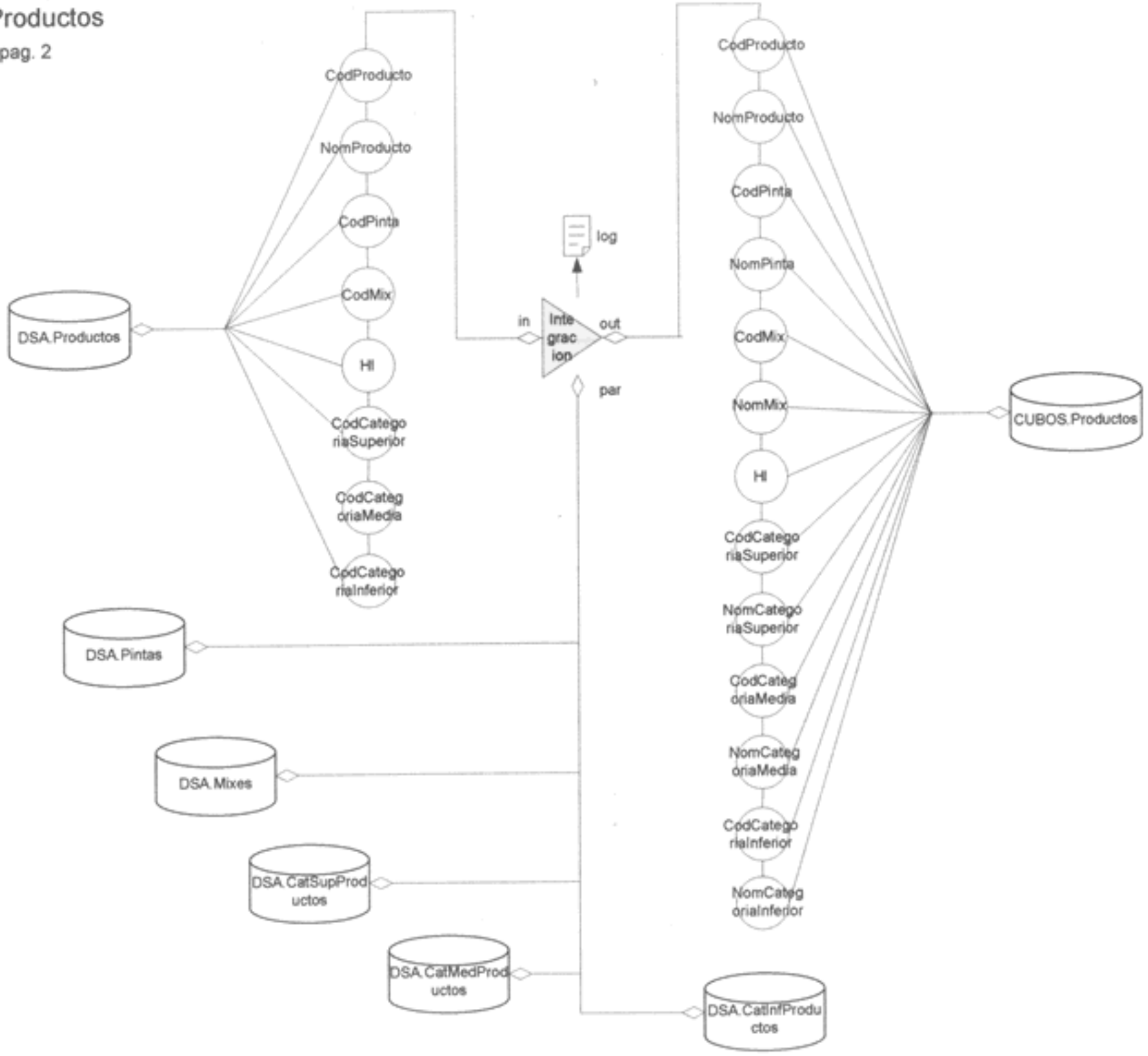


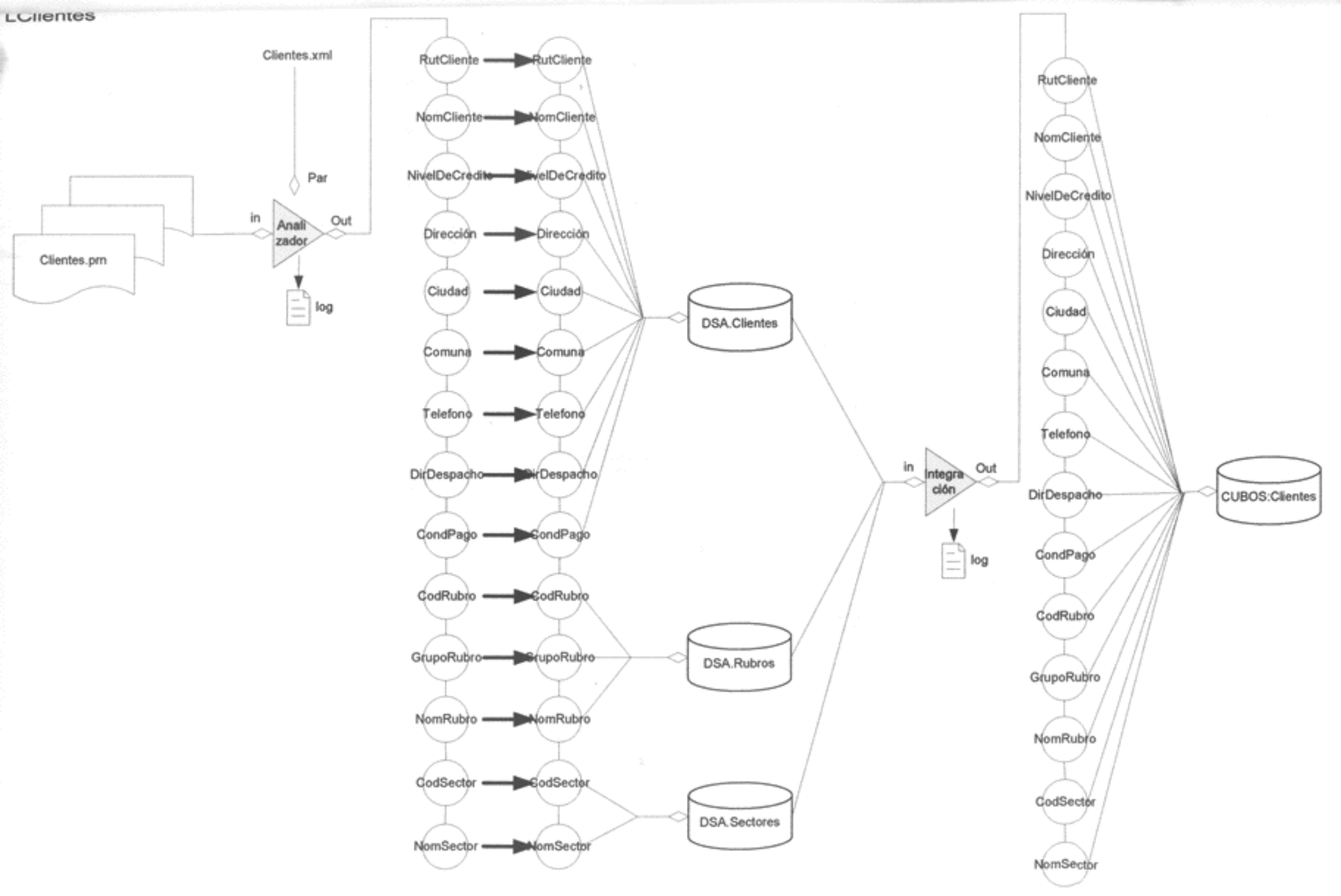
ETLProductos

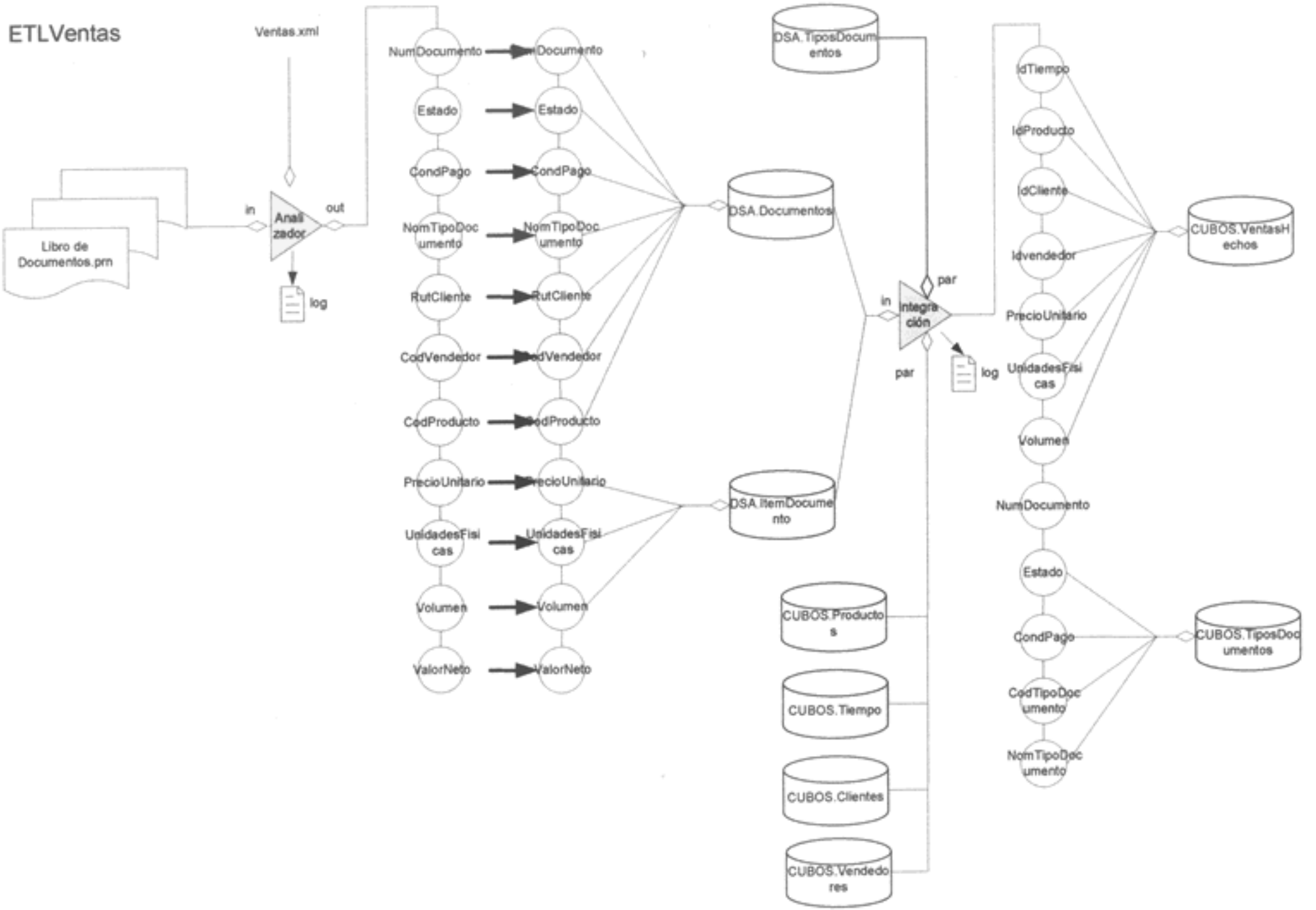
pag. 1



ETLProductos




LClientes
Clientes.xml
Par
Clientes.prn
in
Analizador
Out
log
RutCliente
NomCliente
NivelDeCredito
Dirección
Ciudad
Comuna
Telefono
DirDespacho
CondPago
CodRubro
GrupoRubro
NomRubro
CodSector
NomSector
DSA.Clientes
DSA.Rubros
DSA.Sectores
in
Integración
Out
log
CUBOS:Clientes


ETLVentas
Ventas.xml
Libro de Documentos.prn
in
Analizador
out
log
NumDocumento
Estado
CondPago
NomTipoDocumento
RutCliente
CodVendedor
CodProducto
PrecioUnitario
UnidadesFisicas
Volumen
ValorNeto
Documento
Estado
CondPago
NomTipoDocumento
RutCliente
CodVendedor
CodProducto
PrecioUnitario
UnidadesFisicas
Volumen
ValorNeto
DSA.Documentos
DSA.ItemDocumento
DSA.TiposDocumentos
in
Integración
par
par
log
CUBOS.Productos
CUBOS.Tiempo
CUBOS.Clientes
CUBOS.Vendedores
IdTiempo
IdProducto
IdCliente
Idvendedor
PrecioUnitario
UnidadesFisicas
Volumen
NumDocumento
Estado
CondPago
CodTipoDocumento
NomTipoDocumento
CUBOS.VentasHechos
CUBOS.TiposDocumentos

ETLVendedores