EasyBI.cl

Estrategia para la Implementación y Administración Inteligente de DataWarehouse (EIAD).

Orientada a minimizar las tareas de administración y adaptabilidad por cambios en los modelos de negocio en ambientes de alta complejidad.

Fecha: Marzo del 2012.
Versión: 1.0

Autor: Sebastián Rodríguez Robotham
e-mail: srodriguez@easybi.cl

Resumen

Este documento describe una estrategia genérica, aplicable a cualquier plataforma para la implementación y administración de DataWarehouse, cuyo principal objetivo es lograr el éxito, sustentabilidad y adaptabilidad del proyecto tanto a corto como largo plazo. La estrategia está enfocada principalmente en proyectos complejos, sin embargo puede emplearse en proyectos de cualquier envergadura.

Para asegurar el éxito de la solución, esta estrategia considera experiencias y elementos que en diversos proyectos han sido factores de fracaso o estanco, principalmente los relacionados a satisfacer nuevos requerimientos de los usuarios finales, en términos de calidad y confiabilidad de la información, agilidad en la adaptación a los cambios, facilidad en la creación de informes y cuadros de mando, rapidez en los tiempos de respuesta de las consultas y costo de la administración de la infraestructura, entre otros.

Finalmente, esta estrategia desarrolla conceptos claves, tales como la adaptación a los cambios, administración de procesos (Jobs), control de errores y mallas de procesos, elementos que son centrales en la estrategia EIAD y que han sido olvidados en la mayoría de la literatura relacionada a DataWarehouse, centradas principalmente en metodologías de administración de proyectos, creación de cubos y/o algunos procesos de carga.

Contenido

Introducción

Los proyectos de DataWarehouse, por lo general, giran en torno al diseño de tablas de hechos, dimensiones y la construcción de un proceso ETL que permita extraer la información desde los diferentes sistemas origen y cargarlos hacia un modelo OLAP. Al terminar el desarrollo, los procesos se prueban, los analistas generan algunos informes y se da por cerrado el proyecto. Sin embargo, en la medida que pasa el tiempo surge un gran número de necesidades que no fueron previstas en el transcurso del proyecto (algunas imposible de detectar, otras por falta de rigurosidad) y peor aún, incidencias en cuanto al diseño de los modelos OLAP y rendimiento del mismo, lo que es resuelto (muchas veces solo parchado) con enorme trabajo y esfuerzo innecesario.

El paradigma del diseño de DataWarehouse y Datamart es distinto al de hace 12 años atrás, en ese entonces era una regla común asumir que el diseño de los DataWarehouse no cambiaría, y que los procesos se mantendrían igual durante mucho tiempo. No obstante este paradigma ha cambiado, y hoy los gerentes y analistas necesitan agregar información, modificar criterios y ratios en forma rápida y dinámica, se agregan nuevos sistemas origen, sufren modificaciones o son reemplazados por otros nuevos. ¿Y qué pasa con el DataWarehouse?, el esfuerzo para efectuar estos cambios es enorme y los errores en la carga y cálculos en la información hace que se tomen malas decisiones, lo que ocasiona mucha pérdida de tiempo y dinero. Lo anterior ocurre principalmente porque los encargados del proyecto no diseñaron los procesos y ambiente OLAP para aceptar cambios, sino para terminar lo antes posible, asumiendo erradamente que el modelo no cambiaría.

Sin bien es cierto esta estrategia puede ser adoptada para cualquier proyecto de DataWarehouse / Datamart, es altamente recomendable utilizarlo en ambientes de alta complejidad, entendiendo por complejo lo siguiente:

- Gran cantidad de fuentes de información o sistemas origen (mayor a 5)
- Las fuentes de información pueden traer los mismos conceptos, con datos en distintos formatos (ejemplo: el nombre del cliente, una fuente con formato “nombre, apellido” y otra fuente con formato “apellido, nombre”)
- Gran cantidad de información (tablas de hechos resultantes mayores a 100 millones de registros).

Existe una gran variedad de formas para implementar proyectos de almacenes de datos, ¿Cuál de ellas es mejor?, dependerá principalmente del contexto del proyecto, calidad de datos, tiempo disponible para su implementación, complejidad y cantidad de fuentes de información, apoyo de la alta gerencia, etc., es por ello que no existe una estrategia única para llevar a buen puerto esta tarea, sin embargo esta propuesta considera una gran cantidad de buenas prácticas de diversos

autores y experiencia práctica de implementación y mantención de otros proyectos de DataWarehouse.

Objetivos

El objetivo de este documento es entregar una estrategia para facilitar el desarrollo de nuevos DataWarehouse y/o Datamart, con un enfoque centrado en la administración automática, flexible, ágil y de fácil adaptabilidad, que logre evitar los problemas actuales de adopción de nuevas necesidades, tanto tecnológicas como de negocio.

Como consecuencia se logrará un mejor rendimiento de la solución implantada, tanto financiero como operativo, y una mejora en los procesos de toma de decisiones por parte de la organización.

Como ilustra la figura 1, el gran esfuerzo detrás de una solución de inteligencia de negocios se encuentra en el ambiente DataWarehouse (sin restar importancia a las herramientas de visualización y descubrimiento de información), es la columna vertebral de todos los desarrollos relacionados al uso de herramientas estadísticas y Business Analytic. Sin una base de información confiable, accesible y flexible será extremadamente complejo que los proyectos complementarios sean exitosos. Es por lo anterior que este documento se centra en las mejores prácticas y técnicas que aseguren el éxito y sostenibilidad de la administración del ambiente DataWarehouse.

Figura 1: Incidencia en el éxito de proyectos de Inteligencia de Negocios.

Alcance de la Estrategia.

La estrategia abarca las dos primeras etapas del plan de trabajo descrito en el capítulo 1 (Entorno y Plan de Trabajo), esto es la etapa de Datos y DW/DM.

Por otro lado la estrategia definida en este documento no es prescriptiva, es decir no está diseñada específicamente para ninguna plataforma de software o hardware, sin embargo se han hecho implementaciones utilizando esta estrategia en ambientes SQL Server 2005 y 2008R2 e Integration Service. Los ejemplos de código y diseño que son utilizados en este documento han sido desarrollados en estas plataformas.

Mapa de navegación.

Los capítulos están distribuidos de la siguiente forma:

Capítulo 1. Entorno y Plan de Trabajo.
Introducción del ambiente organizacional previo a la implementación de una solución de DataWarehouse, y qué elementos debe considerar un ambiente óptimo.

Capítulo 2. Arquitectura del DataWarehouse.
En este capítulo se describen las diversas arquitecturas existentes para implementar proyectos de DataWarehouse, y se desarrolla la arquitectura propuesta por la estrategia EIAD.

Capítulo 3. Administración de Conflictos de Información.
Se especifican los principales problemas que existen cuando se desarrolla este tipo de soluciones, y cómo a través de la utilización de la arquitectura propuesta se logra dar respuesta a ellos.

Capítulo 4. Mecanismos de Control y Administración de Procesos.
En este capítulo se explica por qué no es suficiente la elección de una arquitectura para asegurar el éxito y sustentabilidad de la solución a largo plazo.

Capítulo 5. Resumen de la Estrategia y Arquitectura EIAD.
Como toda estrategia, existen beneficios y desventajas frente a otras alternativas, en este capítulo se describen las principales características relacionadas a EIAD.

1. Entorno y Plan de Trabajo.

Antes de comenzar con la descripción de la arquitectura propuesta para el desarrollo de DataWarehouse, es importante contextualizar el entorno de trabajo bajo el cual se implementa este tipo de soluciones, y describir a grandes rasgos el plan de trabajo genérico que debe ser considerado.

Áreas de Información y Gestión no Planificadas

Por lo genera una empresa o área que aún no tiene un programa formal de inteligencia de negocios vive diariamente en el caos de información, con muchos informes generados manualmente por diversos analistas (o encargados de generar la información sin los conocimientos técnicos y/o académicos adecuados), y cada uno de ellos utilizando sus propias formas de trabajar y organizar la información. En muchas de estas empresas, las personas o áreas que han surgido en forma no planificada cumplen en parte con su objetivo y resuelven en gran medida los requerimientos de información de los jefes o gerentes, sin embargo el costo que hay detrás de esta generación de informes es muy grande, al igual que la probabilidad de entregar la información con errores, debido en parte al cruce manual de datos de diversas fuentes.

Desde el punto de vista empresarial, existe un riesgo operacional enorme, por lo general los procesos para obtener la información, criterios para hacer los cálculos y forma de generar los informes no están documentados y solamente los conoce la persona encargada de realizar estas tareas, la información no está debidamente respaldada, todos o la mayor parte de los procesos están alojados en computadores y no en servidores con las políticas de seguridad adecuados, existe un gran número de fuentes de información que es utilizada por diversos analistas en forma inconexa, generando cada uno informes de gestión distintos, lo que ocasiona múltiples versiones de la verdad, produce inconsistencias y confusiones en la organización… y el listado con riesgos puede seguir creciendo.

La siguiente figura muestra el proceso genérico de confección de información de empresas que no tienen una estrategia formal planteada.

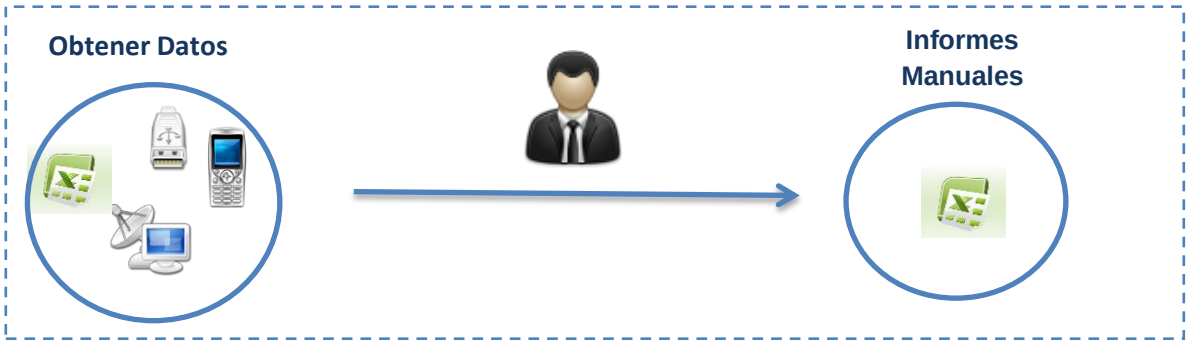


Figura 1.1: Proceso de generación de informes sin metodologías.

Si los riesgos operacionales y de generación de información son tan grandes, ¿Por qué las empresas no hacen nada al respecto y trabajan de esta forma por años?, las respuestas son variadas, sin embargo se repiten las siguientes:

- No hay presupuesto para comprar servidores y software e implementar un área como corresponde,
- Lo único que interesa es que los informes estén a fin de mes (o al final del día, dependiendo de la periodicidad de la información), a los jefes no les interesa qué deben hacer los analistas para cumplir con lo solicitado, o el esfuerzo detrás de ello.
- Desconocimiento tanto de los analistas que generan la información, como de los gerentes en cuanto a como se debe organizar un área de este tipo y qué se necesita para optimizar y automatizar los procesos.

Por lo general, los cambios se producen cuando las personas encargadas de generar los informes se van de la empresa y nadie más sabe como generarlos (en internet hay cientos de blogs que cuentan historias al respecto), cuando un virus o algo pasa al computador destinado a generar los informes y se pierde gran parte de la información, o cuando llega a la organización un gerente que conozca de gestión de información y quiera implementar una estructura adecuada.

Proceso y plan de trabajo genérico para el desarrollo de la gestión de información.

Para evitar el caos y riesgos en el proceso de generación de información, no es suficiente contar con tecnología de punta (tanto a nivel de hardware y software), sino que importante aún es tener claro un plan de trabajo a largo plazo y metodologías que apoyen a la consecución de los objetivos planteados, recordemos que tanto el hardware y software solo son herramientas que deben apoyar al cumplimiento de metas.

La siguiente figura ilustra los elementos básicos que debe considerar cualquier estrategia para desarrollar un área de gestión de información e inteligencia de negocios.

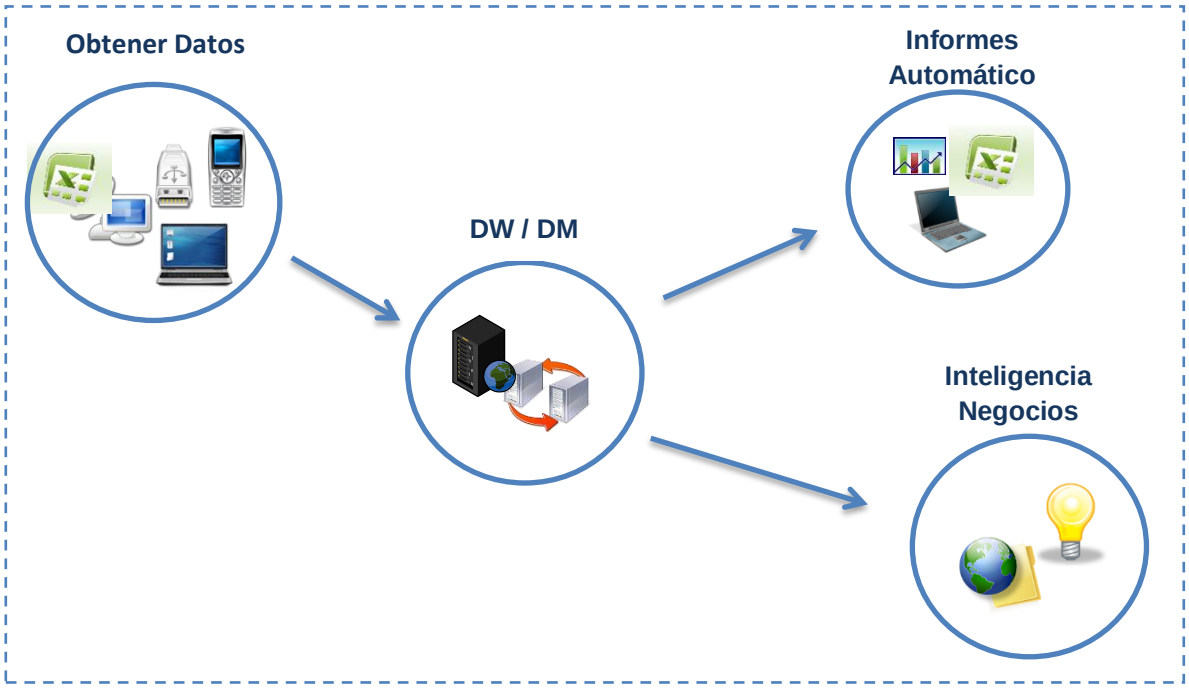


Figura 1.2: Proceso plan de trabajo genérico de gestión de información.

A continuación se describen las principales características de cada elemento del plan de trabajo.

- **Obtener Datos**: la obtención de los datos es una de las tareas primordiales en todo el proceso, el éxito, fracaso y credibilidad del proyecto dependen en gran medida de la calidad y comprensión de la información. La facilidad de acceder a ella y la comunicación que exista entre el equipo del DW/DM y las personas que administran los sistemas origen permitirá mitigar gran parte de los riesgos inherentes a las modificaciones y actualizaciones de estos sistemas. En la actualidad es muy común que los sistemas sean actualizados con gran frecuencia o a nuevas versiones (ya sea por evitar la obsolescencia tecnológica, cambios en las reglas de negocio, cambios hechos por el regulador local, etc.), es por ello que mantener los canales de comunicación abiertos con los administradores de sistemas tiene alta importancia, y no basta contactarlos solo al inicio del proyecto para solicitar el acceso a las bases o la generación de interfaces.

- **DW / DM (DataWarehouse / Datamart):** la información obtenida desde los sistemas origen será recopilada, limpiada y validada en este ambiente, asegurando que exista integridad y consecuencia en los datos. Como se comentó anteriormente, para mantener la información actualizada no basta con identificar la fuente de información y generar un proceso que la extraiga desde los sistemas origen y la cargue en el ambiente DW /DM, sino que mantenerla actualizada, recogiendo los cambios y actualizaciones de los sistemas origen es un reto bastante complejo. Dentro de los factores de éxito a considerar en esta etapa se encuentran:

- la facilidad y velocidad de acceder a los sistemas origen, minimizando el impacto para los usuarios habituales de dichos sistemas,
- la cantidad de información histórica disponible para análisis, que será utilizada en las dos etapas posteriores.
- La rapidez del ambiente y el tiempo de respuesta capaz de entregar al usuario final.

El gran objetivo de este ambiente es mantener la información histórica disponible para que los usuarios puedan consultarla en cualquier momento.

- **Informes Automáticos:** el objetivo de esta etapa es entregar a los usuarios de diversas áreas información suficiente para conocer, entender y administrar el negocio, entrando al nivel 2 de la pirámide de desarrollo BICC (ver anexo B). Lo anterior se logrará a través de una o varias herramientas que otorgue a los usuarios autonomía para extraer y generar sus propios informes de gestión desde diversas perspectivas, la programación de alertas tempranas y la confección de los siguientes elementos:

  - Informes de Trabajo: información para asignación de tareas diarias.
  - Informes de Gestión: Evaluación del trabajo hecho por las áreas.
  - Cuadros de Mando / Dashboard: Evaluación del desempeño de las áreas.
  - Scorecards: Evaluación global del área u organización.

- **Inteligencia de Negocios:** El objetivo de esta etapa es descubrir nuevo conocimiento para mejorar la toma de decisiones, a través de la optimización en los procesos de negocio, aumentar la comprensión del negocio e incluso descubrir o generar nuevas fuentes de ingreso, a través de la creación o modificación de productos o servicios.

  Lo anterior se canalizará a través del uso de diversas herramientas y técnicas, entre ellas se encuentran los estudios y modelos estadísticos, análisis predictivos, regresiones, generación de árboles de decisión, análisis de diversos escenarios (what if), procesos presupuestario, entre otros.

  Es posible iniciar esta etapa en paralelo a la etapa anterior, sin embargo es recomendable contar con algo de madurez en la generación de informes automáticos, debido a la importancia de conocer el estado actual del negocio.

2. Arquitectura del DataWarehouse.

La arquitectura y diseño de los procesos es uno de los puntos más críticos a la hora de desarrollar este tipo de soluciones, según lo que he visto en encuestas, foros de internet, empresas consultoras y experiencia de otros colegas, son muy pocos los profesionales que usan una arquitectura formal para realizar el proyecto.

Muchas soluciones BI se desarrollan como un gran proceso ETL, sin separar los sistemas origen y procesos internos de carga al OLAP. Tal vez al inicio del proyecto no es tan importante (principalmente por la exigencia de tiempos breves de desarrollo y entregables), sin embargo los proyectos va sufriendo modificaciones, adaptaciones y mejoras, por tanto al no tener separados los procesos es muy difícil hacer seguimiento y actualizaciones en forma segura.

A continuación se presenta el esquema general con los elementos y variantes que puede considerar cualquier arquitectura de DataWarehouse, sin importar el grado de complejidad que este tenga.

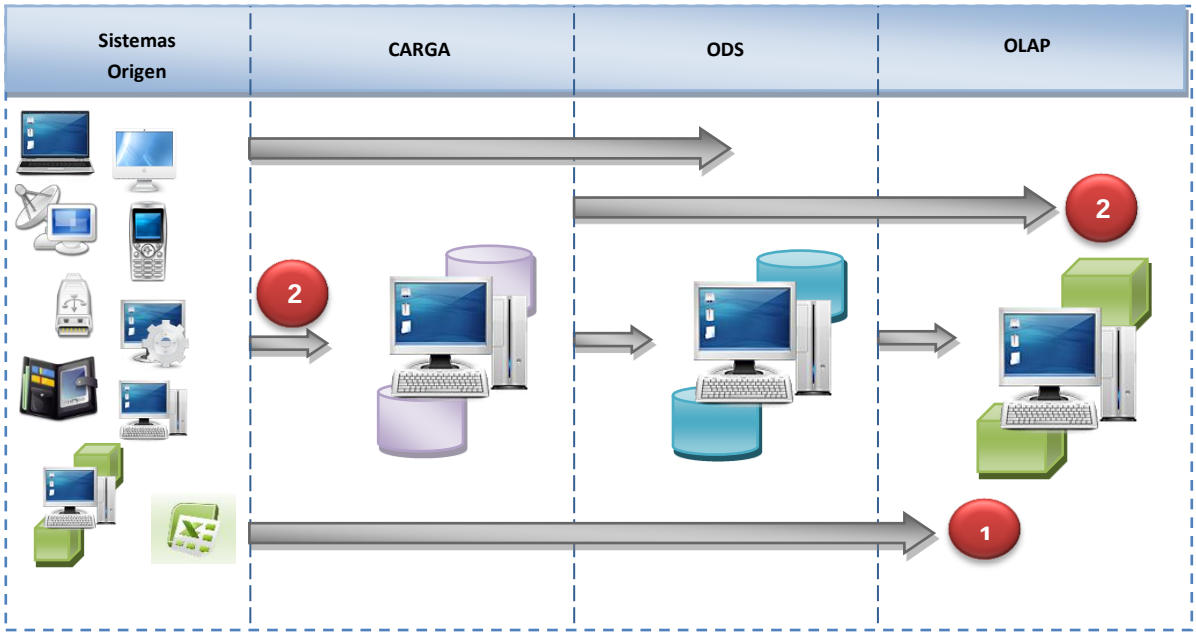


Figura 2.1: Arquitecturas de Implementación de DataWarehouse.

Las flechas muestras el flujo de información de un ambiente a otro. Como se comentó anteriormente, las dos variantes más utilizadas de esta arquitectura es la carga directa desde los sistemas de origen al ambiente OLAP (flujo 1) y el uso de un área de Staging (flujo 2). Sin embargo, todas estas combinaciones de flujos son usadas en diversos proyectos.

La gran ventaja de utilizar las variantes 1 e incluso la 2, es el tiempo de implementación de los proyectos, al pasar directamente al ambiente OLAP se requiere de mucho menos desarrollo, sin

embargo, como veremos más adelante, estas estrategias incrementan las labores de administración y adaptabilidad, sobre todo en proyectos de alta complejidad.

2.1. *Arquitectura propuesta*

La arquitectura propuesta se basa en la estrategia que logra minimizar de mejor forma los problemas comunes en la administración y adaptabilidad de los DataWarehouse, aunque igualmente se consideran algunos problemas de implementación. Lo anterior es muy difícil de realizar en proyectos que ya están implementados, por este motivo lo ideal es hacer un esfuerzo en el diseño inicial del proyecto.

Tal vez a muchos implementadores de soluciones de BI no les haga mucho sentido preocuparse de la administración y/o adaptabilidad de las soluciones, ya que creen erróneamente que estas soluciones no cambian en el tiempo o simplemente porque se encargan de desarrollar proyectos y nunca más ven a los clientes. Para los que mantenemos y desarrollamos proyectos con múltiples fuentes de información y cientos de miles de registros, sabemos que esto es crítico tanto para el éxito del proyecto, como para la aceptación y confiabilidad de los usuarios finales. Este tipo de herramientas, al estar bien implementadas y administradas se transforman en una ventaja competitiva de las empresas, por lo cual una fácil y rápida adaptación a los cambios es fundamental para el éxito de la estrategia corporativa y la satisfacción de los usuarios finales. En el capítulo 5 se entregará una comparación con las principales ventajas y desventajas de optar por una implementación basada en esta estrategia.

La siguiente figura ilustra la arquitectura EIAD, optimizada para minimizar el esfuerzo en la administración y adaptabilidad del DataWarehouse.

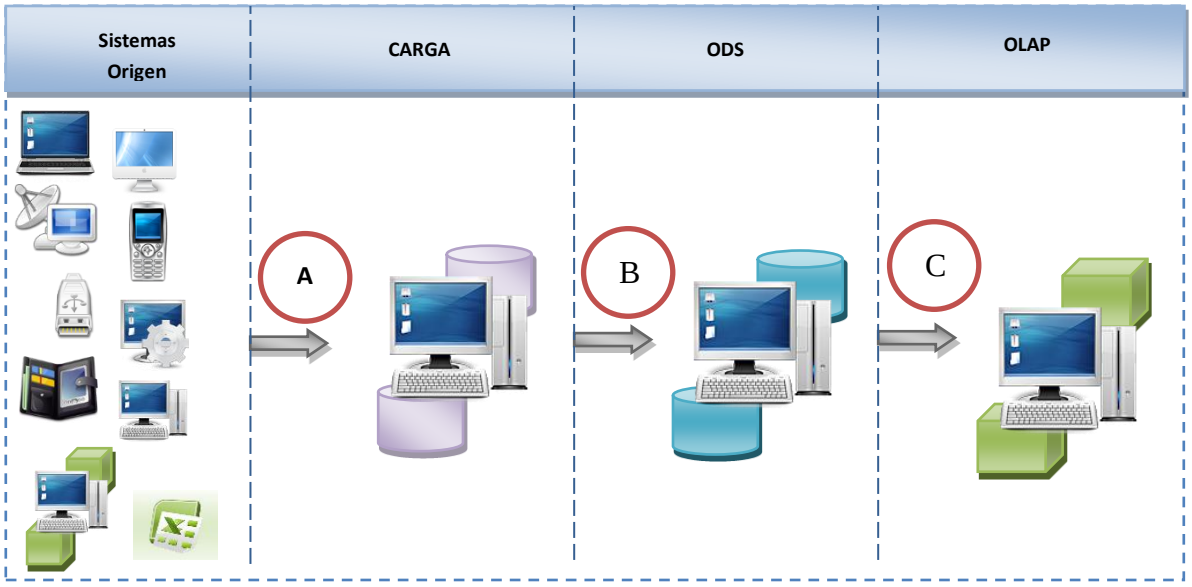


Figura 2.2: Arquitectura propuesta.

El flujo se define de la siguiente forma:

A. Cada sistema origen debe pasar por un ambiente de carga, cuyo principal objetivo es obtener la información lo más rápido y con el menor impacto posible en el rendimiento del sistema origen (muy similar al Staging área planteado por Ralph Kimball), también es posible hacer algunos procesos de limpieza básicos, como la homologación de los códigos del sistema (igualar largos, convertir textos a números, etc.).
B. El segundo paso es traspasar la información al ambiente ODS (Operational Data Store), el objetivo de este ambiente es validar la integridad y calidad de la información, el modelo de datos es similar a un sistema transaccional.
C. El tercer y último paso es la carga de los cubos OLAP a partir de uno o varios sistemas ODS.

Cada ambiente es independiente a los otros, es decir, las tablas del ambiente de carga no tienen relación directa con las tablas del ambiente ODS, y el diseño de la estructura del ambiente ODS no tiene directa relación con el sistema de origen, este diseño modular otorga una mayor flexibilidad y adaptabilidad al diseño completo de la solución, más adelante se explicará con mayor detalle.

La siguiente figura muestra el flujo detallado de un sistema de ventas (En el ejemplo el sistema de ventas en tienda es distinto al sistema de ventas por internet).

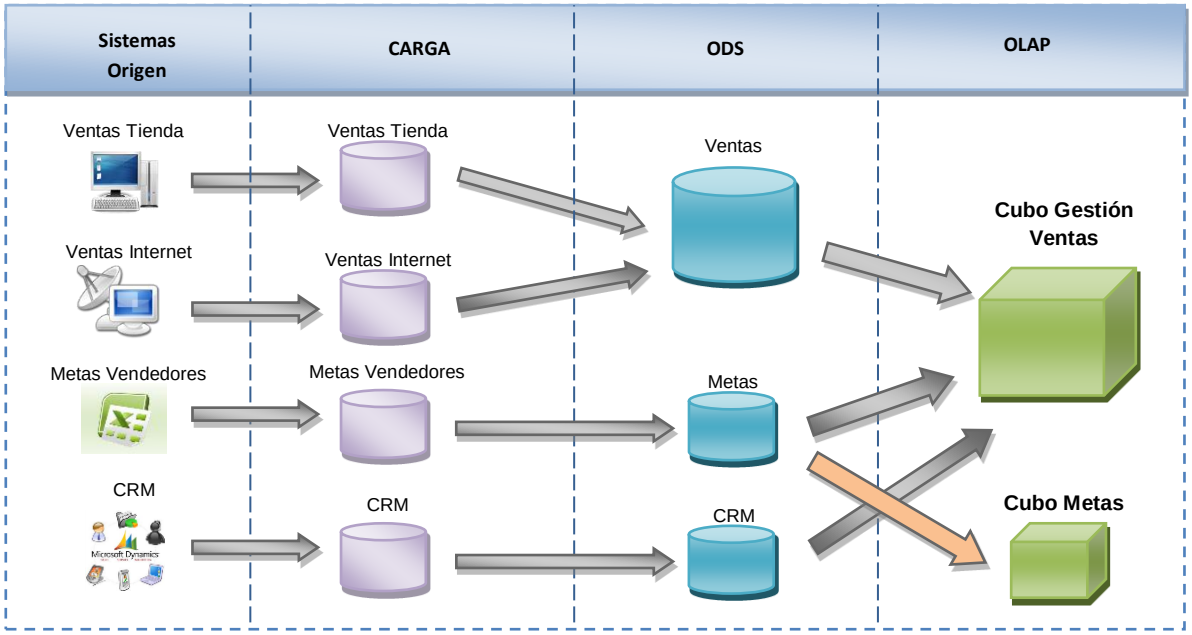


Figura 2.3: Arquitectura propuesta en detalle.

Este flujo detallado deja más clara la estrategia de implementación, cada cilindro en el diagrama representa un ambiente, no necesariamente una tabla, una base de datos o un esquema, eso

queda a criterio de cada implementador y depende de cada escenario, lo recomendado es que cada componente del modelo (Carga, ODS, OLAP), tengan archivos físicos diferentes para optimizar la utilización de discos en el servidor.

Lo interesante de esta propuesta es que permite la reutilización del esfuerzo realizado en la carga de información. Como se muestra en el diagrama anterior, los proceso del ambiente de Carga y ODS del sistema de metas para vendedores fueron hechos una sola vez, y pueden ser utilizados por diversos cubos, la información será consistente ya que aplica una sola lógica de carga y validación de la información, y la administración es muy sencilla: si hay un cambio en la base Excel de metas se hace el cambio en un solo lugar y será visible para todos los cubos.

En cambio si hubiéramos utilizado la estrategia 1 o 2 de la figura 2.1, tendríamos que haber hecho a lo menos dos veces el proceso de carga. Bajo ese mismo escenario algunos podrían simplemente copiar el primero proceso y adaptarlo para el segundo, pero como veremos más adelante las tareas de administración de esos dos procesos son mucho más caros que invertir tiempo en hacer un solo buen modelo de carga para que posteriormente sea ocupado por todos los cubos que lo requieran.

En escenarios aún más complejos y cambiantes, la empresa podría migrar el antiguo sistema Excel de metas a un sistema más sofisticado con bases de datos, si la información administrada fuera muy similar, lo único a cambiar en el diseño es el sistema origen y hacer un nuevo proceso en el ambiente de carga “Metas Vendedores”, el ambiente ODS no sufriría cambios ya que es independiente a la estructura del sistema origen, de esta forma se mantiene el proceso de carga del sistema Excel antiguo para mantener la historia y se remplaza por el nuevo sistema para los nuevos presupuestos y metas. La administración de este tipo de requerimientos es extremadamente sencilla y flexible, mantiene la trazabilidad de la información para revisiones o auditorías, y para los usuarios finales es transparente.

A continuación se explica en mayor detalle las características y roles de cada ambiente de la arquitectura basada en la estrategia EIAD.

Ambientes de Carga.

Cada sistema origen tiene sus propios procesos de carga, en caso de ser implementados con procedimientos almacenados, pueden ser uno o varios, por lo general se utiliza un procedimiento almacenado para cargar toda la información de los mantenedores o tablas maestras, y otro procedimiento almacenado para las tablas transaccionales, esto minimiza aún más el impacto por cambios en los sistemas origen.

Lo ideal en este ambiente es contar con una tabla de carga por cada tabla del sistema origen, sin embargo esto también queda a criterio del implementador, debido a que en ocasiones es posible unir varias tablas para optimizar los procesos o simplemente por motivos de diseño.

Para optimizar la carga de información, en lo posible las tablas no deben tener claves primarias ni Constraints, las actividades de validación de integridad y calidad de la información son tareas del ambiente ODS, sin embargo, es posible incluir algunas rutinas de conversión de información.

Muchas veces la información la obtendremos de distintas formas, ya sea conectando directamente a otro motor de base de datos, a través de planillas Excel o archivos de texto planos, es necesario en este ambiente cargar la información con los tipos de datos que corresponde, es decir, si la interfaz que estamos utilizando es un archivo plano, y los números vienen con separación de miles y en formato de texto, es necesario dejarlo en formato numérico para que todos los procesos posteriores puedan entender la información.

Ambientes ODS

Un sistema ODS puede alimentarse de uno o varios ambientes de carga, lo recomendado es que sea uno a uno. En el ejemplo de la figura 2.3, los ambientes de carga “Ventas Tienda” y “Ventas Internet” son utilizados para llenar el mismo ambiente ODS “Ventas”, este diseño podría perfectamente haberse separado en ODS “Ventas Tienda” y “Ventas Internet”, sin embargo es necesario evaluarlo en cada implementación, y dependerá de qué tan homogéneos sean los sistemas, qué frecuencia de actualización tengan, granularidad, etc.

El diseño del modelo de datos debe ser de tipo transaccional, es decir, que a lo menos esté en tercera forma normal. De esta forma podemos asegurar que la información esté consistente (siguiendo con el ejemplo propuesto, que todos los clientes y vendedores existan para todas las ventas).

En algunas ocasiones la información de carga no vendrá consistente, y tendremos que decidir entre detener toda la cadena de procesos hasta que sea corregido el problema en el sistema origen (y volver a ejecutar nuevamente la malla), o agregar manejadores de errores en el mismo proceso, que siguiendo con el ejemplo de la figura 2.3 sería creando el cliente sin nombre en el maestro de clientes del ODS y agregar una marca para revisión manual. Por lo general recomiendo la segunda alternativa (utilizar manejadores de errores en los procesos), ya que no detiene la cadena de carga, la información saldrá a tiempo y permite una revisión posterior por parte de los administradores del DataWarehouse. En la sección 3.2 del capítulo 3 se explica más detalladamente esta estrategia.

Debido a que este ambiente transaccional cargará toda la información al ambiente OLAP, es posible eliminar cierta información histórica del ambiente ODS (queda a criterio de cada

implementador), sin embargo recomiendo mantener toda la historia posible en el ambiente ODS hasta que los cubos hayan alcanzado un grado de madurez suficiente como para no volver a reprocesar el ambiente. Es muy común que después de ser creado un cubo, los usuarios pidan nuevos ratios (ya sea porque no existían o porque no fueron levantados), en el mejor de los escenarios se podrá calcular directamente en el OLAP, pero en el pero de los escenarios habrá que reprocesar la información histórica desde el ODS hacia el OLAP, y cargar nuevamente toda la información histórica al ODS es un costo en tiempo demasiado alto (en algunos casos semanas de carga) que podría ser evitado. La recomendación final es que cada implementador evalúe el costo financiero de mantener la información duplicada en el ambiente ODS por un período de tiempo (por lo menos 6 meses a 1 año) vs el costo de oportunidad de no contar con información estratégica cuando es requerida para tomar decisiones de negocio importantes.

Ambientes OLAP

Finalmente el OLAP "Cubo Gestión Ventas" se alimenta de todos los ambientes ODS que sean necesarios para los usuarios

Tal como muestra la figura 2.3, la utilización de esta estrategia permite mantener una trazabilidad de la relación de cada sistema origen en relación a qué cubos alimenta, lo que facilita un análisis de impacto mucho más claro y efectivo, como se explicó anteriormente.

En relación al diseño físico de los cubos, la arquitectura no obliga a elegir una estrategia específica de almacenamiento (Rolap, Olap u HOLAP), de hecho esta decisión dependerá del tipo de proyecto o tecnologías disponibles por cada implementador.

Si bien es cierto el diseño de los cubos puede ser de tipo Estrella o Copo de Nieve, es altamente recomendado usar únicamente diseños de tipo Estrella por su mejor performance en escenario de cubos con muchas dimensiones (mayor a 8). Hay varias estratégicas en la literatura referidas a unificar dimensiones de tipología "copo de nieve" en una sola de tipo estrella, la mayoría justificadas en que la utilización de espacio en disco al unir varias dimensiones es insignificante comparada con las tablas de hecho, y el costo de hacer join en las consultas es mucho mayor para el usuario final en término de tiempos de respuesta.

También es importante contar con una visión holística de los cubos, saber cuales existen y también cuales deberían ser creados en el corto, mediano y largo plazo, con el objetivo de diseñar de mejor forma las dimensiones compartidas (siguiendo la propuesta "BUS Architecture" planteada por Ralph Kimball).

3. Administración de Conflictos de Información.

Una de las preguntas más comunes planteadas por los administradores, relacionadas con la utilización de información en el DataWarehouse es ¿en cual información debo confiar y mostrar a los usuarios?, dado que tenemos varios sistemas de origen, la información de una entidad, por ejemplo clientes, vendrá de diversos sistemas, por lo que pueden haber muchos problemas en relación a la integridad y confiabilidad de información. Hacer un análisis exhaustivo a las diversas fuentes de información para determinar cual es la mejor es una tarea titánica, más aún si, una vez identificada la mejor, debemos corregir en múltiples procesos de carga. Al implementar los procesos bajo la estrategia EIAD, facilitará en forma importante la manera de administrar y aplicar las reglas de negocio.

A continuación se explican los principales conflictos de información que se pueden encontrar al implementar proceso de ETL, y como la estrategia EIAD ayuda a resolverla y administrarla de forma óptima.

3.1. *Información de una entidad inconsistente, proveniente de varios sistemas de origen.*

Como podemos ver en el diseño de la figura 2.3, la información de los clientes podría venir de 3 fuentes de información diferentes: “Ventas Tienda”, “Ventas Internet” y “CRM”, ¿Cuál tiene la información correcta y más completa?, claramente la respuesta dependerá del conocimiento que tengamos de los distintos sistemas y fuentes de información, pero muy común es “Todas y Ninguna”. Como cada sistema de origen es independiente y por lo general no están integrados (ni siquiera se comunican), la información puede no estar completa en ningún sistema (es por ello que se construyen los DataWarehouse), por tanto deberemos asignar distintos niveles de credibilidad a la información, supondremos para efectos de este ejemplo que la información del CRM es la más certera, tiene la información de la mayoría de los clientes actualizados, sin embargo, está incompleto, ya que los clientes que compran por internet pueden ser clientes nuevos y aún no se han ingresado manualmente al CRM, ¿Qué hacemos?, ¿Cómo administramos este caos de información y qué mostramos a los clientes?.

En el diseño del OLAP “Cubo Gestión Ventas”, lo más probable es que exista una única dimensión de “Clientes” que se deberá cargar, esa dimensión será compartida por todos los cubos existentes (siguiente lo propuesto por Ralph Kimball y el “BUS Architecture”), por tanto deberá existir un único proceso de carga desde el ambiente ODS hacia el ambiente OLAP, el siguiente algoritmo resolverá este conflicto:

a) Agregar todos los clientes nuevos desde ODS “CRM”, que no existan en la dimensión de clientes
b) Agregar todos los clientes nuevos desde ODS “Ventas Tienda”, que no existan en la dimensión de clientes
c) Agregar todos los clientes nuevos desde ODS “Ventas Internet”, que no existan en la dimensión de clientes
d) Actualizar la información de los clientes desde ODS “CRM” que hayan sufrido modificaciones en el ODS “CRM”

El orden de este algoritmo no es antojadizo, de hecho primero se agregan todos los clientes nuevos desde el sistema ODS “CRM”, ya que se confía más en esa información, luego se agregan los clientes nuevos de los otros dos sistemas. Según las reglas de negocio de este ejemplo la información de los clientes se mantienen y actualiza en el CRM, por tanto cualquier cambio que sufra esta información es necesario refrescarla en la dimensión de clientes.

El paso “D” puede ser remplazado por “Insertar” en vez de “Actualizar” en caso que se esté utilizando un tipo de SCD (Slow Changing Dimension) más complejo que la simple actualización de información.

En resumen, al utilizar la esta estrategia y arquitectura EIAD, la administración y mantención de los conflictos de información se torna mucho más simple y flexible, ya que los procesos de carga de las dimensiones están centralizados en un solo proceso y son compartidos por todos los cubos, lo que mantiene la consistencia de información en los diferentes sistemas de gestión OLAP. Incluso si inicialmente no hubiera existido el CRM y la dimensión de clientes se alimentara solamente desde los maestros de clientes de los ODS “Ventas Tienda” y “Ventas Internet”, incluir el sistema origen CRM y modificar los procesos de carga de la dimensión “Clientes” para que lean la información de dicho sistema sería extremadamente sencillo, nuevamente sin mayores problemas para los usuarios finales o ambientes existentes.

3.2. *Información no íntegra en una fuente de información.*

Algunos de los sistemas de origen que cargamos deberían estar, a lo menos, en tercera forma normal para asegurar que, por ejemplo, cualquier venta que se haga esté asociada a un cliente en el maestro de clientes, sin embargo esto muchas veces no pasa, y nos encontramos con ventas a clientes que no existen en el maestro de clientes. Resolver este tipo de inconvenientes es bastante sencillo utilizando la estrategia y arquitectura EIAD, hay que aplicar el siguiente algoritmo en el proceso ETL del ambiente ODS.

a) Agregar todos los registros desde el maestro de clientes del sistema origen, que no existan en el maestro de clientes en ODS
b) Agregar todos los i.d. de clientes desde el transaccional de ventas del sistema origen, que no existan en el maestro de clientes en ODS y **agregar una marca en el maestro de clientes en ODS para identificar como "incidencia"**

Este simple algoritmo permite agregar, por lo menos, el I.d. del cliente en el maestro de clientes en ODS y asignar una marca de incidencia, para que sea revisado por el administrador e identifique si el problema proviene del sistema origen, o si es un error en el proceso ETL del DataWarehouse.

Estas marcas de incidencias deben existir en todas las tablas maestras, y también debe haber un proceso automático que avise al administrador de sistemas de dichas incidencias para que sean revisadas y la información sea completada, ya sea a través de otro proceso automático o un proceso de actualización manual de información.

Como se puede deducir, en ningún momento se detuvo la malla de procesos porque hubo una incidencia de este tipo, pero lo interesante es que se agrega una marca de incidencia que puede ser identificada, analizada y corregida. Lo ideal, de todas formas, es que la marca de incidencia nunca desaparezca de los maestros y sea manejada a través de estados de incidencia, por ejemplo "detectada", "revisada" y "resuelta", de esta forma no se pierde la trazabilidad y estadísticas de las incidencias y correcciones.

3.3. *Información que cambia a través del tiempo.*

Es común encontrar información que cambia a través del tiempo, por ejemplo puede ocurrir que por malas prácticas en la administración de los sistemas origen, se reutilicen los códigos de vendedores, es decir, cuando se va un vendedor y llega otro, en vez de crear un nuevo código, utilizan el mismo código del vendedor anterior y lo asignan al vendedor nuevo, de esta forma no hay que hacer nuevamente la asignación de cartera. Lamentablemente este tipo de prácticas induce a errores en la información histórica y, por ende, en la toma de decisiones.

Una forma sencilla de solucionar este tipo de inconvenientes es a través de versiones de la información en los maestros, esta es una práctica que puede ser encontrada en los SCD de Ralph Kimball, y se aplica creando un código interno del DataWarehouse (específicamente en el ambiente ODS) para la entidad, que para efectos de este ejemplo es la entidad de Vendedores. Como resultado el maestro de vendedores en el ambiente ODS tendrá el código interno de DW (y esta será la PK), el código del sistema origen y el nombre del vendedor, luego para cada cambio en la tupla código sistema origen y nombre de vendedor se generará una clave interna nueva (PK), y de esta forma será posible diferenciar el mismo código de vendedor en el sistema origen, con

distinto nombre de vendedor a través de la historia. También es posible incorporar fechas para saber cuando se produjeron estos cambios o en qué fechas estaba vigente uno u otro vendedor.

La siguiente figura muestra un ejemplo del resultado final del proceso mensual de carga de vendedores, suponiendo que de un mes a otro cambió el vendedor "V01", pero se asignó el mismo código al vendedor nuevo.

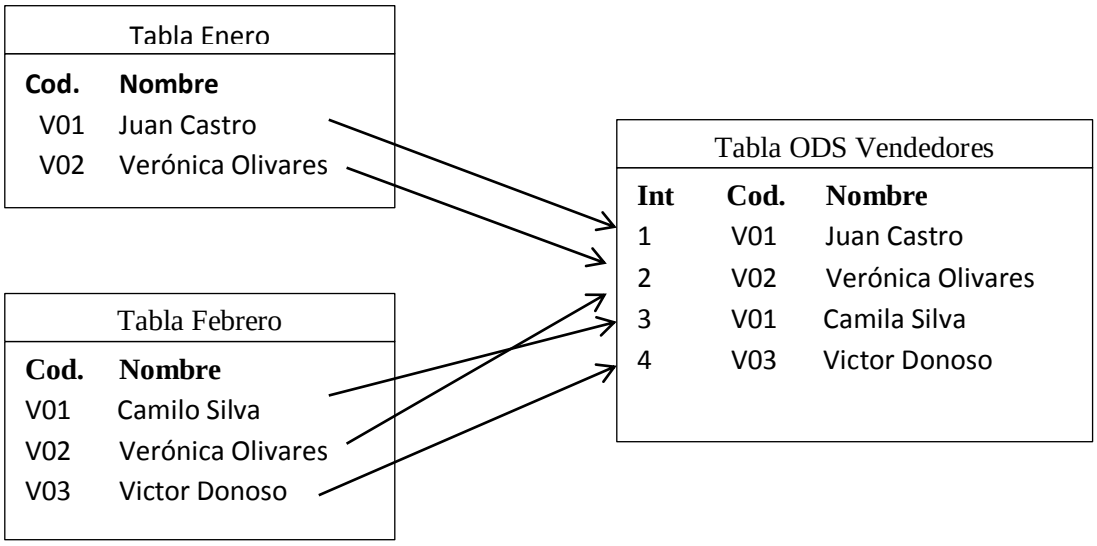


Figura 3.1: Proceso de actualización de claves subrogadas desde ambiente Carga a ODS.

Esta práctica no es nueva y, como se mencionó anteriormente, ha sido propuesta por otros autores, sin embargo al utilizar formalmente el ambiente de ODS propuesto por la estrategia y arquitectura EIAD, es posible centralizar toda esta lógica de negocios en este ambiente, y los diversos cubos que son cargados por los ETL del ambiente OLAP no tendrán que lidiar con este problema.

4. Mecanismos de control y administración de procesos.

Ya se han señalado los beneficios de utilizar la arquitectura EIAD en ambientes de alto dinamismo y/o complejidad, sin embargo hay otros elementos importantes que también es necesario considerar y administrar. Cuando la construcción de los procesos de Carga, ODS y OLAP son terminados y se comienzan a utilizar, por lo general el rendimiento en cuanto a tiempo y recursos será bastante aceptable, sin embargo en la medida que pasa el tiempo (semanas o meses, dependiendo de la periodicidad de carga), el tiempo de ejecución de éstos será cada vez más lentos. Es probable que estos incrementos de tiempo sean aceptables para algunos equipos de trabajo, sin embargo en ocasiones puede generar muchos inconvenientes, sobre todo de cara a la calidad de servicio entregado a los usuarios finales.

Es bastante normal observar estas variaciones en los tiempos de ejecución, sobre todo por la gran cantidad de información que hay disponibles en estos días, sin embargo es importante diferencia si el aumento en los tiempos corresponde solamente a incrementos en la cantidad de datos procesados, o hay porciones de código y procesos que no fueron optimizados (ya sea porque se pasó por alto o simplemente eran imprevisibles).

Para identificar el tipo de problema (ej. problema con el rendimiento por causa de una mala codificación o por otros motivos), primero debemos encontrar el o los procesos y cuales son las tablas y campos involucrados (ver figura 4.1), y esta es una tarea bastante compleja si no se cuenta con las herramientas adecuadas.

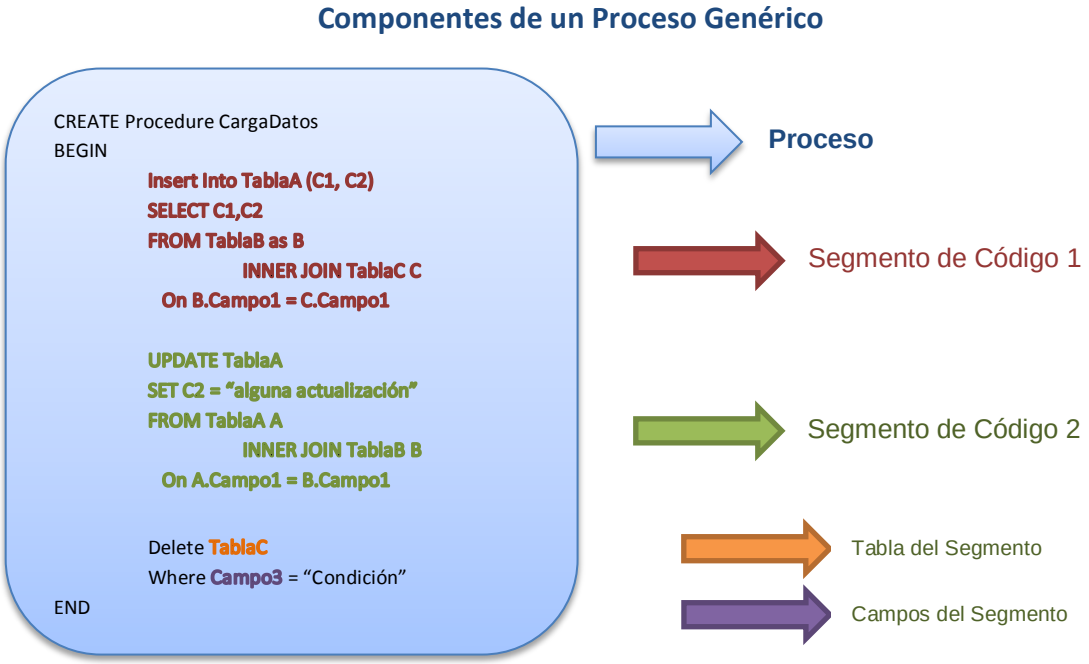


Figura 4.1: Estructura de un proceso genérico (Proceso → Segmento de Código → Tablas → Campos)

Encontrar los “mega procesos” que toman mayor tiempo en ejecutar debe ser bastante sencillo para la mayoría de las herramientas ETL, ya que por lo general entregan información del inicio y fin de los procesos, por el contrario si estamos trabajando con procedimientos almacenados deberemos implementar mecanismos propios que entreguen dicha información (ver anexo A).

Sin embargo lo anterior no es suficiente para determinar que ***fragmento de código*** dentro del proceso tiene problemas de rendimiento, para encontrar esto, por lo general, es necesario editar los procesos y ejecutarlos manualmente, paso por paso, lo que conlleva invertir una gran cantidad de tiempo en hacer la depuración, con el riesgo de afectar los procesos normales de carga y la disponibilidad de información, tanto de los sistemas de origen, como de los usuarios finales del DataWarehouse. Por otro lado hacer esta depuración manual no asegura que encontremos el problema, ya que estaremos ejecutando los procesos en condiciones diferentes a las normales, por ejemplo el ambiente de producción puede ejecutar simultáneamente varios procesos, entre ellos el que estamos evaluado, y si lo ejecutamos en forma independiente para hacer la depuración no obtendremos los mismos resultados.

Principales Problemas

Algunos de los problemas que son muy difíciles de detectar en ambientes ETL tradicionales son los siguientes:

- Qué fragmentos de código, dentro de un proceso, es el causante de la mayor cantidad de tiempo de ejecución.
- En caso que algún proceso no finalice correctamente, ¿qué fragmento de código fue el causante de dicho error?
- En caso de demora de los procesos y que aún no terminen, ¿qué proceso es el que está corriendo actualmente, y qué fragmento de código es el causante de dicho retraso?
- Comparativamente, ¿qué proceso(s) ha(n) incrementado su tiempo de ejecución en los últimos “N” días, semanas o meses?
- Comparativamente, ¿qué proceso se ha comportado en forma anormal en las últimas semanas o meses, es decir, ha tenido tiempos de ejecución diferentes?
- ¿qué fragmentos de código del proceso identificado anteriormente es el que produce la demora en la ejecución de dicho proceso?
- ¿qué proceso o fragmento de código, es el causante de la mayoría de los errores de ejecución en los últimos “N” días/semanas/meses?

Debido a todo lo anterior es indispensable tener un proceso ETL sólido en todos los ambientes, que entregue la mayor cantidad de información posible acerca de la ejecución de los procesos y los fragmentos de código relevantes, de tal forma que el hacer una evaluación minuciosa de un proceso o grupo de procesos sea una tarea trivial, y no una verdadera cruzada imposible de

alcanzar. Para lograr esto, cada ambiente (Carga, ODS, OLAP) deberían considerar los siguientes elementos:

a) Control de tiempos de ejecución de cada proceso
b) Control de tiempos de ejecución de cada fragmento de código relevante, dentro de cada proceso.
c) Historial de ejecución de procesos (de los puntos a y b) suficiente para evaluar y analizar tendencias.
d) Mallas de ejecución que permitan reprogramar en forma automática las tareas en caso de algún error.
e) Capacidad de enviar e-mail o mensajes en forma automática a los administradores de las áreas de informática, en caso de ocurrir algún error.

El ambiente óptimo, en resumen, debe permitir el aviso de contingencias en forma automática a los responsables de cada proceso crítico (por ejemplo enviar e-mail al área de producción en informática para que revisen las interfaces), para que la correcta ejecución de los procesos dependa en la menor medida posible de intervención humana, ya que esta última es una de las principales causas los errores en las cargas de información, mientras mayor intervención humana exista en un proceso de carga, mayor será la cantidad de errores posibles.

Ejemplo de Administración y Control de Procesos

A continuación se muestran un ejemplo del ambiente de ejecución de procesos que contiene todos los elementos mencionados anteriormente, la ejecución de estos procesos se lleva a cabo todos los días, y se registra la información del proceso completo y además de cada fragmento de código relevante, con ello es posible ver, en cualquier momento, cuanto se demora un proceso, en caso de haber un error, es muy fácil detectar qué fragmento causó el problema y efectuar las acciones correspondientes (revisar, evaluar, corregir y volver a ejecutar los procesos).

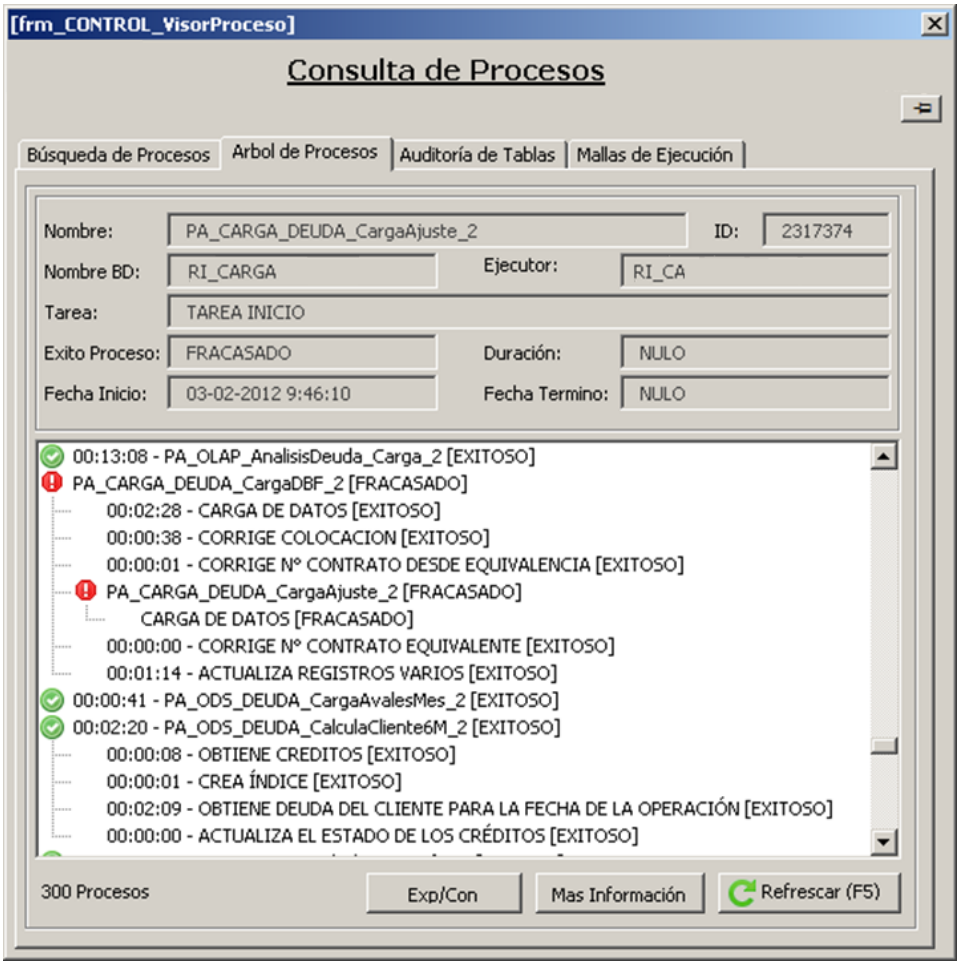


Figura 4.2: Monitor de procesos.

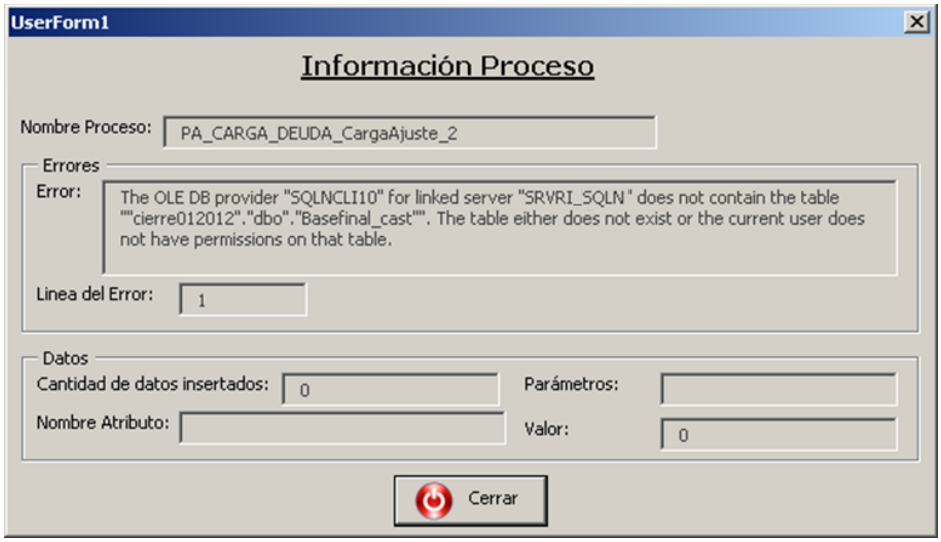


Figura 4.3: Detalle del error en monitor de procesos.

En la práctica, esta herramienta es utilizada para:

- Monitorear la ejecución de los procesos diarios/semanales/mensuales, consultar a cualquier hora del día qué procesos se están ejecutando.
- En caso de error, identificar qué fragmento de código dentro del proceso fue la causante del problema
- Almacenar información histórica de la ejecución de los procesos y fragmentos de código, luego esta información es utilizada para evaluar los aumentos en los tiempos de ejecución, tanto a nivel de proceso como a nivel de fragmento de código.

Además del control y seguimiento de los procesos, existe una malla de ejecución de procesos programada en SQL Server 2008R2 que se encarga de ejecutar en forma automática y en paralelo los procesos, según la fecha/hora planificada y siempre y cuando los procesos antecesores hayan sido ejecutados en forma correcta. En caso de error de algún proceso, éste se reprograma en forma automática (según la configuración de cada proceso, por ejemplo se puede reprogramar para 10 minutos después de la última ejecución).

A continuación se presentan las principales características de la malla de ejecución de procesos:

- Administración de los procesos que se ejecutarán, con los siguientes parámetros:
  - N° de reintentos en caso de error
  - T° en minutos para reprogramar el proceso fallido
- Administración de las dependencias de cada proceso:
  - Los procesos hijos no se ejecutarán hasta que los procesos padres estén correctamente ejecutados
- Ejecución de procesos en paralelo desde componente Agent de SQL Server 2008R2, a través de un procedimiento almacenado.

En el anexo “A” se encuentra parte del código que permite administrar los procedimientos almacenados de Carga, ODS y OLAP.

5. Resumen de la Estrategia y Arquitectura EIAD

Como se ha visto a lo largo de este documento, la utilización de la estrategia y arquitectura EIAD en ambientes de alta complejidad entrega múltiples beneficios, pero también tiene algunas desventajas frente a otras estrategias de implementación (como por ejemplo las estrategias 1 y 2 planteadas en la figura 2.1).

A continuación se describen las principales características de la estrategia y arquitectura EIAD, expresadas en beneficios y desventajas.

5.1. *BENEFICIOS*

A continuación se presentan los principales beneficios de EIAD frente a las estrategias 1 y 2 de la figura 2.1.

- Estandarización de los procesos para cada fuente de información externa: Todos los procesos tendrán, a lo menos, un ambiente de Carga y ODS, lo que mejora el orden y estructura del proyecto.
- Administración del cambio está intrínseco en la estrategia EIAD, facilita y promueve la rápida adaptabilidad a los cambios del negocio y entorno, a través de su arquitectura modular los ambientes de CARGA, ODS y OLAP.
- Facilidad para optimizar los procesos ETL: al contar con información histórica y detalle de cada proceso y de cada fragmento de código relevante, es posible detectar donde está el código no optimizado y tomar las medidas correctivas y preventivas adecuadas (por ejemplo rehacer el código, crear índices, particionar o comprimir archivos, etc.).
- Facilidad para auditar los procesos: la explicación de los dos beneficios anteriores justifican que la auditoría de estos procesos sea fácil y transparente, al estar todos los registros de ejecución y en cada ODS los maestros con las incidencias encontradas y corregidas, es fácil hacer un seguimiento, trazabilidad y documentación de los procesos.
- Facilidad en la adaptabilidad de la solución, al ser una implementación “modular en el ambiente ODS”, es posible cambiar un sistema ODS sin la necesidad de intervenir el resto de los sistemas, y ajustar los cambios solo en los procesos de carga de cada ambiente OLAP.

5.2. *DESVENTAJAS*

A continuación se presentan las principales desventajas de utilizar EIAD frente a las estrategias 1 y 2 de la figura 2.1

- Tiempo de desarrollo del proyecto: el tiempo de desarrollo del proyecto será superior utilizando esta estrategia, debido a que hay mayor cantidad de código y procesos que implementar para asegurar la sostenibilidad y mejor administración del proyecto en el largo plazo.
- Alto nivel de conocimiento y experiencia del equipo: es necesario que el equipo de desarrollo tenga un nivel de conocimiento más avanzado y también mayor experiencia en la implementación de este tipo de proyectos, lo que conlleva, en teoría, un mayor costo inicial de desarrollo.

6. Anexos

Anexos A. Código SQL del Administrador de Procesos.

A continuación se incluye parte del código que utiliza la herramienta de Administración de Procesos utilizada como ejemplo en el capítulo 4. Este script incluye código que permite controlar la ejecución de los diversos procedimientos almacenados, generando un diccionario de SP fácil de utilizar y con información de cada ejecución.

El siguiente diagrama muestra cómo funciona el framework de administración de procesos para Procedimientos Almacenados.

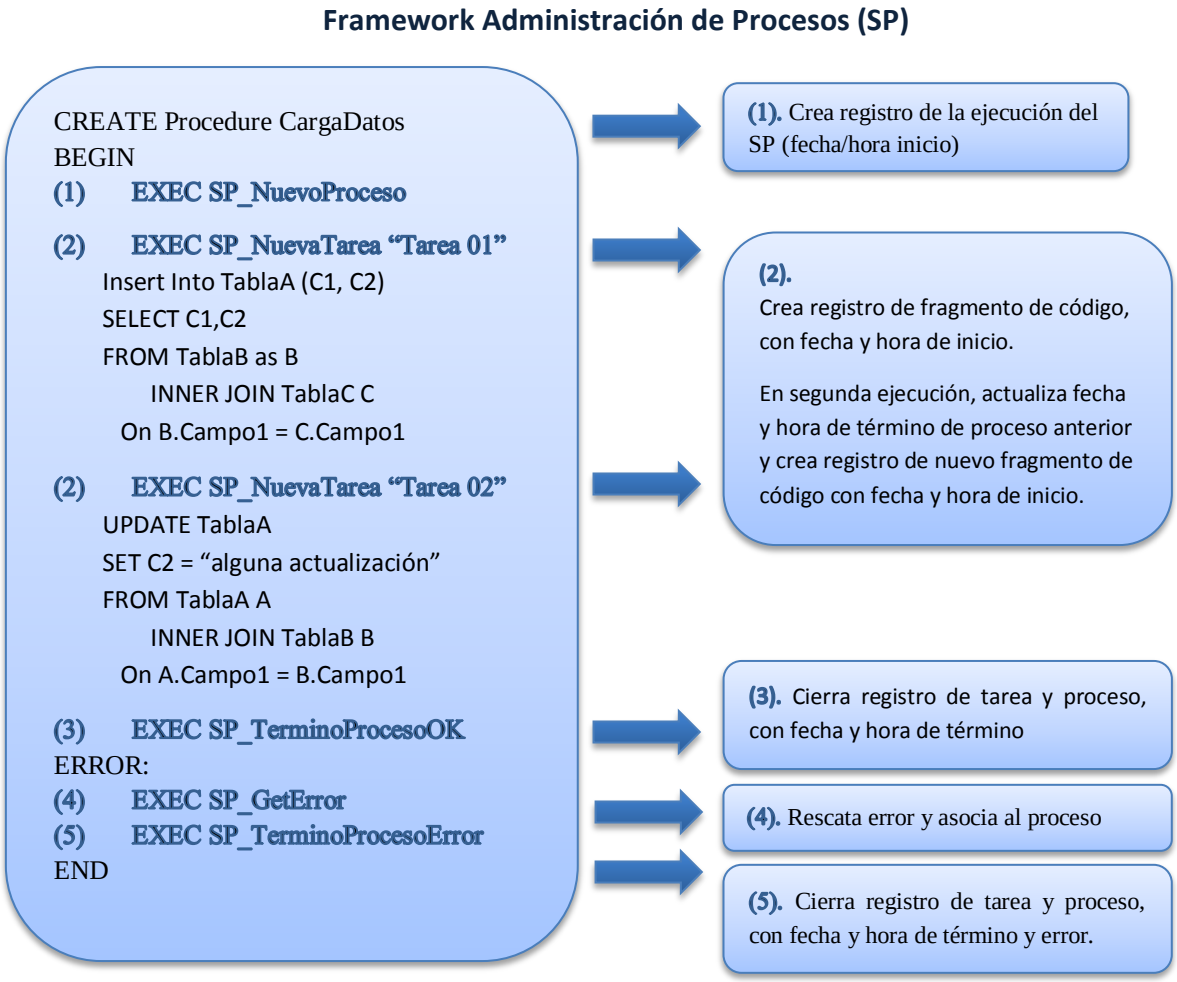


Figura 6.1: Framework de Administración de Procedimientos Almacenados.

El siguiente es un ejemplo del uso del framework en un procedimiento almacenado en SQL Server 2008R2.

Ejemplo de la utilización del Framework.

La base de datos de apoyo a los procesos se llama DMart_Control, y es una base de datos que presta servicios a todos los ambientes (Carga, ODS, OLAP), contiene información de la ejecución de los procesos, la programación de la malla y los errores.

```
CREATE PROCEDURE DemoProceso
       @descripcion           varchar(100)
      ,@Proceso_IdPadre INT
      ,@Error                       INT OUTPUT
as

BEGIN TRY
      /* --------------  I N I C I O   P R O C E S O  ------------- */
      DECLARE @Proceso_Id INT
               ,@BD_Id        INT

      SELECT @BD_Id = DB_ID()
      EXEC DMart_Control..PA_CONTROL_NuevoProceso @Proceso_Id OUTPUT,
@Proceso_IdPadre, @@PROCID, @BD_Id

      --Agrega los parámetros
      EXEC DMart_Control..PA_CONTROL_AgregaValorParam @Proceso_Id,
'@descripcion', @ValorTexto = @Descripcion
      EXEC DMart_Control..PA_CONTROL_AgregaValorParam @Proceso_Id,
'@Proceso_IdPadre', @ValorEntero = @Proceso_IdPadre
      EXEC DMart_Control..PA_CONTROL_AgregaValorParam @Proceso_Id,
'@NoExiste', @ValorEntero = 0

      /* -----------  I N I C I O   P R O C E D I M I E N T O   ---- */
      EXEC DMart_Control..PA_CONTROL_NuevaTarea @Proceso_Id, 'Descripción
Paso 1'
      --Agrega delay para efectos del ejemplo, y ver la dif. De tiempo
      WAITFOR DELAY '00:00:03'

      EXEC DMart_Control..PA_CONTROL_NuevaTarea @Proceso_Id, 'Descripción
Paso 2'
      --Agrega delay para efectos del ejemplo, y ver la dif. De tiempo
      WAITFOR DELAY '00:00:02'

      /* --------------------  F I N   P R O C E S O  ----------- */
      --Termina Proceso
      EXEC DMart_Control..PA_CONTROL_TerminoProceso @Proceso_Id
      --Agrega resultados
      EXEC DMart_Control..PA_CONTROL_AgregaValorResult @Proceso_Id,
'Resultado 01', @ValorTexto='VALOR'
      EXEC DMart_Control..PA_CONTROL_AgregaValorResult @Proceso_Id,
'Resultado 02', @ValorNumero=1000.02
END TRY
BEGIN CATCH
      /* --------  F I N   P R O C E S O   E R R O N E O  --------- */
```

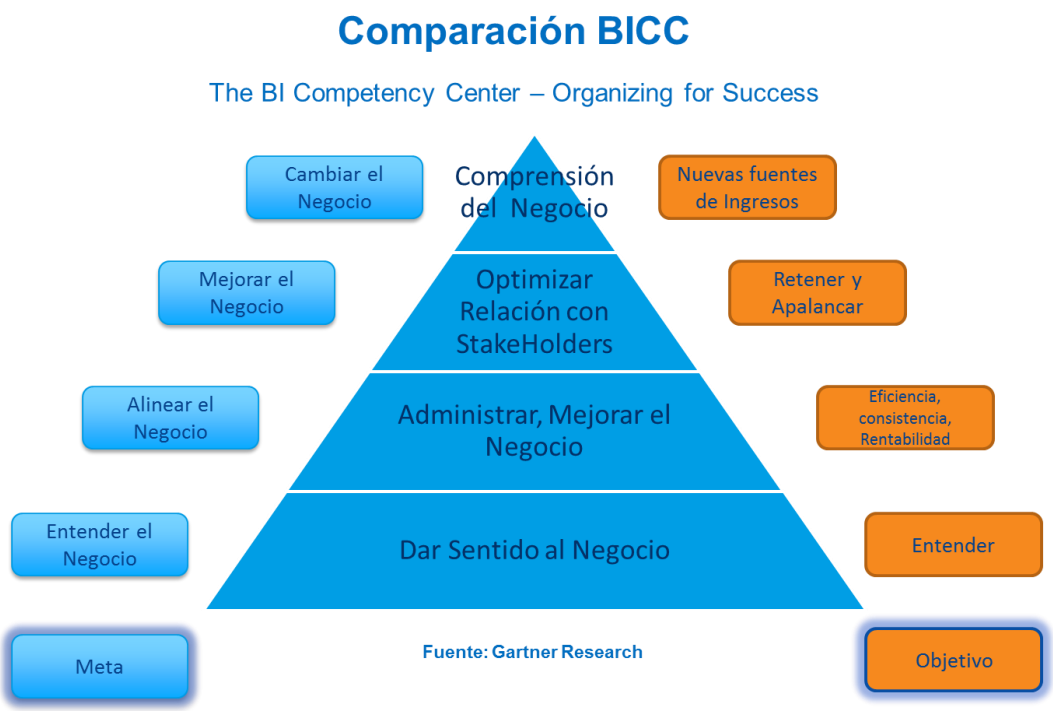
```
      EXEC DMart_Control..PA_CONTROL_GetError @BD_Id, @@PROCID, @Error
OUTPUT
      EXEC DMart_Control..PA_CONTROL_ErrorProceso @Proceso_Id, @Error

END CATCH
```

Para conseguir el código completo del framework de administración de procesos (modelo de datos y procedimientos almacenados) en SQL Server 2008R2, puedes enviar un mail solicitándolo a srodriguez@easybi.cl

Anexos B. Pirámide de desarrollo de Inteligencia de Negocios.

Gartner Research ha desarrollado una pirámide de 4 niveles que permite identificar claramente los objetivos y metas que las organizaciones deben alcanzar en cada nivel para desarrollar sus áreas de inteligencia de negocios, basadas en el concepto BICC (Business Intelligence Competency Center).

La siguiente figura ilustra los niveles de la pirámide, las metas y objetivos.



Fuente: The BI Competency Center – Organizing for Success (Nigel Rayner, 18 Sep. 2001)

Referencias.

La literatura utilizada parcialmente en este documento es la siguiente:

1. The BI Competency Center – Organizing for Success (Nigel Rayner, 18 Sep. 2001).
2. The DataWarehouse Toolkit (2a Edition, Ralph Kimball, Margy Ross).

Adicionalmente, se sugiere leer la siguiente literatura para complementar algunos conceptos de DataWarehouse no descritos por este documento.

1. Microsoft Business Intelligence: vea el cubo medio lleno (Salvador Ramos, SolidQ Press, 2011).