CONTENIDO

Capítulo 1 - Introducción al Data Warehousing

Desde su aparición, a mediados de los años '70, las bases de datos (y la teoría sobre bases de datos) no se han detenido. Las primeras versiones de las bases de datos se centraron alrededor de un único repositorio sirviendo a todos los propósitos orientados al procesamiento de la información (desde el transaccional, pasando por el procesamiento batch, hasta lo analítico). En la mayoría de los casos, el principal foco de las primeras bases de datos fueron los sistemas operacionales o transaccionales. En las últimas décadas, ha surgido una noción más sofisticada de las bases de datos. Por un lado, el objetivo de servir a las necesidades operacionales, y por otro, cubrir las necesidades analíticas de la información.

El mercado de Data Warehousing consiste de herramientas, tecnologías y metodologías que permiten la construcción, uso, manejo y mantenimiento del hardware y software usado tanto para un data warehouse como para los datos en sí mismos. Las encuestas y la realidad marcan que los proyectos de Data Warehousing (o asociados al concepto de Data Warehouse) son las mayores iniciativas después de finalizado los esfuerzos de Y2K. [Ree00a].

Los Data Warehouses (base de datos **OLAP**, on-line analytical processing) son diseñados para cumplir con un conjunto de metas, las cuales son bien diferentes de los objetivos de un sistema transaccional (**OLTP**, on-line transaction processing). Por ejemplo, una meta de los OLTP es maximizar la concurrencia mediante el uso de locks, dicho objetivo no es pertinente en el diseño de DW donde las consultas son sólo del tipo **SELECT** (por los menos en los modelos mas puros que trataremos en este trabajo)

Además de las técnicas de diseño, un desarrollador de Data Warehousing debe focalizarse en entregar un **análisis multidimensional** y capacidades de **reportes ad-hoc** (generación de reportes por parte del usuario experto basados en el conocimiento del negocio). Para realizar esto, el diseñador necesita conocer los requerimientos del negocio tan bien como las técnicas de diseño multidimensional.

Dos de los pioneros en este campo son **Bill Inmon** y **Ralph Kimball**. Inmon es universalmente reconocido con el *"padre del data warehouse"*. Tiene mas de 26 años de experiencia en el campo de las bases de datos y diseño de data warehouses, ha publicado cerca de 40 libros y más de 350 artículos en las más importantes publicaciones. Sus libros han sido traducidos a nueve idiomas. Es conocido globalmente por sus seminarios sobre desarrollos de data warehouses y ha sido orador de las más importantes asociaciones sobre el tema. Antes de fundar Pine Cone Systems, Inmon fue uno de los co-fundadores de Prism Solutions, Inc.

Ralph Kimball fue co-inventor de Xerox Star workstation, el primer producto comercial en usar iconos y ventanas. Fue vice-presidente de Metaphor Computer Systems, fundador y CEO de Red Brick Systems. Kimball es un referente de la metodología dimensional para diseñar grandes data warehouses, fue el que realmente explotó al máximo el tema de data warehousing (podríamos decir que Kimball es el *"padre"* que crió al data warehouse desde pequeño). Actualmente enseña data warehousing a diferentes grupos y ayuda a clientes con técnicas de diseño específicas. Kimball es columnista de la revista Intelligent Enterprise y tiene relación con Sagent Technology, Inc. Su libro *"The Data Warehouse Tookit"* es ampliamente reconocido como un pilar sobre la materia.

Sin lugar a dudas, el data warehousing es parte integral de lo que algunos autores definen como la *"era de la información"* [Ree00b] ya que posibilita la construcción y mantenimiento de estructuras destinadas al **análisis de los datos**, transformando los datos en **información** y la información en **conocimiento**.

En este capítulo veremos un resumen general sobre los conceptos fundamentales de Data Warehousing y Sistemas de Soporte de Decisión (DSS) comparando los sistemas OLTP (On-Line Transaction Processing) versus los sistemas OLAP (On-Line Analytical Processing).

1.1 ¿Qué es un Data Warehouse?

El término Data Warehouse fue acuñado por Bill Inmon a principios de la década de los ´90 y lo definió de la siguiente manera (dada la popularidad e importancia de esta definición preferimos no traducirla completamente): *"Un warehouse es una colección de datos **subject-oriented**, **integrated**, **time-variant** y **non-volatile** para ayudar al proceso de toma de decisiones gerenciales"* [Inm92]. Analicemos cada uno de estos términos al igual que lo hace el autor.

- **Subject-Oriented**: datos que brindan información sobre un *"sujeto"* del negocio en particular, en lugar de concentrarse en la dinámica de las transacciones de la organización.
- **Integrated**: los datos con los que se nutre el data warehouse vienen de diferentes fuentes y son integrados para dar una visión de un *"todo"* coherente.
- **Time-variant**: todos los datos en el data warehouse son asociados con un período de tiempo específico.
- **Non-Volatile**: los datos son estables en el data warehouse. Mas datos son agregados pero los datos existentes no son removidos.

Obviamente que esta definición, ya clásica, debe tomarse como la definición *"pura"* sobre data warehouse. Después de diez años, sin embargo, algunos términos han sido manejados según las necesidades y capacidades del mercado, dando origen a conceptos como el de Data Mart (para referirse a data warehouse sobre áreas específicas en lugar del warehouse corporativo) o data warehouse volátiles, que ante la imposibilidad de almacenar toda la información histórica, almacenan una foto sobre determinado períodos, etc.

Ralph Kimball define data warehouse de una forma más sencilla y práctica pero igual de importante, un data warehouse es *"una copia de los datos transaccionales específicamente estructurados para **consultas** y **análisis**"* [Kim92].

Por nuestro lado, podemos decir que un **Data Warehouse** es una **base de datos orientada al análisis** de la **información histórica** contenida en ella. Dependiendo las necesidades de análisis de la organización puede almacenarse desde unos meses hasta varios años de información. El **modelo** que soporta la información que contiene se encuentra **diseñado**, **estructurado** e **implementado** con la **finalidad y propósito del análisis y navegación** de los datos . Se entiende por navegación o **drilling** de los datos, la posiblidad de ver información correspondiente a **diferentes contextos o entornos**, por ejemplo, analizar las ventas anuales y poder *"abrirlas"* por sucursal, después analizar en más en detalle una sucursal para ver cómo se discriminan las ventas por cada producto, etc.

Según la implementación seleccionada, los datos son almacenados en forma **relacional** (RDBMS, respetando ciertos estándares a nivel de definición y consulta de datos) o en formato **multidimensional** (las bases de datos multidimensionales son arquitecturas propietarias definidas por cada proveedor que son frecuentemente actualizadas desde base de datos relacionales).

Típicamente los usuarios tienen sólo permisos de lectura sobre el warehouse (**read-only**). Comúnmente se dice que los data warehouse son **fuentes secundarias** de información pues no generan información por si mismos, sino que son actualizados desde sistemas fuentes existentes **internamente** en la organización (sistema de ventas, sistema presupuestario, etc.) o sistemas **externos** de información (datos meteorológicos, información de la competencia, cotizaciones de la bolsa, etc.).

Por todo esto los data warehouses son identificados como ambientes **OLAP**, On-Line **Analytical** Processing, en contraposición a los ambientes transaccionales clásicos (**OLTP**, On-Line **Transaction** Processing) [Bed97].

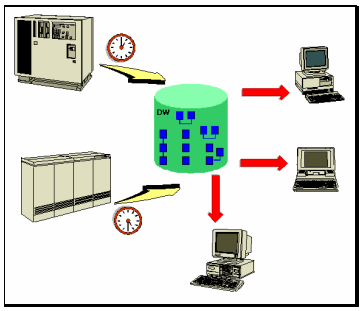

Fig. 1-1 ¿Qué es un Data Warehouse?

1.2 DSS (Decision Support Systems)

El **DW** es el ***"corazón"*** de la arquitectura de un Sistema de Soporte de Decisiones (**DSS**). Es la parte fundamental del funcionamiento de un DSS, dado que es una única fuente integrada de datos y los mismos, dentro del DW, son realmente accesibles permitiendo al analista del negocio trabajar en un ambiente inmensurablemente más fácil que en un ambiente clásico transaccional.

El Sistema de Soporte de Decisiones provee los mecanismos de acceso a los datos y el análisis de los mismos. El DSS Engine es el ***"cerebro"*** de la arquitectura, es el motor que **traducirá** los requerimientos de los usuarios en las correspondientes sentencias de consulta para el data warehouse y el que **interpretará** los resultados devueltos por el warehouse para mostrarlos según lo solicitado por el usuario. Realmente es el componente que concentra la **inteligencia del sistema**, pues diferentes DSS Engine pueden cambiar notablemente la explotación y el comportamiento de un mismo warehouse.

El sistema asiste a los tomadores de decisiones proveyendo varios tipos de análisis como ser, reportes de tendencias, de comparación y análisis ad-hoc. Podemos decir que un Sistema de Soporte de Decisiones basado en técnicas de Data Warehousing transforma los **datos** que tiene una organización en **información**, con el objetivo de tomar mejores decisiones de negocio basados en un análisis dimensional.

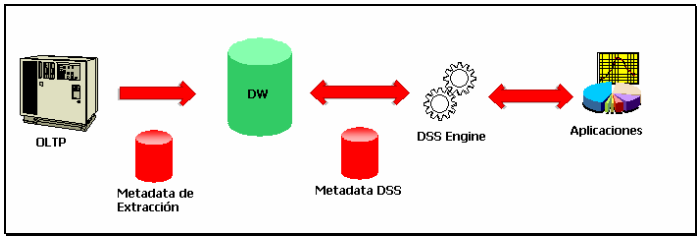

Fig. 1-2 Arquitectura de un DSS

Es importante aclarar que durante este trabajo nos concentraremos en **DSS basados en técnicas de Data Warehousing**, pero ambos conceptos tienen vida propia. Durante mucho tiempo las organizaciones trabajaban con Sistemas de Soporte de Decisión pero no tenían el **volumen ni la calidad** de información que brinda un Data Warehouse. Típicamente, cada analista juntaba información de diferentes fuentes en alguna planilla de cálculo y en base a ciertas fórmulas y macros tomaba decisiones sobre políticas de costos, precios, etc. Por otro lado, los warehouses pueden ser utilizados en sistemas expertos que guardar información histórica con el fin del **aprendizaje** de un sistema (p.e. un robot que circula por diferentes laberintos y por cada iteración –recorrida del laberinto desde la entrada hasta la salida- va *"aprendiendo"* en base a la experiencia que le brinda la información histórica –data warehouse-)

1.3 OLTP vs. OLAP

Los sistemas operacionales y los DWs difieren ambos en **diseño** y **funcionalidad**. Entre los numerosos puntos de diferencias se encuentran, los **objetivos** principales de construcción, la **orientación** o alineación de los datos, la **integración**, la **historicidad**, el **acceso de datos** y **manipulación**, los **patrones de uso**, la **granularidad** de los datos, el **perfil de los usuarios**, el **ciclo de vida**, etc.

En ocaciones, un sistema fuente puede actuar también como una base de datos para el soporte de decisión, pero este caso no elimina las diferencias en la naturaleza de una aplicación OLAP y un sistema OLTP, de hecho posiblemente las resalte. Es importante detenerse sobre las diferencias entre ambos ambientes pues, la mejor manera de entender OLAP, es entender las diferencias con los sistemas transaccionales tradicionales [Bed97].

1.3.1 Objetivos principales

Los **OLTP** tienen como objetivos principales asistir a **aplicaciones específicas**, p.e. ATM, y mantener integridad de los datos. Mientras que los **OLAP** apuntan a asistir en el **análisis del negocio**, identificando tendencias, comparando períodos, gestiones, mercados, índices, etc. mediante el almacenamiento de datos históricos [Bed97].

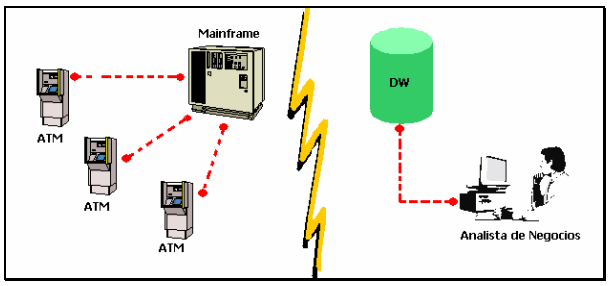

Fig. 1-3 OLTP vs. OLAP. Objetivos

1.3.2 Alineación de los datos

Los **OLTP** están alineados **por aplicación**. Diferentes sistemas tienen distintos tipos de datos, los cuales son estructurados por aplicación. Se focaliza en el cumplimiento de requerimientos de una aplicación en especial o una tarea específica.

En cambio, los sistemas **OLAP** están alineados **por dimensión**. Todos los tipos de datos integrados en un solo sistema. Los datos son organizados definiendo dimensiones del negocio (áreas temáticas o sujetos). Se focaliza en el cumpliento de requerimientos del análisis del negocio.

Los distintos mercados verticales tienen diferentes perspectivas desde los ambientes transaccionales y desde los ambientes analíticos, p.e., en el mercado bancario existen numerosas aplicaciones de cuentas y préstamos a nivel operacional mientras que en un data warehouse la información estaría organizada por cliente, tipo de cuenta y tiempo. En los retails hay aplicaciones sobre registro de ventas, manejo de inventario o stock y presupuesto y en el ambiente de data warehousing hablaríamos de productos, sucursales, tiempo y las diferentes variables propias de negocio (unidades vendidas, monto neto vendido, monto bruto vendido, precio promedio, unidades presupuestadas, cantidad de piezas en stock, etc.) [Bed97].

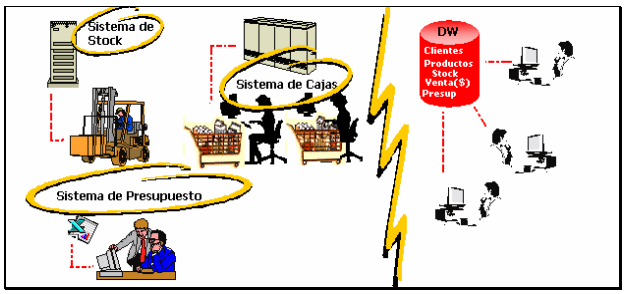

Fig. 1-4 OLTP vs OLAP. Alineación de los datos

1.3.3 Integración de datos

En los **OLTP**, los datos se encuentran típicamente **no integrados**, son calificados como **datos primitivos** o **datos operacionales**. Los mismos son estructurados independientemente uno de otros, pudiendo tener diferentes estructuras de claves y convenciones de nombres. Son usualmente almacenados en diferentes formatos de archivos, p.e. relacional, VSAM, archivos planos,etc..Incluso, si todos los datos están en formato relacional, los mismos pueden residir en diferentes plataformas de hardware y en distintas RDBMSs.

Sin embargo, en los ambientes **OLAP**, los datos deben estar **integrados**. Son conocidos como **datos derivados** o **datos DSS** dado que provienen de sistemas transaccionales o sistemas de archivos maestros preexistentes en las mismas organizaciones o de sistemas externos de información. El DW, con el objetivo de alinear los datos por áreas temáticas, debe integrar datos operacionales estandarizando estructuras y convenciones de nombres (concepto de diccionario de datos).

Este item da paso a otro importante tema de interés en la actualidad, **Data Cleansing**, que se refiere al reformateo de datos para conformar los estándares del warehouse. La limpieza debe ser realizada por los programas de extracción para asegurar que los datos estén en un formato consistente antes de ser cargados dentro del DW [Bed97].

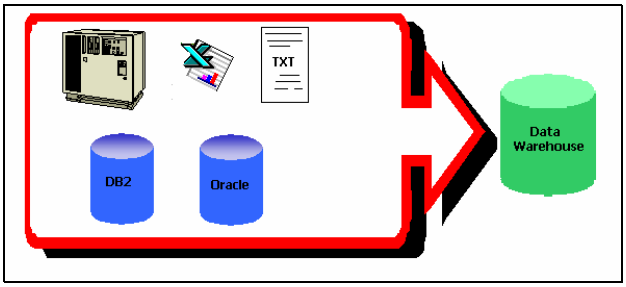

Fig. 1-5 OLTP vs. OLAP. Integración de datos

1.3.4 Historia

Los **OLTP** usualmente retienen datos para **60** a **90 días**, después son resguardados por los administradores de base de datos en almacenamientos secundarios **fuera de línea** (p.e. cintas o en disco a nivel de back up). También es común que contengan sólo **valores corrientes**, p.e., el actual balance de cuentas para clientes y no valores históricos. Puede no incluir el tiempo como un componente de la clave. Por ejemplo, sólo el balance corriente de cuentas es almacenado, por lo tanto, no tiene sentido guardar el tiempo como parte de la clave de los datos.

En cambio los **OLAP** almacenan tanta **historia** como sea necesario para el análisis del negocio, típicamente **dos a cinco años** de datos históricos. Retienen valores para cada período (el atributo más atómico de la dimensión tiempo) en la base de datos. Es decir que almacenan una serie de fotos instantáneas de datos operacionales, la frecuencia con la cual define el nivel de detalle es la que se indica en la correspondiente hoja de la dimensión tiempo. Toda esta cantidad y tipo de historia apunta a ayudar a la generación de reportes de comparación de tendencias y períodos de tiempo.

Por otro lado, las bases de datos orientadas al análisis siempre contienen el **tiempo** como clave dado que una de las principales razones para la construcción del warehouse es el alamacenamiento de **datos históricos** y el **análisis** a lo largo del tiempo [Bed97].

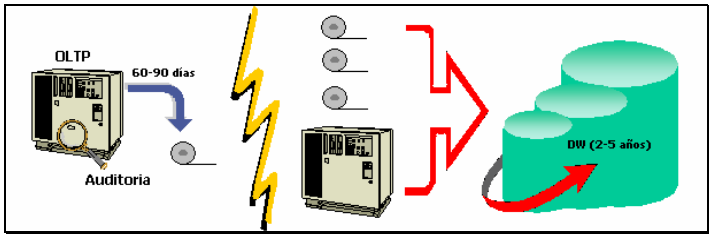

Fig. 1-6 OLTP vs. OLAP. Historia

1.3.5 Acceso y manipulación de los datos

Los sistemas operacionales realizan una manipulación de datos **registro por registro** con grandes cantidades de **inserts**, **updates** y **deletes**. Además necesitan de rutinas de validación y transacciones a nivel registro (OLTP on-line TRANSACTION processing). Generalmente poseen pequeñas cantidades de datos involucrados en un solo proceso o transacción y la puesta a punto de la base de datos para el procesamiento de transacciones, se focaliza en mecanismos de locking y asignación de recursos (tuning específico)

En cambio, los DWs tienen una carga y **acceso masivo de datos**, no se realizan inserts, updates o deletes. La **carga** y **refresco** es **batch** (lo que se conoce como proceso **BULK COPY**). La validación de datos se realiza antes o después de la carga. (nunca a nivel registro o transacción). Principalmente se realizan sentencias de **SELECT** sobre varios registros y tablas (OLAP on-line ANALYTICAL processing), teniendo grandes volúmenes de datos involucrados en un único proceso o análisis [Bed97]. Es por ello que generalmente no se respetan las formas normales tan necesarias en los sistemas operacionales clásicos. Las anomalías que tienden a subsanar estas reglas de normalización no se presentan en los sistemas OLAP donde la carga de la información está automatizada y puede permitirse el manejo de redundancia controladada como punto para la mejora de los tiempos de respuesta de las consultas a la base de datos.

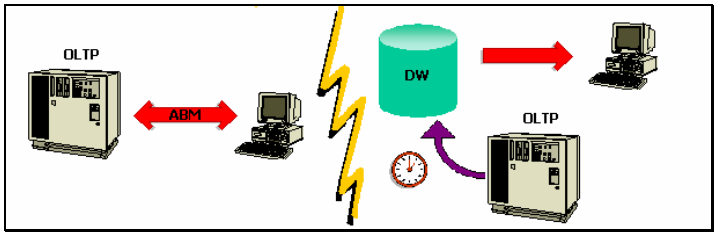

Fig. 1-7 OLTP vs. OLAP. Acceso y manipulación de los datos

1.3.6 Patrones de uso

Los sistemas transaccionales normalmente mantienen un patrón de uso **constante** requiriendo grandes cantidades de recursos y consumiendo sólo el tiempo referido a la transacción.

En contraposición, los warehouses tienen un patrón de uso **liviano con picos de usos** eventuales en el tiempo (afectados por la disponibilidad de los datos y el flujo de trabajo del negocio). Los picos de uso suceden el mismo dia de cada semana y el mismo dia de cada mes (cuando los datos están por primera vez disponibles o cuando el negocio necesita por primera vez un reporte) [Bed97].

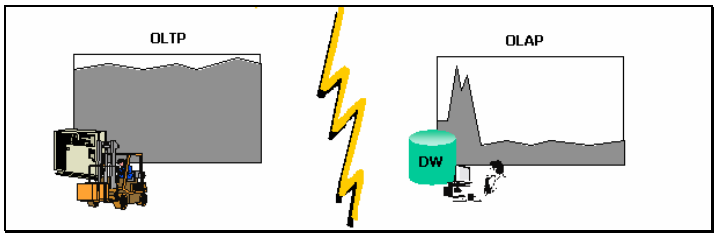

Fig. 1-8 OLTP vs. OLAP. Patrones de uso

1.3.7 Granularidad de los datos

En los sistemas operacionales se encuentran los datos a **nivel detallado o nivel transaccional**. Una transacción incluye a nivel atómico cada uno de los componentes de su estructura (fecha, hora, código de cliente, código de movimiento, importe, etc.)

En un comienzo, los DW contenían **información sumarizada** hasta cierto nivel (el permitido por los volúmenes de consolidación de los cubos multidimensionales de la década de los '80). Con el avance en el desarrollo de las bases de datos durante la década de los '90 se pudo incorporar la posibilidad de llegar incluso a nivel de detalle (**nivel de transacción**) en los data warehouses. (Ver más adelante evolución de los data warehouses, MOLAP – HOLAP – ROLAP).

La diferencia en la granularidad de los datos viene dada por el uso de los mismos. Si bien un data warehouse puede tener información a nivel transaccional, el objetivo de esta granularidad mínima está asociado con el deseo de realizar ciertos tipos de análisis que requieren que la información esté a es ese nivel de detalle (análisis de **market basket**), pero no significa que veamos la información a nivel transaccional.

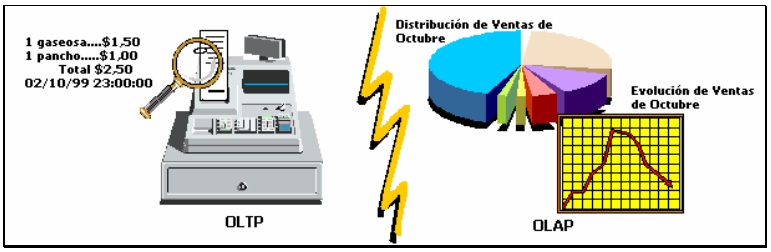

Fig. 1-9 OLTP vs. OLAP. Granularidad

1.3.8 Perfil de usuario

Dado que los OLTP tienen como objetivo asistir a **aplicaciones específicas** y asegurar la integridad de los datos, el perfil de usuario que interactúa con dichos sistemas se encuadra dentro de los empleados operacionales de una organización (**comunidad operativa**).

Por el contrario, dado el **objetivo estratégico** y el nivel de información que manejan los DW, el perfil de usuario sobre este tipo de sistemas corresponde a la **comunidad gerencial**, la cual esta a cargo de la toma de decisiones.

Fig. 1-10 OLTP vs. OLAP. Perfil de usuario

1.3.9 Ciclo de vida

Los ambientes operacionales pueden ser desarrollados por el clásico **SDLC** (Ciclo de vida del desarrollo de sistemas). Según señalaba Inmon [Inm92], el DW operaba bajo un ciclo de vida bastante diferente, a veces denominado **CLDS** (el inverso de SDLC). El clásico SDLC es guiado por los requerimientos. En una etapa posterior se comienza con el diseño y luego con el desarrollo. El CLDS es casi la inversa, comienza con los datos. Una vez que se identifica a los datos, los mismos son integrados y luego testeados. Más tarde son desarrolladas las aplicaciones de explotación y finalmente son atendidos los requerimientos de consulta de los usuarios. Dado el particular flujo del ciclo de vida, se lo suele llamar **data-driven** (guiado por los datos) en contraposición al tradicional guiado por los requerimientos (**requirement-driven**) del SDLC. Ejemplos de esta métodología de trabajo puede también encontrarse en [Gol99] donde se propone un enfoque metodológico al estilo de Inmon denominado *Dimensional Fact Model* (DFM), que comienza con el análisis de los sistemas de información y luego se cubre los requerimientos de los usuarios en base a la información disponible en los sistemas fuentes.

Esta visión de Inmon, ha cambiando bastante en los últimos años. Cada vez más se le da importancia a lo que se conoce como **query profile**, el perfil de consulta de los usuarios, es decir, los futuros análisis que harán y cuáles son los requerimientos para hacer un análisis provechoso que ayuden a cumplir con los objetivos de negocio marcados por las organizaciones. Si bien los datos disponibles tienen un valor fundamental en el modelado y diseño del DW, en esta nueva visión se lo utiliza más para **contrastar** con los requerimientos relevados con los usuarios más que como fuente única de modelado. Los requerimientos del negocio son el ***"centro"*** del data warehouse [Kim98].(Ver más adelante Ciclo de Vida del DataWarehousing **BDL** propuesto por Ralph Kimball [Kim98])

Es importante destacar también que el ciclo de vida de un DW es **evolutivo y cíclico** ajustándose al ciclo de vida espiral aplicado en otros ambientes de desarrollo. Es por esta y otras razones que comúnmente se dice que **Data Warehousing es un proceso**.

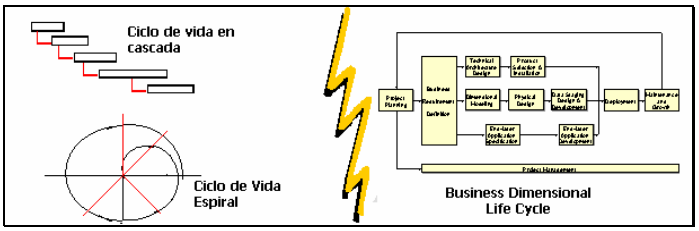

Fig. 1-11 OLTP vs. OLAP. Ciclo de Vida

Como explica Roger Pressman [Pres93], el paradigma del modelo espiral para la ingeniería del software es actualmente el enfoque más realista para el desarrollo de sistemas de gran escala pues permite al desarrollador y al cliente entender y reaccionar a los riesgos en cada nivel evolutivo utilizando la creación de prototipos, en cualquier etapa de evolución, como un mecanismos de reducción de riesgos.

Capítulo 2 - El Ciclo de Vida Dimensional del Negocio (BDL)

Tomaremos como base el ciclo de vida de los Data Warehouses definido por Ralph Kimball [Kim98]. El marco presentado por Ralph Kimball con el nombre de Business Dimensional Lifecycle (BDL) ilustra las diferentes etapas por las que debe pasar todo proceso de Data Warehousing. Este enfoque de implementación de data warehouses es ilustrado en la siguiente figura. Este diagrama ilustra la secuencialidad de tareas de alto nivel requeridas para el efectivo diseño, desarrollo e implementación de data warehouses. El diagrama muestra una vista general del mapa de ruta de un proyecto en el cual cada rectángulo es un mojón que nos indica dónde estamos parados, por dónde pasamos y hacia dónde debemos dirigirnos.

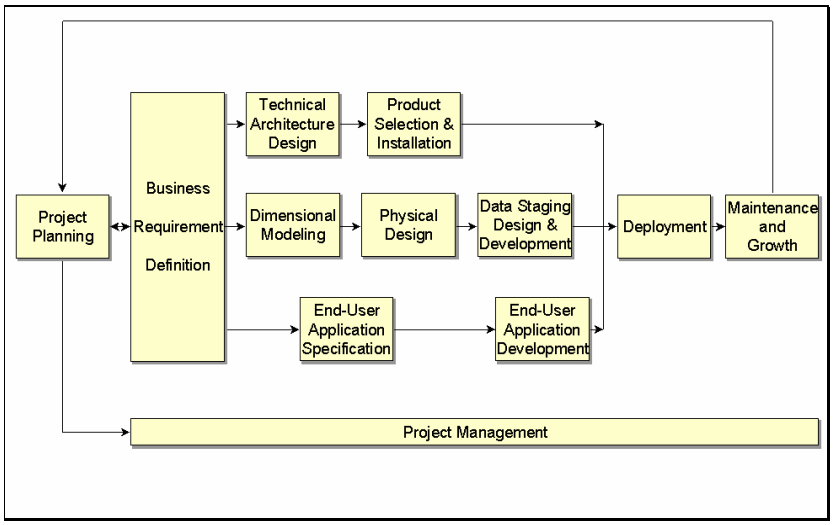

Fig. 2-1 Business Dimensional Lifecycle propuesto por Ralph Kimball

Es importante aclarar, como lo hacen los autores, que el BDL no intenta reflejar un proyecto en término de tiempos y plazos. Como se puede notar cada rectángulo del diagrama tiene el mismo ancho, con la excepción del gerenciamiento del proyecto. Cualquiera que haya pasado por algún proyecto de Data Warehousing sabe que la magnitud de recursos y tiempo requerido para cada rectángulo del ciclo de vida no es igual. El BDL se focaliza en secuencialidad y concurrencia no en tiempos y plazos.

A continuación pasaremos de describir cada una de las etapas del BDL.

2.1 Planificación del Proyecto

La planificación busca identificar la definición y el alcance del proyecto de data warehouse, incluyendo justificaciones del negocio y evaluaciones de factibilidad.

La planificación del proyecto se focaliza sobre recursos, perfiles, tareas, duraciones y secuencialidad. El plan de proyecto resultante identifica todas las tareas asociadas con el BDL e identifica las partes involucradas.

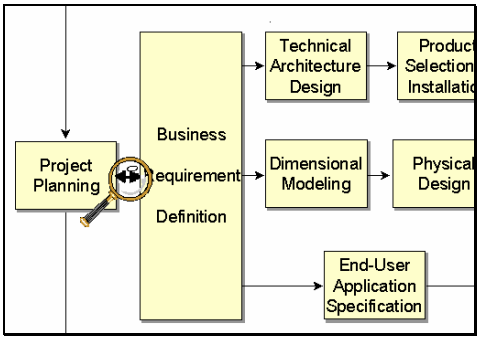

Fig. 2-2 Iteracción entre Planficación y Requerimientos

La planificación del proyecto es dependiente de los requerimientos del negocio, como es denotado en el diagrama del BDL, ya que los requerimientos del negocio determinan el alcance del proyecto, defininen los recursos necesarios, etc y la planificación acotará los requerimientos ya sea por cuestiones de recursos y/o tiempo.

Esta etapa se concentra sobre la **definición** del proyecto (identificación del escenario del proyecto para saber de dónde surge la necesidad del data warehouse). Según sentencia Kimball, *"Antes de comenzar un proyecto de data warehouse o data mart, hay que estar seguro si existe la demanda y de dónde proviene. Si no se tiene un sólido usuario sponsor y no hay usuarios entusiasmados, posponga el proyecto"*. Factores asociados con estas etapas incluyen: identificación de los usuarios sponsors, convincentes motivaciones del negocio, cooperación entre áreas de sistemas y negocios, cultura analítica de la organización y análisis de factibilidad (tanto tecnológica como de disponibilidad de datos). Para medir estos factores propone un test de buena disposición del proyecto dónde describe diferentes escenarios posibles [Kim98]. Adicionalmente propone técnicas (relevamientos de alto nivel, priorización de requerimientos y pruebas de concepto) para mitigar las deficiencias que el proyecto pudiera tener en algunos de los factores mencionados anteriormente.

Cómo metodología de estas etapas propone identificar el alcance preliminar basándose en los **requerimientos del negocio** y no en fechas límites (deadlines) construyendo la justificación del proyecto en términos del negocio con indicadores como el **ROI** (retorno de inversión), **NPV** (valor presente neto) y el **IRR** (índice de retorno interno).

A nivel de planificación del proyecto, establece la identidad del mismo, el personal (staff): los usuarios sponsors, lideres, gerentes del proyecto (tanto de sistemas como del sector usuarios), equipo *"corazón"* del proyecto (analistas, arquitectos, DBAs, diseñadores, responsables de extracción, desarrolladores, instructores, etc.), equipo "especial" del proyecto (soporte, seguridad informática, programadores, analistas QA), el desarrollo del plan del proyecto, el seguimiento y monitoreo.

2.2 Definición de los Requerimientos del Negocio

Un factor determinante en el éxito de un proceso de Data Warehousing es la interpretación correcta de los diferentes niveles de requerimientos expresados por los diferentes niveles de usuarios. La técnica utilizada para relevar los requerimientos de los analistas del negocio difiere de los enfoques tradicionales guiados por los datos [Inm92] [Gol99].. Los diseñadores de los data warehouses deben entender los factores claves que guían al negocio para determinar efectivamente los requerimientos y traducirlos en consideraciones de diseño apropiadas.

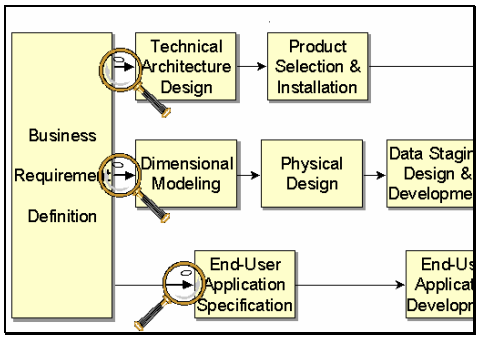

Fig. 2-3 Iteracción entre Requerimientos del Negocio y etapas subsiguientes

La definición de los requerimientos del negocio establece la base para las tres etapas paralelas subsiguientes focalizadas en la tecnología, los datos y las aplicaciones por lo cual es altamente crítica y es el centro de atención del BDL.

Los usuarios finales y sus **requerimientos** impactan **siempre** en las implementaciones realizadas de un data warehouse. Según la perspectiva de Kimball [Kim98], los requerimientos del negocio se posicionan en el **centro** del *"universo del data warehouse"*. Como destaca siempre el autor, los **requerimientos del negocio deben determinar el alcance** del data warehouse (qué datos debe contener, cómo debe estar organizado, cada cuánto debe actualizarse, quiénes y desde dónde accederán, etc).

Kimball da consejos y técnicas para descubrir eficazmente los requerimientos del negocio. Estas tácticas y estrategias se focalizan sobre las entrevistas de relevamiento (diferentes tipos, preparación de la entrevista, roles a cubrir, búsqueda de información pre-entrevista, selección de entrevistados, desarrollo de los cuestionarios, planificación, preparación de los entrevistados, conducción de la entrevista, contenido, cierre, revisión de resultados, etc.).

2.3 Modelado Dimensional

La definición de los requerimientos del negocio determina los datos necesarios para cumplir los requerimientos analíticos de los usuarios. Diseñar los modelos de datos para soportar estos análisis requieren un enfoque diferente al usado en los sistemas operacionales. Básicamente se comienza con una matriz donde se determina la dimensionalidad de cada indicador y luego se especifican los diferentes grados de detalle (atributos) dentro de cada concepto del negocio (dimensión), como así también la granularidad de cada indicador (variable o métrica) y las diferentes jerarquías que dan forma al modelo dimensional del negocio (BDM) o mapa dimensional.

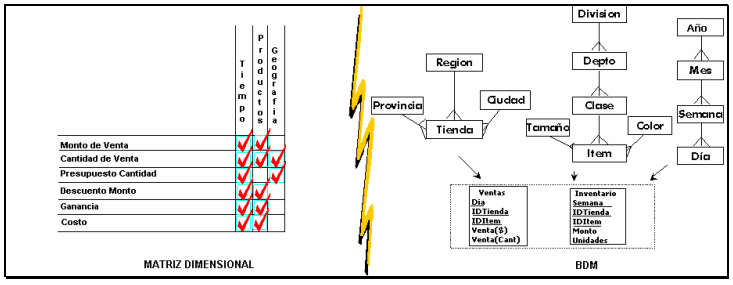

Fig. 2-4 Diagramas para Modelado Dimensional

Matriz Dimenisonal que determina la dimensionalidad de cada indicador y modelo dimensional del negocio (BDM) que ilustra las diferentes jerarquías dentro de cada dimensión.

Ralph Kimball es realmente un referente en el tema de modelado dimensional como lo demuestran sus numerosos papers y libros publicados al respecto. Gran parte de las técnicas básicas en este tema (identificación de dimensiones, atributos, star schemas, snowflake schemas, soluciones verticales, etc.) son tratadas e introducidas en [Kim92] constituyendo la base de la teoría sobre modelado dimensional.

En [Kim98] Kimball dedica tres capítulos a la etapa del modelado dimensional identificada como una de las etapas siguientes al relevamiento de requerimientos del negocio del BDL. En el primer capítulo sobre esta etapa (Capítulo 5, A First Course on Dimensional Modeling) introduce, reafirma y justifica algunos de los conceptos desarrollados en [Kim92] comparando el BDM con ERM, analizando algunos *"mitos"* sobre el modelado dimensional, describiendo las diferentes arquitecturas de construcción, esquemas de modelos y de fact tables, definiendo el concepto de Data Mart, introduciendo conceptos nuevos sobre arquitecturas **BUS**, conformed dimensions, standard facts y explicando las técnicas básicas del modelado dimensional.

En el capítulo siguiente de [Kim98] (Capítulo 6, A Graduate Course on Dimensional Modeling) se introducen conceptos más avanzados del modelado, tales como, relaciones muchos a muchos en esquemas estrella, role-playing dimensions, relaciones recursivas, manejo de granularidades diferentes, múltiples unidades de medida, modelos multimoneda, bandas de rangos, consultas ROLAP avanzadas, análisis market basket, atributos puercoespín, etc.

Finalmente, en el último capítulo dedicado a esta etapa (Capítulo 7, Building Dimensional Models), Kimball propone una metodología para construir modelos dimensionales en ambientes reales. Comienza fijando los prerequisitos para realizar el **BDM**, después identifica los correspondientes Data Marts (como conjunto de indicadores), las correspondientes dimensiones, luego sigue con la construcción de una matriz dimensional de alto nivel (**Data Warehouse Bus Architecture Matrix**), donde las filas son los Data Marts y las columnas son todas las dimensiones marcando la intersección de aquellas dimensiones que están en juego en determinados Data Marts. La metodología continúa con el diseño de cada Data Mart, teniendo como primer paso la **elección del Data Mart**, segundo la **declaración de la granularidad**, tercero la **elección de las dimensiones** y cuarto la **elección de las facts** o indicadores. Además, Kimball recomienda un conjunto de gráficos y diagramas que ayudarán a esta etapa del proyecto (Data Warehouse Bus Architecture Matrix, diagrama de Fact Table, detalle de Fact Table y diagrama de Dimensiones). Finalmente, Kimball propone una serie de pasos y pautas para el equipo de trabajo asignado al modelado dimensional que ayuda a identificar y resolver futuros problemas de forma temprana, como así también el análisis de alto nivel de los sistemas fuentes como para mejorar las estimaciones y alcances del modelo realizado.

2.4 Diseño Físico

El diseño físico de las base de datos se focaliza sobre la selección de las estructuras necesarias para soportar el diseño lógico. Algunos de los elementos principales de este proceso son la definición de convenciones estándares de nombres y seteos específicos del ambiente de la base de datos. La indexación y las estrategias de particionamiento son también determinadas en esta etapa.

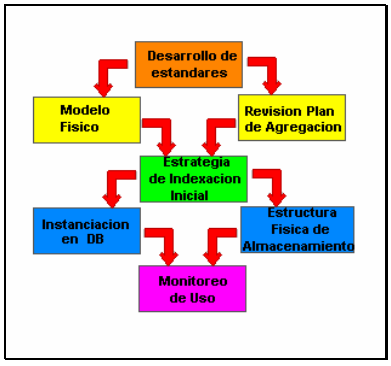

Fig. 2-5 Proceso de diseño físico a alto nivel

Kimball dedica dos capítulos al tema del diseño físico, uno especialmente para las agregaciones (Captítulo 14, [Kim98]) por considerarlas como la técnica de tuning de **mejor impacto** sobre la performance de las consultas efectuadas por los usuarios (según su experiencia un factor de 100 o incluso de 1000) y otro capítulo más genérico (Captítulo 15, [Kim98]) donde resalta algunos puntos tales como: planes de desarrollo del modelo físico, convenciones de nombres y estándares, uso de sinónimos, uso de herramientas de modelado para la generación y mantenimiento del modelo físico, estimaciones de volúmenes, planes y estrategias de indexacción, consideraciones sobre memoria y tamaño de bloque, parametrización, partitioning, sistemas de tolerancia a fallas, monitoreo, etc.

Uno de los problemas inherentes al modelado dimensional y a su implementación relacional es el llamado **n-way join**, el hecho de tener que hacer **join de cada lookup contra la BT de una por vez**. En algunos RDBMS esto se resuelve mediante el seteo de **STAR JOIN** en el motor, mejorando el rendimiento en unas **60 veces** por sobre la utilización del join secuencial en un equipo de iguales características [Kim98]. El Star Join realiza todas las combinaciones posibles entre del criterio de selección fijado y con este conjunto realiza el Join con la BT correspondiente. Dado que esta técnica demanda la realización de un **producto cartesiano**, algunos motores no la provee pues en teoría el producto cartesiano no es recomendable, pero en el caso de un modelo dimensional es mucho más eficiente que hacer el join entre lookups y BT una a una. No sólo para el acceso a los indicadores o métricas se necesita la optimización del join, la consulta de todo modelo dimensional involucra la necesidad de la utilización de esta sentencia altamente costosa en los RDBMS, pues constantemente se está navegando entre los datos para tener diferentes vistas de la evolución y diferentes entornos de análisis de los indicadores del negocio.

2.5 Diseño y Desarrollo de Presentación de Datos

Esta etapa es típicamente la más subestimada de las tareas en un proyecto de data warehouse. Las principales subetapas de esta zona del ciclo de vida son: la extracción, la transformación y la carga (ETL process). Se definen como procesos de extracción a aquellos requeridos para obtener los datos que permitirán efectuar la carga del Modelo Físico acordado. Así mismo, se definen como procesos de transformación los procesos para convertir o recodificar los datos fuente a fin poder efectuar la carga efectiva del Modelo Físico. Por otra parte, los procesos de carga de datos son los procesos requeridos para poblar el Data Warehouse.

Todas estas tareas son altamente críticas pues tienen que ver con la materia prima del data warehouse: los datos. La desconfianza y pérdida de credibilidad del data warehouse serán resultados inmediatos e inevitables si el usuario choca con información inconsistente. Es por ello que la calidad de los datos es un factor determinante en el éxito de un proyecto de data warehousing. Es en esta etapa donde deben sanearse todos los inconvenientes relacionados con la calidad de los datos fuente.

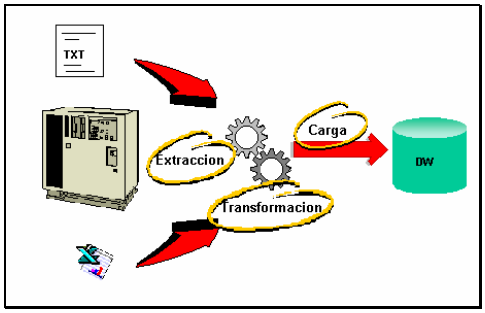

Fig. 2-6 Proceso ETL

Como advierte Kimball [Kim98], el proceso de Data Staging es el **iceberg** de un proyecto de datawarehousing. Son muchos los desafíos que deben enfrentarse para lograr datos de alta calidad de los sistemas fuentes. Como comentamos anteriormente, en general es una de las etapas más subestimadas, que siempre termina tomando más tiempo del previsto. Ralph Kimball propone entonces un plan de 10 items que ayudarán a guiar esta etapa del BDL.

Plan:

1. Crear un diagrama de flujo fuente-destino esquemático, de una página y de muy alto nivel.
2. Probar, elegir e implementar una herramienta de data staging.
3. Profundizar en detalle por tabla destino, gráficamente describir las reestructuraciones o transformaciones complejas. Gráficamente ilustrar la generación de las claves surrogadas. Desarrollo preliminar de la secuencialidad de los trabajos.

Carga de dimensiones:

4. Construir y probar la carga de una tabla dimensional estática. La principal meta de este paso es resolver los problemas de infraestructura que pudieran surgir (conectividad, transferencia, seguridad, etc.)
5. Construir y probar los procesos de actualización de una dimensión.
6. Construir y probar las cargas de las restantes dimensiones.

Fact Tables y automatización:

7. Construir y probar la carga histórica de las fact tables (carga masiva de datos). Incluyendo búsqueda y sustitución de claves.
8. Construir y probar los procesos de cargas incrementales.
9. Construir y probar la generación de agregaciones.
10. Diseñar, construir y probar la automatización de los procesos.

2.6 Diseño de la Arquitectura Técnica

Los ambientes de data warehousing requieren la integración de numerosas tecnologías. Se debe tener en cuenta tres factores: los requerimientos del negocio, los actuales ambientes técnicos y las directrices técnicas estratégicas futuras planificadas para de esta forma poder establecer el diseño de la arquitectura técnica del ambiente de data warehousing.

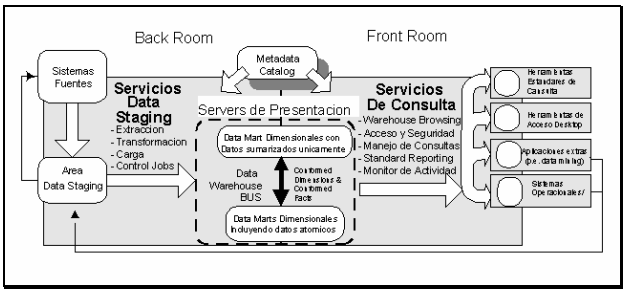

Fig. 2-7 Modelo de Arquitectura Técnica a alto nivel

Ralph Kimball hace una analogía entre los planos arquitectónicos de una casa y la arquitectura de un warehouse, nadie comenzaría diciendo, *"Hey, tengo algo de madera y algo de concreto, vamos a construir una casa!"*. Hay que tener un **plan** antes de comenzar, no es simplemente reoordenar y explotar la información. Al igual que en una contrucción, los planos sirven para comunicar los deseos entre los clientes y el arquitecto, como así también para medir esfuerzos y materiales necesarios para la obra (comunicación, planificación, flexibilidad y mantenimiento, documentación, productividad y reuso). Finalmente, argumenta Kimball, *"un buen conjunto de planos, como cualquier buena documentación, nos ayudará más tarde cuando sea tiempo de remodelar o hacer incorporaciones"*.

Kimball dedica seis capítulos a la arquitectura de un Data Warehouse (track técnico del BDL). Una de las ventajas que trae la utilización de una arquitectura BUS, como la propuesta por el autor, es la fácil incorporación de nuevas instancias a la arquitectura (**plug and play**). Entre los muchos ensayos y enfoques existentes para la definición y desarrollo de arquitecturas de sistemas, Kimball se basa en el Framework propuesto por John A. Zachman de Zachman International y lo adapta a un ambiente de datawarehousing, simplificándolo considerablemente ya que en este tipo de ambientes, en teoría, no debe ser atacado los problemas de infraestructura de los procesos transaccionales. El **Data Warehouse Architecture Framework** propuesto por Kimball incluye tres áreas (columnas): el área de arquitectura de datos (qué), el área de arquitectura técnica (cómo) y el área de infraestructura (dónde). A su vez, cada una de estas áreas tienen diferentes niveles de detalles (filas): Nivel de Requerimientos del Negocio, Nivel de Modelo Arquitectónico, Nivel de Modelo Detallado, Nivel de Implementación. Como podemos inferir de las identificaciones de los diferentes niveles, Kimball recomienda un enfoque **top-down**, comenzando con una visión global y dividiendo la arquitectura en pequeñas piezas hasta llegar al grado donde las piezas pueden ser realmente implemementadas.

Como siempre el énfasis está puesto en los **requerimientos del negocio**, son nuestra guía primaria para desarrollar nuestra arquitectura y para priorizar. La arquitectura técnica se divide en dos partes, el **back room** (la parte interna del warehouse) y el **front room** (la cara pública del warehouse), interacturando constantemente. Mientras los requerimientos del negocio nos dicen **qué** necesitamos hacer, la arquitectura técnica nos responde el interrogante de **cómo** lo haremos.

Kimball [Kim98] desarrolla toda esta arquitectura en varios capítulos (Back Room Technical Architecture –Capítulo 9-, Architecture for the Front Room –Capítulo 10-, Infraestructure and Metadata –Capítulo 11-) por lo cual referenciamos a los mismos si se desea interiorizar sobre estos temas.

2.7 Selección de Productos e Instalación

Utilizando el diseño de arquitectura técnica como marco, es necesario evaluar y seleccionar componentes específicos de la arquitectura como ser la plataforma de hardware, el motor de base de datos, la herramienta de ETL o el desarrollo pertinente, herramientas de acceso, etc.

Una vez evaluados y seleccionados los componentes determinados se procede con la instalación y prueba de los mismos en un ambiente integrado de data warehousing.

Caracteristica	Peso	Producto A	Producto B	Producto C	...
Capacidades Basicas de ETL					
xxxxxxxxxxxxxxxxxxxxxxxx	85				
xxxxxxxxxxxxxxxxxxxxxxxx	70				
Job Control & Scheduling					
xxxxxxxxxxxxxxxxxxxxxxxx	90				
xxxxxxxxxxxxxxxxxxxxxxxx	85				
Metada & Estandares					
xxxxxxxxxxxxxxxxxxxxxxxx					
xxxxxxxxxxxxxxxxxxxxxxxx					
Info de Vendedor	70				
xxxxxxxxxxxxxxxxxxxxxxxx	85				
xxxxxxxxxxxxxxxxxxxxxxxx					
Puntaje Total					
Ranking					

Fig. 2-8 Ejemplo de matriz de evaluación de productos

2.8 Especificación de Aplicaciones para Usuarios Finales

No todos los usuarios del warehouse necesitan el mismo nivel de análisis. Es por ello que en esta etapa se identifican los diferentes roles o perfiles de usuarios para determinar los diferentes tipos de aplicaciones necesarias en base al alcance de los diferentes perfiles (gerencial, analista del negocio, vendedor, etc.)

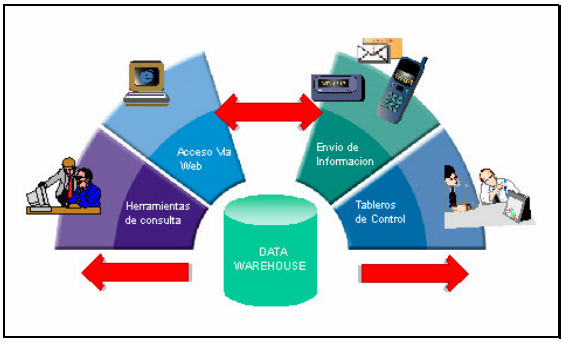

Fig. 2-9 Variedad de interfases por perfil

Los diferentes roles o perfiles de usuarios determinan la interfase o ventana al warehouse. Herramientas de diseño de reportes y consultas avanzadas para analistas, tableros de control para gerentes, acceso mediante inter/intra net para usuarios internos/externos remotos, envío de información por dispositivos no estándares para usuarios internos/externos, etc.

Kimball se concentra sobre el proceso de creación de aplicaciones *"templates"*. Comienza definiendo el concepto de la aplicación para usuario final y su rol en el acceso a la información del negocio. Brinda un marco metodológico bastante estándard en lo que ha desarrollo de aplicaciones (como piezas de software) se refiere. Divide el proceso de creación de las aplicaciones para usuarios finales en dos grandes fases: **especificación y desarrollo**. Clasifica a los usuarios según su perfil de consulta, desde usuarios con un perfil más estratégico y menos predecibles (**power users**) hasta usuarios netamente operacionales que consumen una serie de reportes estándares (**final users**) pasando por los usuarios gerenciales con uso de interfases *push-button* (**EIS users**).

Kimball destaca, como uno de los requisitos a cumplir por las aplicaciones, la posibilidad de hacer análisis **AD HOC**. Análisis ad hoc es simplemente la **habilidad para los usuarios de cambiar los parámetros sobre un reporte** para crear sus propias versiones personalizadas de ese reporte. De está forma se dará respuesta a necesidades como "Quiero ver este reporte, pero por mes en lugar de trimestre", "Puedo ver este reporte a nivel provincia en lugar de región?", "Puedo ver esta evolución de ventas mensual pero sólo de mi equipo de ventas?". Más importante aún es que las respuestas a estos interrogantes lo resuelven los **propios usuarios** sin la necesidad de la intervención del departamento sistemas y **máximizando el tiempo de análisis** por sobre el tiempo de construcción e integración de la información.

Kimball define entonces que las aplicaciones *"templates"* para usuarios finales proveen el marco (layout) y la estructura de un reporte para ser especializado por un conjunto de parámetros. El usuario selecciona los parámetros de una pick list o aceptando los valores por defecto cuando ejecuta el template. Este enfoque orientado a parámetros permite a los usuarios generar docenas o potencialmente cientos de reportes de estructura similar desde un mismo template [Kim98]. Todo esto de forma amigable y con interfases gráficas que hacen uso de todo el trabajo realizado en la contrucción del warehouse en etapas previas del BDL.

Una advertencia que hace Kimball es el tiempo que existe entre el **relevamiento** y **especificación** de las aplicaciones para usuarios finales y el momento del **desarrollo** e **implementación** de la misma. Como pudimos ver, durante las primeras semanas del proyecto se realiza el relevamiento de los requerimientos de los usuarios y recién una vez que existen datos en el warehouse (aunque más no sea datos de prueba) puede comenzarse con la construcción de la aplicacion final (al menos es lo recomendable), es decir luego del diseño lógico y físico. Kimball insta entonces a manejar con mucho cuidado el tema de la **documentación explícita** (tratar que lo menos posible quede en la propiedad intelectual de los que hicieron el relevamiento). Sino se deberá realizar un esfuerzo significativo en el redescubrimiento de los puntos especificados en etapas tempranas del proyecto.

Kimball destaca cuatro pasos principales (siempre enfatizando el hecho de involucrar a los usuarios en cada uno de estos pasos) [Kim98]:

- Determinación del conjunto de templates iniciales (identificar reportes candidatos, clasificarlos y priorizarlos)
- Diseño de la estratégia de navegación dentro de la aplicación (esquema de pantallas, esquema de carpetas –directorios-, criterios de agrupamiento -por datos, por dueño, por regla del negocio, etc.-)
- Determinación de estándares (nombre de objetos, ubicación de objetos, formato de las salidas)
- Detalle de las especificaciones (definición: nombre, descripción o propósito, frecuencia, parámetros, restricciones, layout, etc.)

2.9 Desarrollo de Aplicaciones para Usuarios Finales

Siguiendo a la especificación de las aplicaciones para usuarios finales, el desarrollo de las aplicaciones de los usuarios finales involucra configuraciones del metadata y construcción de reportes específicos.

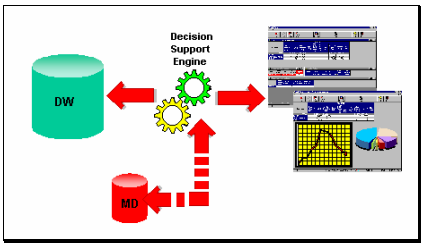

Fig. 2-10 Configuración de Metadata y construcción de reportes

Una vez que se ha cumplido con todos los pasos de la especificación y se tiene la posiblidad de trabajar con algunos datos de prueba, comienza el **desarrollo de la aplicación** [Kim98]:

- Selección de un enfoque de implementación
  - Basado en Web
    - ✓ Inter/Intra net
    - ✓ Usuarios altamente distribuidos
    - ✓ Manejo centralizado de nuevas versiones
  - Herramienta propietaria
    - ✓ Mayor complejidad de uso
    - ✓ Para usuarios más capacitados
    - ✓ Instalación local
  - EIS
    - ✓ Acceso estructurado
    - ✓ Secuencialidad de pantallas
    - ✓ Push-Button
  - Interfase personalizada
    - ✓ API (Application Programming Interface)
    - ✓ Desarrollos propios sobre la base de un conjunto de funcionalidades
- Desarrollo de la aplicación
  - Definición de herramienta de acceso al MetaData
  - Desarrollo de templates y esquema de navegación de la aplicación
  - Selección de reportes para pre-ejecución
- Prueba y verificación de datos
  - Descripciones
  - Información duplicada
  - Relaciones entre atributos
  - Consistencia e integridad de datos con sistemas fuentes

- Documentación y Roll Out
  - Retroalimentación con los resultados de la puesta en producción
- Mantenimiento
  - Nuevos templates
  - Incorporación de nuevos sistemas fuentes
  - Monitoreo de performance
  - Eliminación de templates en desuso

2.10 Implementación

La implementación representa la convergencia de la tecnología, los datos y las aplicaciones de usuarios finales accesible desde el escritorio del usuario del negocio. Hay varios factores extras que aseguran el correcto funcionamiento de todas estas piezas, entre ellos se encuentran la capacitación, el soporte técnico, la comunicación, las estrategias de feedback. Todas estas tareas deben ser tenidas en cuenta antes de que cualquier usuario pueda tener acceso al data warehouse.

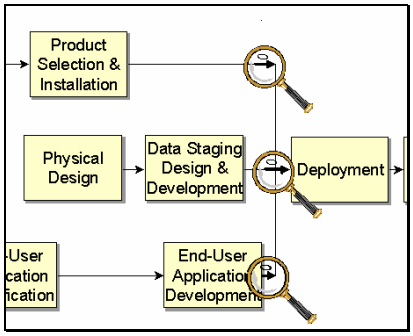

Fig. 2-11 Implementación como convergencia de la tecnología, los datos y las aplicaciones

En este capítulo, mucho menos técnico y más orientado al gerenciamiento del proyecto, Kimball [Kim98] presenta una serie de tareas que deben cumplirse para garantizar un fin de proyecto y un producto de calidad.

La tecnología que reside en el escritorio del usuario es la última pieza que debe ser ubicada antes de la salida a producción (roll out o deployment). Desafortunadamente, afirma Kimball, las organizaciones frecuentemente subestiman el esfuerzo y el tiempo requerido para esta etapa. Kimball propone entonces una checklist sobre actividades que deberían ocurrir antes de la implantación para asegurar que la infraestructura correspondiente al ambiente del usuario este correcta. Esta checklist incluye: configuración de hardware, conexión a las bases, acceso a intranet o internet, direcciones LAN (si no son dinámicamente asignadas), auditorías de tecnología sobre las configuraciones en las que se encontraban las PCs, preveer actualizaciones de hardware y software (determinando responsables, proyecto o área de usuario), verificaciones de seguridad (logon de red y base de datos), prueba de procedimientos de instalación en una variedad de máquinas, planificación de instalación con la correspodiente educación a los usuarios, etc.

Ralph Kimball dedica gran parte de este capítulo a la **educación de los usuarios finales** afirmando que *"una robusta estrategia de educación para usuarios de negocios es un prerequisisto para el éxito del proyecto"*. Entre sus recomendaciones para cumplir exitosamente con esta tarea se encuentran dos aspectos, por una lado el **alcance de la educación**, y por otro, la **metodología de presentación**. Para lo primero, debe instruirse al usuario en tres aspectos claves: **contenido del warehouse, aplicación y herramientas de acceso**. En cuanto a lo segundo, para los usuarios finales no es fácil entender los límites entre las aplicaciones, los contenidos y las herramientas del warehouse. Para los usuarios el data warehouse es un todo, no la suma de componentes discretos, por lo tanto la educación también debe reflejar la misma perspectiva. **No** debe interpretarse educación de los usuarios finales como educación **sólo sobre las herramientas de acceso**. La educación sobre herramientas es inútil al menos que se complemente con educación sobre el contenido del warehouse (cuáles datos están disponibles, qué significan, cómo se usan y para qué usarlos). Otro factor clave es la correcta entrega de **niveles de educación** según el perfil del usuario (usuario final, power user, etc.)

El armado de un **ambiente de training** (como subconjunto de datos de producción) es siempre recomendable, dado que los tiempos de respuesta en producción, si bien pueden ser apropiados para el análisis de la información, tal vez no sean lo mejores para un curso de un día de duración. Además, el contenido del warehouse puede variar entre curso y curso haciendo más complicada la administración y mantenimiento del material educativo.

Kimball establece dos premisas para regular las relaciones entre la educación y el warehouse. Realizar educación de usuarios finales **sólo si el warehouse está listo** (en tiempo y forma) y establecer la política de *"Sin Educación, entonces Sin Acceso"*. Cumpliendo estas dos premisas se puede garantizar el éxito del proyecto en esta etapa final del ciclo de vida.

El **soporte** es la otra columna que ayudará a implementar con éxito el proyecto. Se deberá definir claramente los **niveles de soporte** (a nivel aplicación, modelo, calidad de datos) de forma transparente para el usuario. La **documentación** juega un papel importantísimo en la ayuda a usuarios finales, dado que el datawarehouse evolucionará rápidamente, la documentación escrita puede quedar obsoleta también de forma prematura. La documentación en línea empieza entonces a tener vida propia, ya sea dentro del warehouse en sí mismo o como herramientas anexas, como el data warehouse web site propuesto por Kimball.

Finalmente, la propuesta de Kimball se orienta a la **metodología de implementación**. Para ello propone un esquema de versionado (**release framework**). Primero se pasa por la versión **Alpha**, primera oportunidad para el **grupo de trabajo** de conducir una prueba del sistema de principio a fin. Todos los componentes del sistema deben ser testeados (infraestructura técnica, extracción, transformación, carga, procedimientos de calidad, performance, templates, etc.). Típicamente es de naturaleza iterativa y sólo puede decirse que se termina esta versión cuando el grupo de trabajo confía plenamente en la calidad del warehouse como un todo (por lo tanto a la hora de estimar los tiempos debe tenerse en cuenta que este será el objetivo final de la versión Alpha por lo cual debe asignarse un tiempo prudencial). Durante la etapa Alpha un conjunto limitado de usuarios finales son provistos con acceso al warehouse (sólo aquellos que pertenecen al grupo de trabajo). Luego viene la versión **Beta**. El objetivo de esta versión es conducir una prueba a **nivel usuario** de principio a fin. El grupo Beta está formado por los power users, la determinación de la cantidad de este grupo es también importante, pues si son pocos la aplicación no se probará adecuadamente, y si son muchos se estará salteado la etapa de testing cayendo directamente en una salida a producción.

Una vez superadas estas dos versiones, al igual que en un proceso de diseño e implementación de software, llegamos a un estado **GA** (general availability). Lo que sugiere Kimball es que, todo cambio y/o modificación que se realice posteriormente al warehouse, pase internamente por un estado **Alpha y Beta** aunque externamente sea una **nueva versión, release o pack** desde el último GA.

2.11 Mantenimiento y crecimiento

Como se remarca siempre, Data Warehousing es un proceso (de etapas bien definidas, con comienzo y fin, pero de naturaleza espiral) pues acompaña a la evolución de la organización durante toda su historia. Se necesita continuar con los relevamientos de forma constante para poder seguir la evolución de las metas por conseguir. Según afirma Kimball [Kim98], "*si se ha utilizado el BDL el data warehouse esta preparado para evolucionar y crecer*". Al contrario de los sistemas tradicionales, los cambios en el desarrollo deben ser vistos como signos de éxito y no de falla. Es importante establecer las prioridades para poder manejar los nuevos requerimientos de los usuarios y de esa forma poder evolucionar y crecer.

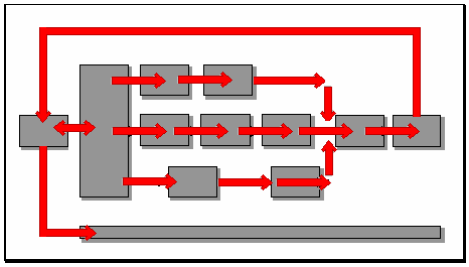

Fig. 2-12 Data Warehousing como proceso de naturaleza espiral

El sentido de las flechas fija claramente el concepto evolutivo y espiral del BDL.

Una vez que se ha construido e implantado el data warehouse no hay tiempo para el descanso, rápidamente debemos estar preparados para administrar el mantenimiento y crecimiento del mismo. Si bien las tareas pueden llegar a parecer similares a las tratadas en otras etapas del BDL, existe una diferencia clave: **los usuarios están ahora accediendo al warehouse**.

Kimball [Kim98] brinda entonces una serie de puntos a tener en cuenta para mantener existosamente el warehouse. Entre ellos se destacan: el continuo soporte y la constante capacitación a usuarios de negocios, el manejo de la infraestructura (monitoreo de base de datos, tráfico, etc.), tuning de rendimiento sobre las consultas, mantenimiento del metadata y procesos ETLs. Otros aspectos involucran el monitoreo regular del cumplimiento de las expectativas sobre el warehouse (variables de medición del éxito del proyecto que se habían fijado en la etapa de relevamiento), relevamiento de casos de estudio (situaciones reales donde una decisión basada en información del warehouse tuvo impacto sobre el negocio –ROI-), constante publicidad interna del uso del warehouse (permitiendo acceso siempre y cuando se tenga la capacitación correspondiente) y constante comunicación con los sectores de negocios y sistemas para asegurar la buena salud del datawarehouse.

En cuanto a cómo prepararse para el crecimiento del warehouse, Kimball recomienda la creación de un comiteé conformado por analistas del negocio y sistemas que establezca **prioridades y procedimientos**. El manejo de prioridades lo discrimina según sean **mejoras menores o mayores** (clasificadas según el impacto en el warehouse y excluyendo los errores, los cuales deben ser resueltos inmediatamente). Como mejoras menores incluye incorporación de datos sencillos, nuevas agregaciones, cambios en niveles superiores del modelo (atributos de alto nivel, lejanos a las bases tables), tareas que en definitiva no lleven mas de días o semanas. Advierte sobre el razonamiento simplista de decidir no aplicar una mejora pues el cambio es muy menor, *"1000 pequeñas mejoras es una ventaja estratégica"*. Dentro de las mejoras de mayor impacto se encuentran las que modifican y/o crean bases tables o atributos asociados a las mismas, tareas que pueden disparar nuevos proyectos de meses de duración.

Finalmente, Kimball concluye su libro garantizando que el seguimiento y cumplimiento de todas las etapas BDL ayudará a obtener un warehouse **mantenible y de controlado crecimiento**. Como es ilustrado en el diagrama del BDL, se deberá volver a iterar sobre el mismo pasando nuevamente por cada una de las etapas para de esta forma administrar correctamente los cambios a warehouses existentes, como así también para respetar y aprovechar definiciones realizadas para otros proyectos.

2.12 Gerenciamiento del Proyecto

El gerenciamiento del proyecto asegura que las actividades del BDL se lleven en forma y sincronizadas. Como lo indica el diagrama, el generenciamiento acompaña todo el ciclo de vida. Entre sus actividades principales se encuentra el monitoreo del estado del proyecto y la comunicación entre los requerimientos del negocio y las restricciones de información para poder manejar correctamente las expectativas en ambos sentidos.

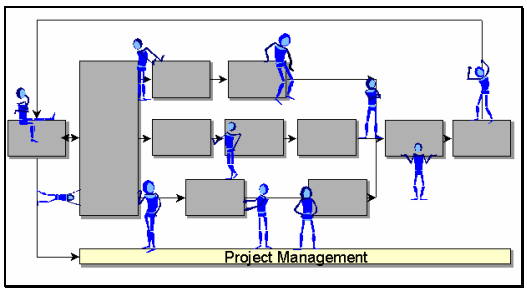

Fig. 2-13 Gerenciamiento del proyecto

El gerenciamiento del proyecto debe asegurar que las actividades del BDL se lleven en forma y sincronizadas.

2.13 Puntos problemáticos de un DW

Según identifica Jeff Bedell [Bed97], los principales puntos de atención que pueden llegar a complicar un proyecto de data warehousing se discriman en según las siguientes tres áreas:

- **Rutinas de Carga**. Incluye programas de extracción y limpieza de datos. Surgen problemas en este punto dada la falta de integración y estructura consistente (alineada) entre los sistemas fuentes.
- **Mantenimiento**. Dados los diferentes períodos de almacenamiento para OLTP y OLAP y el hecho de que los DW son sistemas secundarios de información, otro problema surge para sincronizar los datos entre los sistemas operacionales fuentes y los warehouses.
- **Tuning**. Dado los patrones de uso y los métodos de acceso típicos de los sistemas OLAP, diseñadores y administradores deben realizar cambios significativos a los implementados en el tuning de sistemas OLTPs.

Por nuestra experiencia en este campo, podemos decir que, además de estos puntos problemáticos propios de un ambiente OLAP, un proyecto de data warehousing no esta ajeno a otros inconvenientes que son comunes a todos los proyectos tales como, falta de compromiso, relevamientos contradictorios, requerimientos incompletos, etc.

Capítulo 3 - El Modelo Dimensional del Negocio (BDM)

El modelo dimensional utilizado en el modelado de data warehousing, organiza y presenta los datos definiendo **dimensiones** (líneas o áreas temáticas del negocio). Por ejemplo, en los modelos de retail es muy común encontrar las mediciones estructuradas por las dimensiones sucursal, producto y tiempo.

De esta forma, permiten analizar la información a distintos niveles de agregación dentro de las diferentes dimensiones. Dentro de cada dimensión se puede definir los niveles de agregación o sumarización para cada análisis, a estos niveles de granularidad se los caracteriza con el nombre de **atributos**.

Dentro de este contexto, un análisis consiste en definir el nivel de agregación que califique el conjunto de datos resultado, independientemente para cada dimensión.

En este capítulo definiremos los conceptos centrales necesarios para la construcción del modelo lógico y lo compararemos brevemente con la otra técnica de modelado ampliamente popular, los diagramas entidad relación.

3.1 Componentes básicos

A la hora de realizar un modelo dimensional del negocio (BDM) se hace necesaria la presentación de los siguientes componentes que definiremos a lo largo de este capítulo:

- **Dimensiones** (Dimensions)
- **Atributos** (Attributes)
  - Elementos (Attributes Elements)
- **Relaciones** (Attribute Relationships)
  - Jerarquías (Hierarchies)
- **Variables** o **Indicadores** (Facts o Metrics)

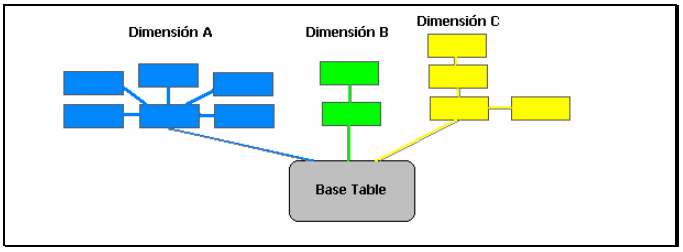

Fig. 3-1 Esquema de un modelo dimensional

3.1.1 Dimensiones

Son las **áreas temáticas, líneas del negocio** o **sujetos del negocio**. Las mismas proveen un método general para organizar la información corporativa. Definidas como un grupo de uno o más atributos, separados y distintos uno de otros (es decir, que no se comparten atributos), según el esquema de implementación elegido, puede ser que no se encuentren explícitamente en el data warehouse, sino que se presenten sólo como un recurso **conceptual** o **lógico** que ayuda a mostrar múltiples perspectivas de los datos permitiendo realizar análisis por diferentes dimensiones o incluso cruzando información entre distintas dimensiones (Ver más adelante Star Schema y SnowFlaked Schema).

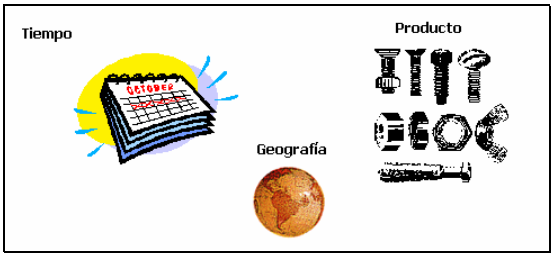

Fig. 3-2 Dimensiones

3.1.2 Atributos

Los atributos son una **agrupación de elementos** o items dentro de una dimensión. Representan **categorías** o clases de elementos que tienen el mismo nivel lógico dentro de una dimensión donde todos los elementos de un atributo se relacionan con otros atributos de la dimensión de la misma forma. La finalidad de los atributos es ver la información de cada dimensión a **diferentes niveles de detalle** y **agrupar** los datos para ser analizados.

Permiten definir niveles de agregación y presentar datos agrupados por uno o más atributos. Los atributos son niveles dentro de una dimensión o calificadores de una dimensión. En un modelo ER, muchos de los atributos serían clasificados como entidades y no como atributos. Por ejemplo, item de un producto; los items (como entidad del DER) pueden tener algunos atributos característicos como ser el color y el tamaño. En un modelo dimensional, item, color y tamaño son todos tratados como atributos de la dimensión producto.

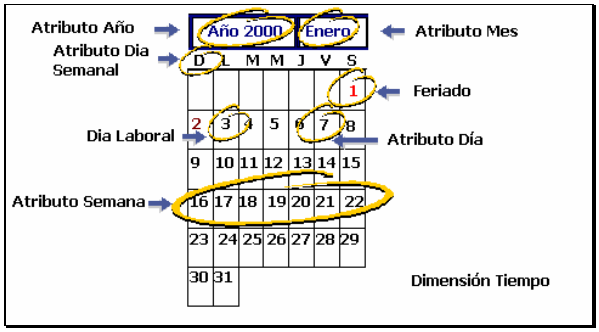

Fig. 3-3 Atributos

3.1.3 Elementos

Son las **instancias** o valores de los atributos que, como componentes atómicos del modelo, permiten clasificar el rendimiento del negocio. Es importante aclarar que si bien no forman parte del BDM es aconsejable su incorporación para un mayor entendimiento del modelo en etapas tempranas del ciclo de vida (relevamiento)

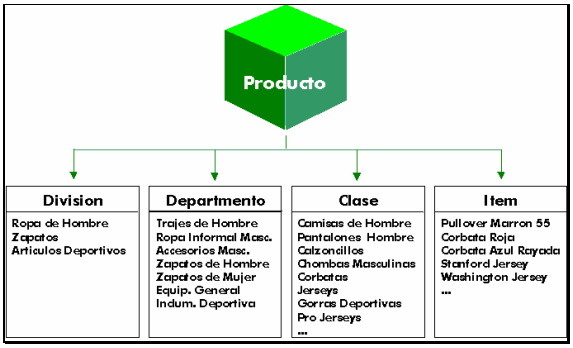

Fig. 3-4 Elementos o instancias de atributos

3.1.4 Relaciones

Son **asociaciones lógicas de atributos** dentro de una jerarquía definida por las instancias de los atributos y transitivas dentro de una jerarquía. Al igual que en DER existen diferentes tipos de relaciones (clasificadas por su cardinalidad):

- Uno-a-uno (1:1)
- Uno-a-muchos (1:M)
- Muchos-a-uno (M:1)
- Muchos-a-muchos (M:N)

Es importante aclarar que los atributos **dentro** de una dimensión están directamente relacionados uno con otros a través de los diferentes tipos de **relaciones** antes definidos. En cambio, los atributos en **diferentes** dimensiones están relacionados uno con otros a través de los **indicadores** o variables del negocio definidas como intersección de las dimensiones.

3.1.5 Jerarquías

Representadas por un **ordenamiento lógico** dentro de la dimensión, se encuentran formadas por los diferentes tipos de relaciones entre los atributos de una misma dimensión. Pueden existir múltiples jerarquías dentro de una dimensión pero siempre es posible identificar una **jerarquía principal** o columna vertebral de la dimensión y **jerarquías secundarias** o descriptivas compuestas por atributos característicos definidos desde la jerarquía principal.

Dentro del contexto de navegación del modelo dimensional, se puede decir que las diferentes jerarquías definen el **mapa** de caminos para el *"drilling"* o la *"navegación"* de los datos. Haremos **drill-up** o **roll-up** cuando nos movamos hacia un atributo superior dentro de la jerarquía (navegación ascendente), **drill-down** o **roll-down** cuando analicemos información a mayor nivel de detalle (navegación descendente), **drill-within** cuando nos movamos entre la jerarquía principal y la característica desde un atributo hacia cualquier otro que no sea ni descendiente ni ancestro dentro de la misma dimensión (navegación intradimensional) y **drill-across** para analizar información sobre diferentes dimensiones (navegación interdimensional).

Como convenciones del modelado, la jerarquía principal se dibuja verticalmente desde el atributo más agregado (arriba) hasta el más atómico (abajo) y las jerarquías características se adicionan por los costados.

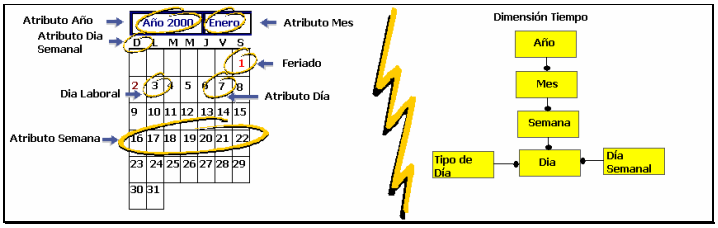

Fig. 3-5 Relaciones y Jerarquías

3.1.6 Indicadores

Son las **variables** o **métricas** que ayudarán a medir la **performance del negocio**. Existen dos tipos de indicadores: **básicos** y **derivados**.

Los indicadores básicos, primitivos o crudos existen físicamente en el warehouse junto a los atributos que los caracterizan, pueden venir de diferentes sistemas fuentes y tener distintos niveles de granularidad o agregación. Por ejemplo, la variable Venta($) es llevada diariamente mientras que el indicador Unidades en Stock es seguido semanalmente.

Por otro lado, los indicadores derivados o métricas calculadas se construyen a partir de los indicadores base y pueden o no estar almacenados físicamente en el data warehouse (es típicamente una decisión de tuning). Un ejemplo clásico de métrica derivada es Margen de Ganancia, la cual se define como la resta entre Precio y Costo (ambos indicadores básicos).

3.2 Comparación con modelo ER

Al momento de presentar el modelado dimensional, parece inevitable hacer la comparación con los modelos entidad-relacion (ERM) dada su gran penetración y aceptación en los ambientes de base de datos relacionales y su incorporación (en la etapa de análisis de requerimientos) dentro del ciclo de vida del software.

Si bien los modelos dimensionales pueden parecer similares a los clásicos DER (Diagrama Entidad-Relación), hay algunas diferencias que enumeraremos a continuación.

- El término **entidad** no es usado en el modelado dimensional.
- Algunas entidades en un DER serán mapeadas a atributos en un modelo dimensional.
- Una **dimensión** es una agrupación lógica de atributos que relaciona a cada uno de ellos.
- Indicadores son los valores usados para medir el rendimiento del negocio, por ejemplo, costo, precio, margen, etc.
- Los indicadores se definen como intersección de las diferentes dimensiones.

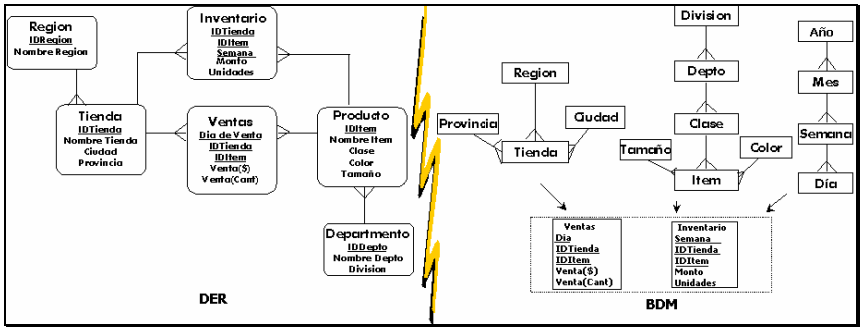

Fig. 3-6 DER vs. BDM

Según sentencia Kimball [Kim95], no tiene sentido utilizar el DER para realizar el relevamiento de requerimientos y luego aplicar una serie de reglas para transformalo en un modelo dimensional. El DER puede ser útil para contrastar los requerimientos de los usuarios con la estructura de los sistemas operacionales existentes pero no para ser usado de base en la definición del BDM.

Capítulo 4 - Construcción básica del esquema físico

Como sabemos, un esquema físico relacional está organizado, básicamente, con los siguientes componentes:

- **Tablas**
  - Nombre de tabla
  - Nombres de columnas
  - Tipos de datos de columnas
- Definición de **clave primaria**
- Referencias de **claves foráneas**.

Debemos enfrentarnos entonces al desafío de traducir nuestro modelo dimensional del negocio al *"lenguaje"* que entienden las bases de datos. Este mapeo puede realizarse siguiendo unos pasos básicos que finalizan en un esquema normalizado (el cual se detalla en los siguientes puntos).

Luego de obtener el esquema pueden realizarse diversas técnicas de desnormalización que apunten a mejorar los tiempos de respuestas de las consultas realizadas sobre el esquema definido. También puede identificarse cuán dinámicas o estáticas son las relaciones entre los atributos con el objetivo de planificar el seguimiento de la historia entre estas relaciones (**Versioning**).

En este capítulo describiremos entonces las diferentes técnicas y métodos para mapear los conceptos lógicos del modelo dimensional en las correspondientes estructuras del esquema físico. Además presentaremos diferentes esquemas de implementación analizando ventajas y desventajas de cada uno de ellos.

4.1 Mapeo desde Modelo Lógico hacia Esquema Físico

Como proceso de mapeo del modelo dimensional a un esquema físico se puede decir que la construcción del esquema involucra la definición con componentes propios (tablas, filas y columnas), de todos los componentes del modelo dimensional (atributos, relaciones e indicadores).

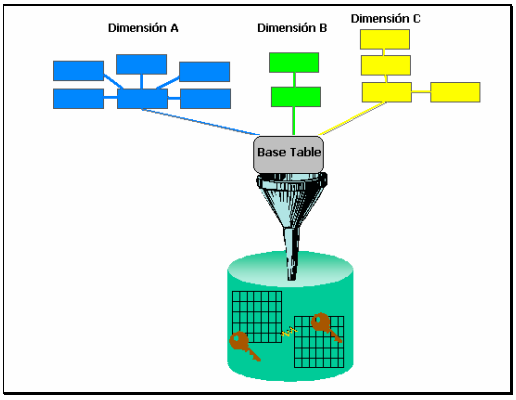

Fig. 4-1 Mapeo del BDM a Esquema Físico Relacional

Paralelamente o después de la definición de las estructuras iniciales, se debe examinar las estrategias de optimización y tuning del warehouse, como así también definir la frecuencia de refresco y los diferentes perfiles de uso.

A la hora de definir nuestro esquema físico del modelo dimensional se identifican tres tipos de tablas:

- **Tablas lookup**
  Contiene una columna identificatoria (id) y, si existe, una columna de descripción del atributo que representa
- **Tablas relacionales**
  Contiene el id de dos o más atributos definiendo, de esta forma, la asociación entre ellos.
- **Tablas base**
  Contiene columnas que representan los indicadores o variables del negocio y ids de los atributos que indican el nivel de granularidad de las métricas.

Podemos ver entonces que surgen tres tipos de columnas o campos:

- **Identificación**
  - *Contiene códigos de identificación*
  - *Requerido para todos los atributos*
  - *Preferiblemente numéricos (procesamiento más rápido)*
- **Descripción**
  - *Contiene descripciones, generalmente, en formato texto.*
  - *Opcional para todos los atributos*
- **Variables** (Fact Columns)
  - *Numéricas*

4.2 Construcción de tablas Lookup

Se comienza asociando una lookup a cada atributo definido en el BDM de tal forma de ir construyendo un esquema normalizado. Se identifica la clave primaria y la descripción de cada atributo para representarla como los correspondientes campos de la tabla lookup. Dado que el campo descripción es opcional puede dejarse sólo el identificador y utilizarlo con ambos propósitos.

Existen algunos atributos característicos, como por ejemplo el Domicilio de un cliente, que no tienen su propia lookup sino que residen en la lookup de algún otro atributo (por ejemplo, en este caso estaría incluido en la lookup Cliente).

Si las tablas lookups representa una dimensión completa y no un atributo en particular, entonces se las conoce también como **tablas dimensionales**, las mismas son utilizadas ampliamente en el esquema físico conocido con el nombre de Esquema Estrella (**Star Schema**) desarrollado más adelante.

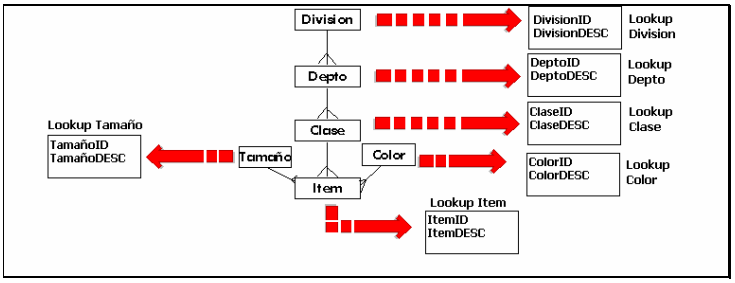

Fig. 4-2 Tablas Lookup

4.3 Relaciones de atributos dentro de una dimensión

Se identifica las relaciones directas entre los distintos atributos (relaciones **Padre-Hijo**), luego se define una tabla para cada relación que contiene los campos ids de cada lookup que representa los atributos asociados. Como se realiza comúnmente, la definición de la clave primaria estará asociada a la cardinalidad de la relación relevada en el BDM.

En el caso de las relaciones que no sean de muchos-a-muchos se puede evitar la creación de una tabla relacional, incluyendo el **id** del padre en la lookup del hijo de la relación y definiéndola como clave foránea referenciándola a la lookup padre. De esta forma también se pueden incluir los ids de todos sus ancestros y mejorar los tiempos de respuesta al reducir la cantidad de joins en las consultas, incrementando, por otro lado el mantenimiento de las tablas.

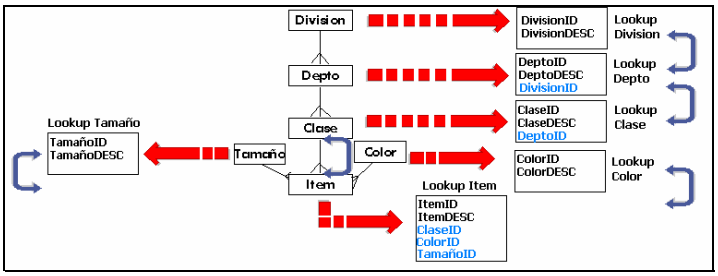

Fig. 4-3 Implementación de relaciones

4.4 Definición de tablas Base

Se debe definir una tabla para cada conjunto de indicadores, agrupados según su **granularidad**. Como se explicó anteriormente, cada tabla de hechos (**fact table** o **base table**) tiene campos numéricos asociados a los indicadores y campos ids para cada atributo que interviene en la *"dimensionalidad"* y *"granularidad"* de la variable a medir. La dimensionalidad de una variable está asociada a los conceptos generales (**dimensiones**) que participan en la definición de un indicador y la granularidad al nivel de detalle o agregación de la misma y define los **atributos** de cada dimensión que caracterizan el valor de la variable. Por ejemplo, si las ventas se cargan diariamente para cada sucursal y cada familia de productos, la dimensionalidad del indicador *Unidades Vendidas* será **Tiempo-Geografía-Productos** y la granularidad será **Dia-Sucursal-Familia**, en cambio, si el stock de mercadería se sigue semanalmente, la dimensionalidad de *Unidades en Stock* será la misma pero la granularidad será **Semana-Sucursal-Familia**.

La clave primaria de la tabla base estará compuesta por todos los ids que definan la dimensionalidad de los indicadores y cada id individualmente referenciará, como clave foránea, a la tabla lookup que represente el atributo asociado.

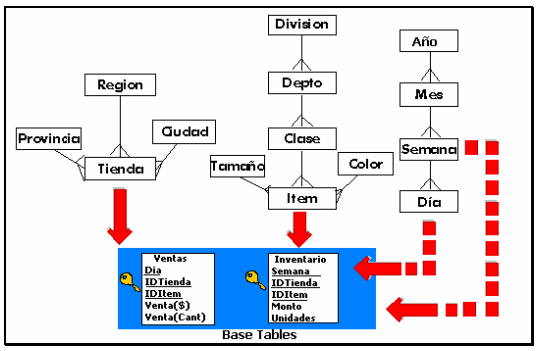

Fig. 4-4 Base Table

4.5 Tipos de esquemas

Tomando como base el trabajo de Jeff Bedell [Bed97], en este capítulo analizaremos algunos de los posibles esquemas físicos que se pueden implementar junto a las ventajas y desventajas asociados a cada uno de ellos .

4.5.1 Esquema Estrella Consolidado I

Las tablas **lookup** representan a una **dimensión completa**. Contienen una **clave genérica**, una **descripción genérica**, un **campo de nivel** que indica la *"altura"* del atributo dentro de la jerarquía y campos **ids** para cada atributo dentro de la dimensión. Por ejemplo, si tenemos tres dimensiones Producto, Geografía y Tiempo, existen tres lookups, cada una con un id genérico del tipo Producto_Key, Geo_Key, Tiempo_Key junto a Producto_DESC, Geo_DESC y Tiempo_DESC mas el campo 'nivel' de tipo numérico y los ids: FamiliaID, LineaID, ItemID en el caso de la dimensión Producto.

Este esquema soporta **solamente una tabla base** y en la misma caen sólo las claves genéricas de las dimensiones junto a los indicadores del negocio. Al no existir tablas relacionales, pues los atributos y sus relaciones de una dimensión se encuentran todos representados en una única lookup, esta estructura no admite relaciones M:N de forma natural.

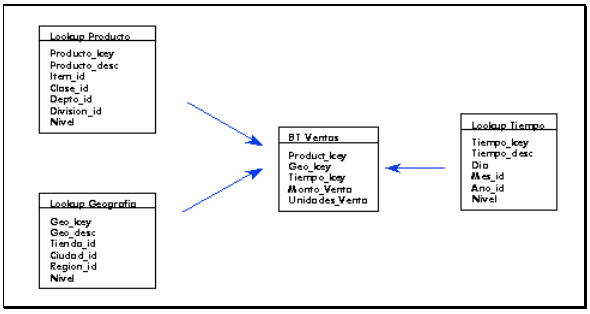

Fig. 4-5 Esquema Estrella Consolidado I

4.5.2 Esquema Estrella Consolidado II

Las tablas lookups tienen la misma estructura básica del Esquema Consolidado I, salvo que el id de cada atributo es reemplazado por la descripción y la descripción genérica es eliminada. La estructura de la tabla base es idéntica al esquema anterior.

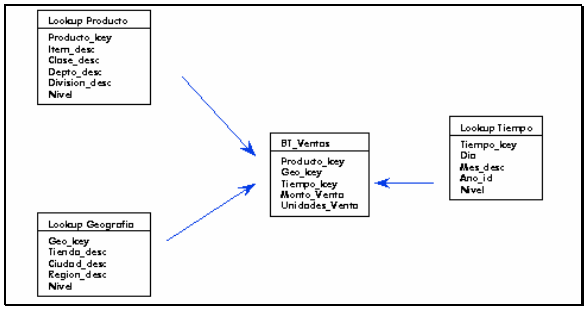

Fig. 4-6 Esquema Estrella Consolidado II

4.5.3 Esquema Estrella Consolidado III

Las tablas lookups son una combinación de los dos esquemas anteriores, contienen la clave genérica de la dimensión, el campo de nivel, los ids y las descripciones de cada atributo de la jerarquía de esa dimensión. La estructura de la tabla de hechos permanece intacta.

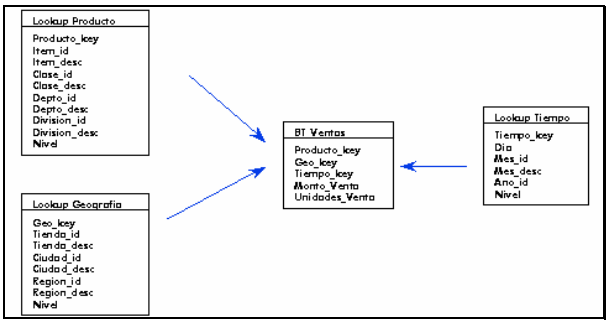

Fig. 4-7 Esquema Estrella Consolidado III

4.5.4 Comparación entre Esquemas Estrella Consolidados

Estos esquemas tienen abundancia de campos en **NULL**, formando una matriz triangular superior en la parte correspondiente a los indicadores de la tabla base. Haciendo un análisis más particular podemos ver que una debilidad que presenta **EEC I** se manifiesta a la hora de requerir un análisis que involucre más de un atributo de una misma dimensión. Por ejemplo, ver las ventas discriminadas por Región y Provincia (ambos atributos de la dimensión Geografía). Dado que existe sólo un campo descripción en la lookup Geografía, para producir el efecto deseado se debe realizar un **self join** para cada atributo de la misma dimensión que se desee mostrar, pudiendo ser esta operación muy costosa.

El **EEC II** resuelve este último problema pero al tener sólo las descripciones (generalmente texto variable) la búsqueda de los atributos correspondientes para una consulta se hace demasiado lenta en contraposición a la búsqueda por campos numéricos como son generalmente los ids.

Por último el **EEC III** combina las características de los EEC I y II pagando el costo de ser un esquema completamente desnormalizado.

En cuanto a la tabla base, la misma es del tipo *"in-table aggregation"* pues todas las consolidaciones (niveles de agregación) se realizan dentro de la misma tabla de hechos, de allí el nombre de **Consolidado** que reciben estos esquemas.

Entre las ventajas de estos esquemas se encuentra la **poca cantidad de tablas** y la **facilidad de generar SQL automáticamente** (para realizar los distintos análisis) ya que la estructura es bastante estática.

Entre las desventajas podemos enumerar la dificultad de **relaciones M:N**, **tabla base** extremadamente **grande** por manejar las agregaciones dentro de sí misma. En general puede decirse que **no es un modelo escalable** en cuanto a **cantidad de atributos** dentro de cada dimensión y en **cantidad de elementos** dentro de cada atributo.

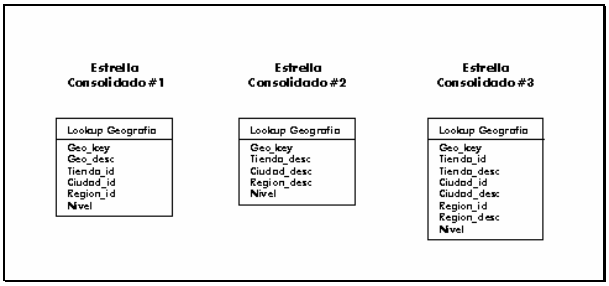

Fig. 4-8 Comparación de Esquemas Estrellas Consolidados

4.5.5 Esquema Estrella Normalizado I

Las tablas **lookups** representan a cada **atributo**. Contienen el id, la descripción y el id del padre inmediato en la jerarquía de la dimensión a la que pertenecen. En caso de relaciones M:N se define la correspondiente tabla relacional que involucra los ids de los atributos asociados.

La **tabla base** tiene un **único nivel** de datos (no tiene agregaciones incluidas). La clave primaria esta formada por los ids de los atributos hojas de las dimensiones.

Este esquema es comúnmente conocido con el nombre de esquema Copo de Nieve (**SnowFlaked Schema**). Como puede verse, este esquema se encuentra en tercera forma normal.

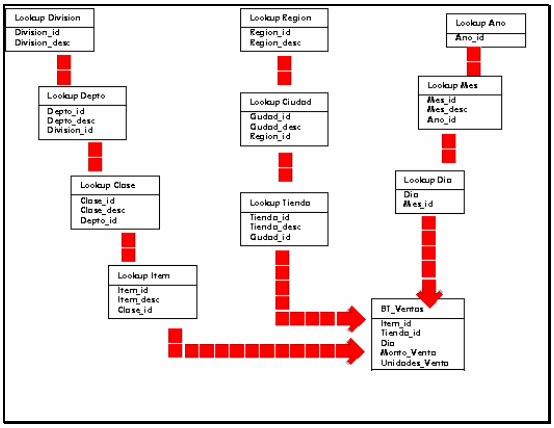

Fig. 4-9 Esquema Estrella Normalizado I

4.5.6 Esquema Estrella Normalizado II

A las tablas lookups se le incorpora los **ids** de **todos sus ancestros** (todas los ids de las tablas que están por encima de ella en la jerarquía de la dimensión). La estructura de la tabla base se mantiene inalterable con respecto al esquema anterior.

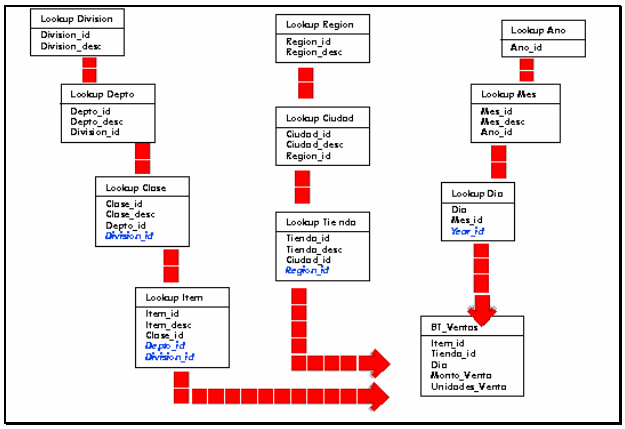

Fig. 4-10 Esquema Estrella Normalizado II

4.5.7 Esquema Estrella Normalizado III

Incluye en las tablas lookups no sólo los **ids** de sus ancestros sino también las **descripciones** de los mismos. La estructura de la tabla base no sufre modificaciones.

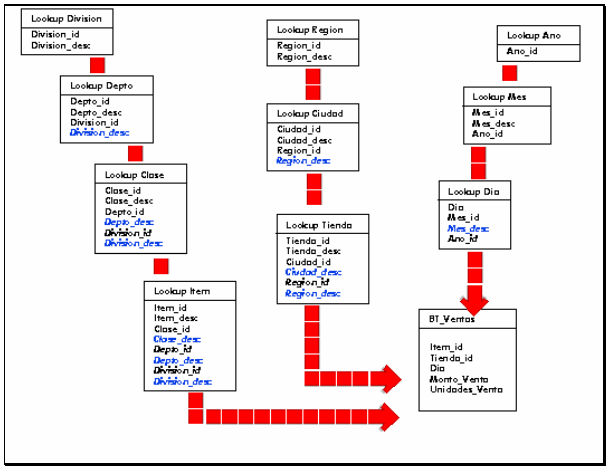

Fig. 4-11 Esquema Estrella Normalizado III

4.5.8 Comparación entre Esquemas Estrella Normalizados

El **EEN I** requiere **mínima cantidad** de espacio pero requiere **muchos joins** a la hora de responder consultas a niveles superiores de las jerarquías o al incluir análisis donde se muestra la descripción de padre e hijo, p.e., nuestro análisis anterior a nivel de Región y Provincia.

El **EEN II**, que involucra la desnormalización de los ids, requiere **espacio adicional** de almacenamiento pero **reduce** el número de **joins** necesario para resolver las consultas. Sólo realiza joins de tablas para mostrar descripciones.

Por último, el **EEN III**, con **desnormalización** de ids y descripciones, requiere **más espacio** de almacenamiento pero **elimina** la necesidad de **consultar más de una tabla** por dimensión. Las tablas lookups que no son hojas son utilizadas en consultas particulares que muestran sólo a ese nivel (como puede ser **tablas de agregación** o listado de elección de elementos de atributos -**pick lists**-).

Estos esquemas soportan la posibilidad de **múltiples fact tables** y niveles de agregación de diferente granularidad que hacen a estos esquemas ser esparsos o densos según la cantidad y criterio de las agregaciones.

Entre los beneficios de este tipo de esquemas podemos encontrar entonces la **flexibilidad** y **escalabilidad**, relaciones **M:N**, manejo óptimo de **pick lists**, **agregación** en tablas individuales (de fácil acceso y mantenimiento).

Como puntos negativos se encuentran el crecimiento en la **cantidad de tablas** y la **dificultad** para generar **SQL** de forma automática.

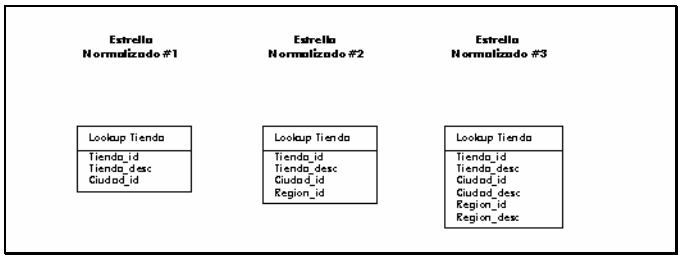

Fig. 4-12 Comparación de Esquemas Estrellas Normalizados

Capítulo 5 - Evolución de arquitecturas

Las arquitecturas de los data warehouses han evolucionado en base al crecimiento, en funcionalidad y rendimiento, de los DBMS. Es por ello, que la arquitectura reinante de los ’80 fue el modelo **MOLAP** (**Multidimensional OLAP**). Estos DWs funcionaban con bases de datos **propietarias** (comúnmente denominados “*cubos multidimensionales*”) dado que muy pocas bases de datos soportaban el tipo de trabajo al que eran sometidas en ambientes de data warehousing.

A comienzos de los ’90, con el gran crecimiento de los **RDBMS**, comienza a hacerse posible la utilización, en los ambientes de data warehousing, de las bases de datos relacionales en forma **directa** y no como fuente secundaria para el armado de cubos multidimensionales. Es así como surge una nueva arquitectura de DW denominada **ROLAP** (**Relational OLAP**) la cual ya se ha consolidado como la arquitectura de los ’90.

En las siguientes secciones analizaremos más en detalle las ventajas y desventajas de estas dos arquitecturas OLAP, **MOLAP** y **ROLAP**, como así también las variantes dentro de cada uno y los ambientes donde se adapta mejor cada una de ellas. Además describiremos brevemente otras arquitecturas alternativas que van surgiendo a medida que se incorpora nueva tecnología y se investiga en nuevos ambientes.

5.1 MOLAP

5.1.1 Capacidad de análisis

- Ofrece **vistas** de objetos **multidimensionales**
- Tiempo de **respuesta cero**, pues tiene todo precalculado.
- Si no se precalcula todo (en general todo el precálculo tiene volúmenes inaceptables) la capacidad de análisis se limita a aquellas porciones del cubo que fueron precalculadas. Existen variantes de los MOLAP donde en caso de no poder responder a una consulta, se precalcula el cubo que responde a esa pregunta y a la periferia, pudiendo llevar esta generación varias horas de construcción (generalmente batch).

5.1.2 Sistema de diseño propietario

- Requiere un cubo multidimensional propietario, especifico por vendedor, para almacenar los datos de forma multidimensional
- Generalmente el cubo se trata de una *"caja negra"* de datos encriptados que pueden residir de forma local o en un servidor MOLAP.
- El cliente interactúa contra la **MDDB** vía un **lenguaje de acceso propietario**.
- MDDB se alimentada desde una RDBMS.
- Flexibilidad y escalabilidad **limitados**.
- Cambios en el modelo dimensional del negocio implican la **generación** de todos los cubos nuevamente.
- Nuevos datos disponibles implican una **ventana batch** de carga bastante extensa para la generación de todos los cubos multimensionales.

5.1.3 Unidad de almacenamiento

- **Multidimensional data cube**, unidad de almacenamiento contra la cual se disparan las consultas.
  - Pre-agregación de todas las posibles intersecciones entre las dimensiones.
  - Rápida respuesta de consultas obviando la agregación en tiempo de ejecución.
- **Ineficiencia** a medida que crecen volúmenes de datos y/o dimensiones
  - Crecimiento exponencial ante cualquier adición al cubo multidimensional.
  - Reducido potencial de almacenamiento de datos (normalmente 5-10 GBs)
  - Reducido número potencial de dimensiones (normalmente 8-10)

5.1.4 Ambientes adecuados

- Modelos dimensionales **pequeños** y **estáticos**.
- Instalaciones dónde el tiempo de respuesta sea crítico.
- **Pocos** volúmenes de datos.
- Análisis de información a **nivel agregado**.

5.2 ROLAP

5.2.1 Capacidad de análisis

- Ofrece **vistas** de objetos **multidimensionales**
- Tiempos de respuestas que rondan entre los **segundos** y los **minutos**. Existen **técnicas de tuning**, **caching**, **materialización de vistas**, **indexación** y **esquema de diseño** que mejoran la performance de respuesta de los ROLAP.
- La capacidad de análisis abarca **todo** lo que se encuentra en el data warehouse o data-mart.

5.2.2 Sistema de diseño abierto

- Permite estrategias con **distintos** motores de base de datos.
- El cliente interactúa **directamente** contra el RDBMS vía **SQL**.
- Provee **flexibilidad** y **escalabilidad**.
- Los cambios en el modelo dimensional del negocio son trasladados al DW e **inmediatamente** se encuentra disponible para las consultas pertinentes.
- La ventana de carga del data warehouse es menor pues no existe el tiempo de generación de los multi-cubos.

5.2.3 Unidad de almacenamiento

- **Data Warehouse**, unidad de almacenamiento contra la cual son procesadas las consultas.
  - Permite almacenar la información en formato relacional.
  - Iguala la performance de los cubos MOLAP con un apropiado esquema de diseño.
  - Explota al máximo las capacidades de paralelismo de los RDBMS.
- No decae en **eficiencia** a medida que crecen los volúmenes de datos o números de dimensiones
  - No tiene límite inherente en cuanto a capacidad de almacenamiento (frecuentemente 100+ GBs). El límite esta dado por el RDBMS. Hay motores que están más preparados que otros para actuar en ambientes de data warehousing, aunque en la actualidad casi todos los motores ya incorporaron funcionalidad asociada a los data warehouses.
  - No tiene límite inherente en cuanto a cantidad potencial de dimensiones (frecuentemente 50+).

5.2.4 Ambientes adecuados

- Modelos dimensionales **grandes** y **dinámicos**.
- **Grandes volúmenes** de datos.
- Necesidad de análisis a **nivel transaccional**.

5.3 Otras arquitecturas

A medida que fueron consolidándose los ambientes OLAP empezaron a emerger modelos híbridos (**HOLAP, Hybrid OLAP**) los cuales trabajaban con su base de datos propietaria, al igual que los MOLAP, pero disparaban contra la base de datos relacional cuando alguna consulta no podía ser resuelta por los cubos precalculados retornando el resultado desde el RDBMS o generando el cubo correspondiente a una periferia de la respuesta para después retornar los datos desde el cubo generado. El problema de esta arquitectura es que la generación de ese segundo SQL que disparaba directamente contra la base relacional no se encontraba optimizado como para aprovechar las funcionalidades propias del motor y los tiempos de respuesta no eran los esperados. Dada la capacidad de algunos RDBMS de brindar ciertos servicios relacionados con ambientes OLAP, muchos MOLAP se auto definieron como HOLAP montando los *"cubos"* en bases de datos relacionales en lugar de arquitecturas propietarias.

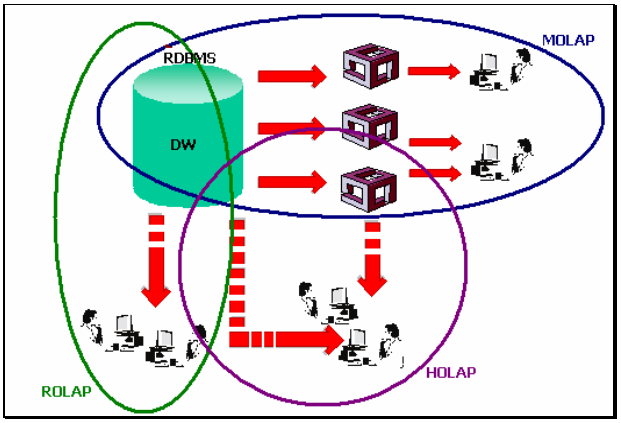

Fig. 5-1 Arquitecturas OLAP

Bibliografía

- [Bed97] Jeff Bedell, "Data Warehousing, Data Modeling and Design", DSSTraining, MicroStrategy, Inc., 1997.
- [Gol99] Matteo Golfarelli y Stefano Rizzi, 1999. A Methodological Framework for Data Warehouse Design. University of Bologna. ACM.
- [Inm92] William H. Inmon, 1992. "Building the Data Warehouse", Wiley-QED John Wiley & Sons, Inc..
- [Kim92] Ralph Kimball, 1992. The Data Warehouse Toolkit, Wiley Computer Publishing
- [Kim95] Ralph Kimball, 1995. Is ER Modeling Hazardous to DSS?, DBMS Magazine October.
- [Kim98] Ralph Kimball, Laura Reeves, Margy Ross, Warren Thornthwaite, 1998. The Data Warehouse Lifecycle Toolkit, Wiley Computer Publishing.
- [Pre93] Roger Pressman, 1993. Ingeniería del Software –Un Enfoque Práctico- . Tercera Edición. McGraw-Hill.
- [Ree00a] www.technologyevaluation.com A definition of Data Warehousing, Micheal Reed, August 24, 2000.
- [Ree00b] www.technologyevaluation.com The necessity of Data Warehousing, Micheal Reed, August 2, 2000.

Indice de Figuras