INTRODUCCIÓN AL DATAWAREHOUSING

Gestión de Datos
Ingeniería en Sistemas de Información
Profesor: Juan Zaffaroni
Universidad Tecnológica Nacional – Facultad Regional Buenos Aires

INDICE

Introducción

Desde siempre las empresas han tomado los datos de sus sistemas operacionales de forma tal de satisfacer las necesidades de información orientada a la toma de decisiones. Esto lo han hecho básicamente de dos maneras distintas:

- Han armado consultas, reportes o listados directamente sobre los mismos datos de los sistemas operacionales. Es el caso de los listados de cierre de caja en una sucursal bancaria.

- Han extraído datos desde sus bases de datos operacionales combinándolos de diversas formas y brindando herramientas a los usuarios para que ellos armen sus consultas o reportes a medida.

Podemos afirmar que un datawarehouse, a grandes rasgos, se construye a partir de recolectar los datos de los sistemas operacionales[1] y colocarlos en un repositorio corporativo centralizado. Reunir los elementos de datos apropiados desde diversos sistemas operacionales en un ambiente integral centralizado, simplifica el problema de acceso a la información y en consecuencia, acelera el proceso de análisis, consultas y el menor tiempo de uso de la información.

Para armar un datawarehouse, los datos extraídos son transformados para eliminar inconsistencias y se los resume o agrega.

Evolución de las consultas a los sistemas de información

- Años 60: Reportes Batch

La información era difícil de encontrar y analizar. Por otro lado, estos reportes no presentaban ninguna flexibilidad al usuario. Ante cada nuevo requerimiento del usuario era necesario reprogramar los reportes.

- Años 70: Decisions Support Systems (DSS) y Executive Information Systems (EIS) basados en terminales

Si bien estos sistemas presentaban información orientada a la toma de decisiones, siendo inflexibles para el usuario y cada nuevo requerimiento necesitaba reprogramarse.

- Años 80: Herramientas desktop de análisis de datos

En estos años aparecieron herramientas más fáciles de usar con GUIs más amigables, como las planillas de cálculo. El problema de estas

[1] O “sistemas transaccionales”

herramientas eran que tomaban los datos directamente de los sistemas operacionales.

- Años 90: Datawarehousing con motores y herramientas OLAP integradas

Definición de datawarehouse

Un datawarehouse tiene las siguientes características:

1. es una recolección de datos que se encuentran orientados a sujetos o temas.
2. sus datos se integran en un único repositorio desde diversas fuentes de la empresa.
3. sus datos almacenados son por lo general menos volátiles que en un sistema transaccional.
4. se usa para el soporte del proceso de toma de decisiones gerenciales.

Un datawarehouse es una base de datos orientada a la toma de decisiones que se mantiene en forma separada de las bases de datos operacionales de la organización.

Un datawarehouse es entonces una colección de datos orientados al sujeto, integrados, históricos y no volátiles que se usan principalmente en la toma de decisiones de las organizaciones.

Diferencias entre un datawarehouse y una base de datos operacional

Se enuncian a continuación las principales características que diferencian a un datawarehouse de una base de datos operacional:

	Base de Datos Operacional	Datawarehouse
Datos	Operacionales	Del negocio
Uso de los datos	Procesamiento repetitivo	Procesamiento analítico
Orientación del diseño	A la Aplicación (basada en Entidad Relación)	Al Tema o Sujeto (star schema, snowflake)
Estructura de datos	Muchas tablas altamente normalizadas	Pocas tablas con cierto grado de desnormalización.
Datos en el tiempo	Actuales	Actuales + históricos
Detalle de los datos	Altamente detallados	Detallados + resúmenes
Cambios en los datos	Continuos	Datos más estables
Cantidad de usuarios	Más que en Datawarehouse	Menos que en la operacional.
Tamaño de Base de Datos[2]	100 MB - GB	100 GB-TB
Cantidad de registros accedidos en una operación[3]	Decenas	Millones

Cuadro 1 – OLAP vs. Datawarehousing

[2] No pretende ser exacto, sino dar una idea de la relación entre los tamaños de las bases de datos. Si bien se solía incluir entre la definición de datawarehouse el hecho de ser bases de datos con gran volumen de información, la realidad demuestra que existen datawarehouses que no cumplen con esta característica por lo cual dicha definición era errónea.
[3] Nuevamente aplica el comentario del ítem anterior.

Ubicación del datawarehouse dentro de los sistemas corporativos

Diferentes autores clasifican a los sistemas de información en las siguientes categorías:

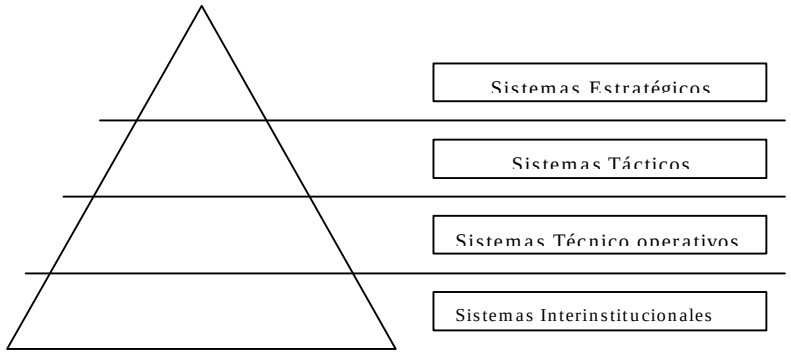


Gráfico 1 – Jerarquía de sistemas organizativos

Sistemas estratégicos

Están orientados a la toma de decisiones. Sus usuarios son, por ejemplo, directores o gerentes de la organización.

Sistemas tácticos

Diseñados para la coordinación de actividades y manejo de documentación. Facilitan la gestión de la información en los niveles intermedios de la organización. Ejemplos: e-mail, coordinación de agendas.

Sistemas técnico-operativos

Son los sistemas tradicionales orientados a la carga de datos del día a día. Ejemplos: sistemas contables, de facturación, de administración de personal, etc.

Sistemas interinstitucionales

Esta capa de sistemas, que recién está surgiendo, tienden a comunicar a la organización con su entorno o mercado. Ejemplo de estos sistemas son las extranets corporativas o la presencia en internet.

Un datawarehouse se ubica en la cima de la pirámide expuesta. Es decir, son bases de datos con información estratégica para la compañía, con lo cual lo podemos clasificar como un sistema estratégico. Es estratégico porque la información que brinda, como ya mencionamos, es orientada a la toma de decisiones. Los reportes que se emiten a partir de los datos almacenados en un datawarehouse permiten conocer aspectos de la evolución del negocio a un nivel macro.

Separación del DW respecto a las bases de datos operacionales

Una pregunta que puede surgir es por qué mantener separados a las bases de datos operacionales de la base de datos orientada a la toma de decisiones, o datawarehouse.

Esto se puede analizar desde el punto de vista de performance y funcionalidad.

Performance

Las bases de datos operacionales están diseñadas y optimizadas para cargas de trabajo y volumen de transacciones conocidos. Por otro lado, consultas analíticas complejas podrían degradar la performance de las operaciones transaccionales que son críticas para el funcionamiento del negocio ya que requieren analizar un conjunto de datos generalmente grande.

Por otro lado, la base de datos de un datawarehouse requiere tener los datos organizados de una manera particular que se analizará más adelante en este texto. También requiere de métodos de acceso a los datos necesarios para visualizar la información de forma multidomensional.

Funcionalidad

Los sistemas de tomas de decisiones requieren de información histórica que los sistemas operacionales, pensando en la performance, por lo general no mantienen. Además, los sistemas de soporte a la toma de decisiones requieren la consolidación (agregación, sumarización) de datos de muchas fuentes heterogéneas. Los datos de estas fuentes pueden ser inconsistentes entre sí ya que al ser sistemas distintos, puede haber representaciones, formatos y códigos de los datos que tienen que ser reconciliados.

Data Mart

Se define un data mart como una base de datos, o un conjunto de las mismas diseñadas para ayudar a gerentes / directores a tomar decisiones estratégicas sobre sus negocios.

Un Data Mart es un subconjunto lógico de un datawarehouse. Un datawarehouse está hecho de la unión de todos sus data marts. Más allá de esta simple definición lógica, se ve a menudo que un data mart es una restricción de un datawarehouse a un proceso o área de negocios de una compañía. Un data mart es probablemente construido por y para un área específica de una organización.

De esta manera, un supermercado puede tener un data mart orientado a compras, otro para el área de marketing, otro de ventas, etc. El datawarehouse de dicho supermercado se forma a partir de la unión de los datamarts.

Características de un Data Warehouse

Como se afirmó anteriormente, un Data Warehouse tiene las siguientes características:

1. Es orientado al tema

Los datos se clasifican en función de los aspectos que son de interés para la empresa. Esto diferencia a un datawarehouse respecto a una base de datos diseñada para un sistema operacional.

Por ejemplo, la base de datos de un sistema operacional se diseña alrededor de las aplicaciones y funciones de dicho sistema, tales como cuenta corriente, depósitos, extracciones, consumos de tarjeta de crédito, etc. Por ejemplo, una aplicación de ingreso de operaciones bancarias puede acceder a los datos sobre clientes, cuentas y transacciones. La base de datos combina estos elementos en una estructura que acomoda las necesidades de la aplicación.

En el ambiente de datawarehousing se organiza alrededor de sujetos tales como cliente, vendedor, producto y actividad. Por ejemplo, para un fabricante, éstos pueden ser clientes, productos, proveedores y vendedores. Para una universidad pueden ser estudiantes, cursos y docentes.

El diseño y la implementación de los datos encontrados en el datawarehouse son afectados por el hecho de que los mismos están pensados en función de los temas de interés.

Las aplicaciones están relacionadas con el diseño de la base de datos y del proceso. En datawarehousing se enfoca el modelo de datos y el diseño de la base de datos. El diseño del proceso (en su forma clásica) no es separado de este ambiente.
Estas diferencias afectan el nivel de detalle con los cuales los datos se encuentran almacenados.

En un datawarehouse, la información que no es usada por el proceso de toma de decisiones no se almacena, mientras que la información de los sistemas operacionales contiene datos para satisfacer de inmediato los requerimientos funcionales y de proceso, que pueden ser usados o no por el analista de soporte de decisiones.

2. Datos permanentes en el tiempo

Un datawarehouse contiene información histórica. Esto es porque toda la información del datawarehouse es requerida en cualquier momento (es decir, no "ahora mismo").

Los datos históricos son de poco uso en el procesamiento operacional. La información del datawarehouse por el contrario, debe incluir los datos históricos para usarse en la identificación y evaluación de tendencias.
La permanencia de los datos en el tiempo se presenta de diversas formas:

a) La información representa los datos sobre un horizonte largo de tiempo - desde cinco a diez años. El horizonte de tiempo representado para el ambiente operacional es mucho más corto - desde valores actuales hasta sesenta a noventa días.

Las aplicaciones que tienen un buen rendimiento y están disponibles para el procesamiento de transacciones, deben llevar una cantidad mínima de datos si tienen cualquier grado de flexibilidad. Por ello, las aplicaciones operacionales tienen un corto horizonte de tiempo, debido al diseño de aplicaciones rígidas.

b) La información del datawarehouse, una vez registrada correctamente, no puede ser actualizada.

Por supuesto, si los datos se han tomado incorrectamente, entonces pueden ser cambiados. Asumiendo que los datos se han guardado adecuadamente, los mismos no son alterados una vez guardados. En algunos casos puede estar contra la ética, e incluso ser ilegal, alterar los datos del datawarehouse. Los datos operacionales, siendo requeridos a partir del momento de acceso, pueden actualizarse de acuerdo a la necesidad.

3. Tiene información integrada

Quizás la característica más importante del ambiente de datawarehousing es que la información encontrada en su interior está siempre integrada.

La integración de datos se refiere a:

- Utilizar nombres consistentes
- Utilizar medidas uniformes en los datos
- Unificar los datos de fuentes múltiples
- Etc.

Una empresa puede contar con una número grande de sistemas operacionales. Quien desarrolló cada uno de dichos sistemas

operacionales puede haber utilizado un criterio propio de nombres de campos, valores para representar una misma cosa, etc. Un datawarehouse, al integrar los datos de dichos sistemas, debe unificar los criterios existentes. Por ejemplo:

a) Los diseñadores de aplicaciones codifican el campo SEXO de diferentes maneras . Un diseñador representa SEXO como una "M" y una "F", otros como un "1" y un "0", otros como "masc" y "fem" e inclusive, como "masculino" y "femenino".

No importa mucho cómo el SEXO llega al datawarehouse. Podría ser “M” y “F” o cualquier otra representación. Lo importante es que sea de cualquier fuente de donde venga, el SEXO debe llegar al datawarehouse en un estado integrado uniforme.

Por lo tanto, los datos que se cargan al datawarehouse deben ser convertidos a ese formato estándar definido. Si se definió que el campo SEXO en el datawarehouse se representa como “M” y “F”, entonces se deben convertir aquellos datos que vengan de otra manera a este formato definido.

b) Los diseñadores y programadores de aplicaciones miden las unidades de medida de tuberías en una variedad de formas. Diferentes programadores de aplicación pueden almacenar los datos de tuberías en una tabla en centímetros, otros en pulgadas, otros en millones de pies cúbicos por segundo y otros en yardas.

Al dar medidas a los atributos, la transformación traduce las diversas unidades de medida usadas en las diferentes bases de datos para transformarlas en una medida estándar común.

Cualquiera que sea la fuente, cuando la información de la tubería llegue al datawarehouse necesitará ser medida de la misma manera.

c) El mismo elemento de datos puede obtenerse desde fuentes múltiples. En este caso, el proceso de transformación debe asegurar que la fuente apropiada sea usada, documentada y movida al depósito.

Esto quiere decir que si el campo “unidades vendidas” puede ser obtenido desde diversos sistemas operacionales o sistemas fuente, se debe asegurar que el sistema elegido sea el apropiado. Además esto debe documentarse adecuadamente.

4. Los datos son no volátiles

En un datawarehouse, la información es útil sólo cuando es estable. Los datos operacionales cambian sobre una base momento a

momento. La perspectiva más grande, esencial para el análisis y la toma de decisiones, requiere una base de datos estable.

Un datawarehouse tiene solamente dos tipos únicos de operaciones sobre sus datos: la carga inicial y el acceso a los mismos. No hay actualización de datos (en el sentido general de actualización), como una parte normal de procesamiento, como sí ocurre en los sistemas operacionales.

Componentes de una arquitectura de Datawarehousing

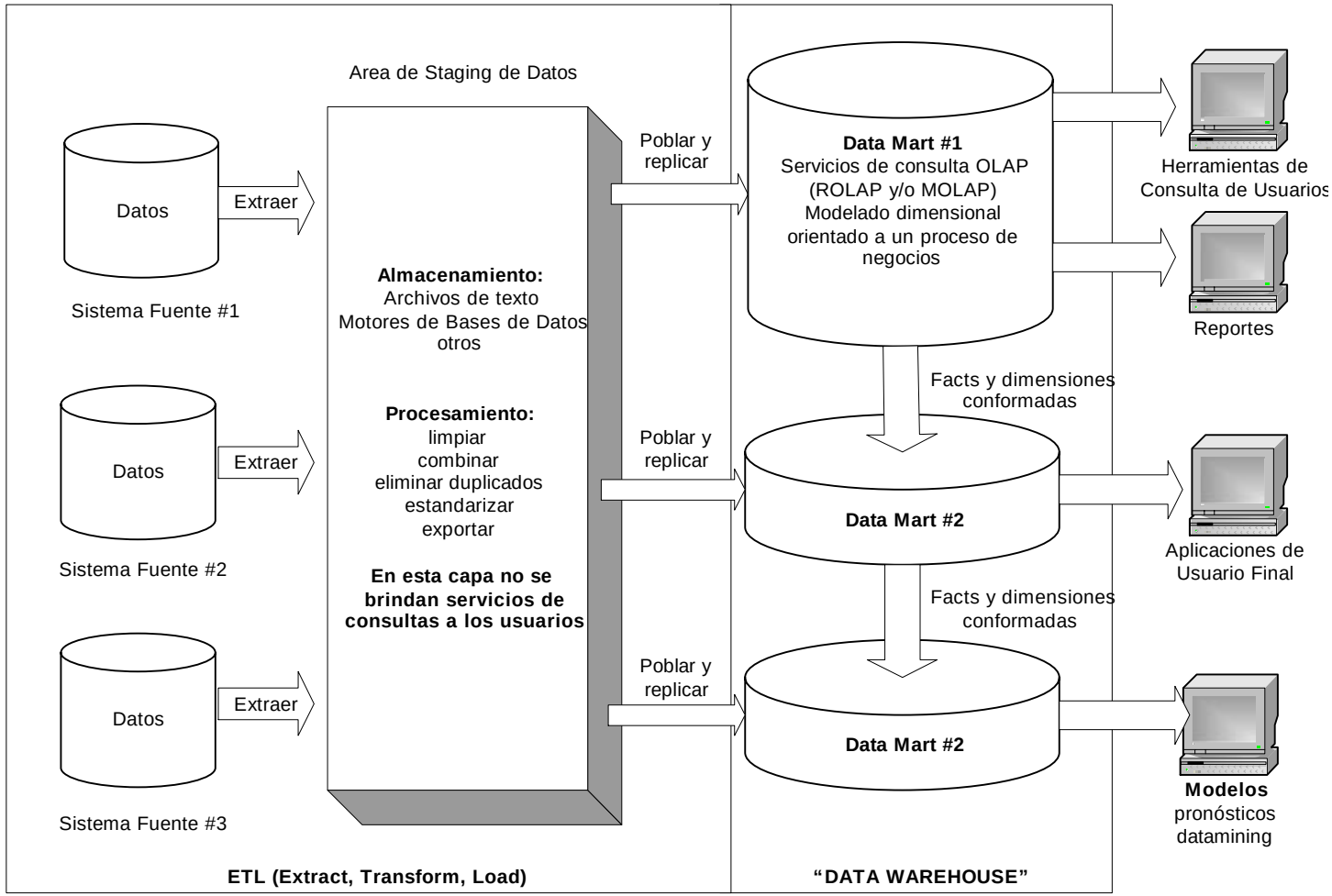


Gráfico 2 – Arquitectura de un Datawarehouse

Como se mencionó en los apartados anteriores, un datawarehouse está compuesto por datos provenientes de diversos sistemas operacionales (o también llamados transaccionales). Cuando se habla de poblar un datawarehouse, se hace referencia al proceso de tomar los datos de dichos sistemas para cargarlos en el datawarehouse. Ahora bien, los datos crudos por sí mimos, en la base de datos del datawarehouse no son útiles para ser presentados. Para presentar los datos al usuario final existen diversas alternativas, algunas de ellas a través de desarrollos a medida o también mediante herramientas existentes en el mercado desarrolladas a tal fin.

Capas de sistemas fuente y staging de datos

Sistema Fuente (Source System)

Denominamos sistema fuente a todo aquel sistema que proporciona datos al datawarehouse para su análisis y explotación.

Como ejemplo de un **sistema fuente** (los sistemas operacionales o transaccionales nombrados anteriormente son sistemas fuente) podemos mencionar a los existentes en los puestos de caja de un supermercado, que se encuentran permanentemente generando registros en bases de datos transaccionales. En la pirámide expuesta en el primer apartado, podemos clasificar a estos sistemas como técnico-operativos.

La prioridad fundamental de un sistema de caja de un supermercado es que se encuentre permanentemente funcionando y los tiempos de respuesta sean válidos para los usuarios. Por otro lado, las consultas que se realizan sobre los movimientos de los puestos de caja son escasas y dado el volumen de la información a almacenar, estos sistemas tienden a no contener información histórica.

Los sistemas fuente son generalmente sistemas operacionales o transaccionales (aunque otros sistemas también pueden ser fuente de datos sin ser de estas clases) cuya función es capturar las transacciones de un negocio. Un sistema fuente es frecuentemente llamado “sistema legacy” en los ambientes de mainframe.

Las máximas prioridades de un sistema fuente son que se encuentre funcionando y disponible. Por otro lado, las consultas a los sistemas fuentes son escasas y por lo general contienen muy poca información histórica.

En cuanto a la base de datos de los sistemas fuente, se las suele denominar bases de datos OLTP (on-line transaction processing).

Por lo general, se puede ver que los sistemas transaccionales cuentan con su base de datos modeladas bajo el esquema de entidad-relación.

Estas bases de datos se encuentran en mayor o menor medida normalizadas de forma tal de evitar duplicidades de datos, optimizando espacio de almacenamiento.

Algunos ejemplos de sistemas fuentes son SAP, Peoplesoft, el ERP de Oracle, un sistema de caja de una sucursal bancaria, o cualquier otra aplicación que una empresa tenga desarrollada a medida con el fin de capturar las transacciones del negocio.

Area de Staging (almacenamiento intermedio) de Datos

Como ya se mencionó anteriormente, cuando se quiere analizar el funcionamiento de una empresa o de un área en particular, es necesario obtener datos de distintos sistemas fuente para analizarlos en su conjunto.

Volviendo al caso de un supermercado, quizás algún directivo o gerente del mismo desee obtener un reporte que indique cuál ha sido el efecto de publicitar una determinada oferta de un producto por televisión. Para poder lograr esto, puede ser necesario integrar los datos de los sistemas transaccionales de caja (que indiquen cantidad de unidades vendidas de un producto) con los sistemas de marketing (que indiquen cuando y en qué canal apareció la oferta).

Por otro lado, podría darse el caso de que para un mismo producto el código que utiliza el sistema de caja sea distinto al código que utiliza el sistema de marketing con lo cual sea necesario integrarlos. O quizás marketing no cuente con un sistema transaccional, sino que los datos se encuentren en planillas Excel.

En definitiva, la función de un Area de Staging de Datos es recibir los datos de los sistemas transaccionales al fin de limpiarlos, transformarlos, combinarlos, integrarlos y eliminar datos duplicados preparando los mismos para ser usados en un datawarehouse o data mart. Es un área de almacenamiento y un conjunto de procesos orientados a realizar estas tareas.

Este conjunto de procesos se conoce comúnmente como procesos de ETL (Extract, Transform, Load – Extracción, Transformación y Carga). En el mercado existen varias empresas que ofrecen herramientas de ETL capaces de integrar datos de diversos sistemas fuente. Como ejemplo, las cinco herramientas con mayor market share en el mercado mundial son: Informatica, Ascential Datastage, SAS, Oracle Warehouse Builder y Genio.

Capa del datawarehouse propiamente dicho

El datawarehouse es una base de datos modelada bajo un esquema denominado "dimensional".[4]

El modelado dimensional es una disciplina específica para modelar datos que es una alternativa al esquema de modelado de entidad-relación.

Un modelo dimensional contiene la misma información que un modelo de entidad-relación pero almacena los datos en un formato simétrico cuyos objetivos de diseño apuntan al entendimiento de los mismos, la performance de las consultas y la resistencia al cambio.

Los componentes de un modelo dimensional se analizan más en detalle posteriormente, pero los podemos definir brevemente:

Fact Table[5]

Es la tabla primaria en cada modelo dimensional, orientada a contener las medidas del negocio. En este apunte, utilizaremos la palabra métrica[6] para representar una medida del negocio. En bibliografía en inglés este término aparece como fact, measure o measurement.

Cada fact table representa una relación de muchos a muchos y contiene un conjunto de dos o más foreign keys (claves foráneas) que hacen referencias a sus respectivas tablas de dimensión.

Dimension Table[7]

La Dimension Table restringe los criterios de selección de los datos de la fact table.
Cada dimensión está definida por su clave primaria que sirve como base para la integridad referencial con la Fact table con la cual hace join. Muchas tablas de dimensión contienen atributos textuales (campos) que son la base para restringir y agrupar las consultas sobre el data warehouse.

Integración de las Dimension Tables con las Fact Tables

Una métrica está siempre en función de las dimensiones. Supongamos que nos interesa el importe vendido por fecha y sucursal. "Importe vendido" es una métrica por lo que residirá en la

[4] Posteriormente se explica el por qué del nombre "dimensional"
[5] En castellano se la denomina también Tabla de Hechos. Usamos la nomenclatura en inglés ya que es la más utilizada. También puede aparecer como tabla de métricas o de indicadores.
[6] También se denomina indicador para algunos autores.
[7] También denominada por algunos autores como Lookup Table.

fact table. Y lo hará en función de los valores de las dimensiones que son fecha y sucursal.

Algebraicamente:
Importe Vendido = f (fecha, sucursal)
Generalizando:
Métrica = f(d1, d2, … , dn)

Las dimensiones son las variables de entrada de la función. Si se representara esto gráficamente nos quedaría una función de dos variables (o dimensiones).

Si vemos la representación algebraica, en el gráfico siguiente se puede entender por qué se emplea el término "modelado dimensional". Más aún, la implementación física del modelo dimensional es comunmente denominada "cubo" cuando se implementa sobre bases de datos multidimensionales[8].

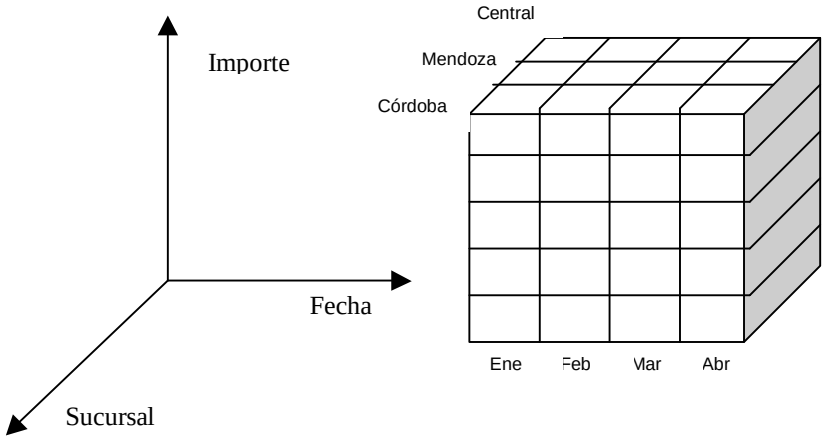


Gráfico 3 – Dimensiones y métricas gráficamente

Operaciones en un modelo de datos multidimensional

Agregación (roll-up)

Agregar los datos implica sumarizarlos en función de alguna de las dimensiones. Por ejemplo, si los datos se encuentran almacenados por sucursal y día, agregarlos implicaría, por ejemplo, obtener el total de ventas por sucursal.

[8] Se define el concepto posteriormente.

Selección (slice)

La selección implica generar un subcubo del cubo original. Por ejemplo, al querer calcular las ventas donde sucursal = 'Córdoba' y meses = Ene, Feb, Marzo. En este caso se arma un subcubo del original.

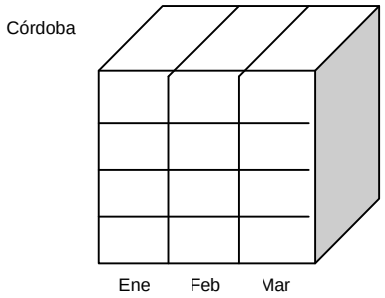


Gráfico 4 - Subcubo

Navegación a datos de mayor detalle (drill-down)

Hacer drill-down denota, estando en un reporte donde se visualizan las ventas de Enero, Febrero y Marzo, seleccionar Enero y ver el detalle de las ventas diarias para ese mes. Es subir el detalle de los datos (o lo que es lo mismo, bajar de nivel en la jerarquía de navegación).

Otras operaciones de visualización (por ejemplo, Pívot)

Hacer pívot se refiere a rotar las dimensiones en las cuales se visualizan los datos. En el caso donde se esté visualizando un reporte donde las filas representan ciudades y las columnas meses, hacer pívot implicaría colocar los meses en las filas y las ciudades en las columnas.

Capa de usuario final

Esta capa está compuesta por todas aquellas herramientas que permiten al usuario visualizar la información contenida en el datawarehouse. Dicha información es usualmente presentada en forma de reportes o gráficos.

OLAP (On-line Analytic Processing)

La actividad general de consultar y presentar datos de un datawarehouse, está implementada por un conjunto de empresas vendedoras de los llamados "productos OLAP". La tecnología que utilizan estas empresas es generalmente no relacional y casi siempre se basan en un cubo multidimensional de datos., aunque existen diferentes empresas de tecnología que implementaron esquemas

OLAP simulando un cubo multidimensional en una base de datos relacional.

MOLAP (multidimensional OLAP)

Cuando la tecnología OLAP se monta sobre una base de datos multidimensional se la suele denominar MOLAP.

Los datos se almacenan en estructuras de almacenamiento basadas en arrays. Las herramientas MOLAP brindan acceso directo a estructuras de datos de este tipo.

ROLAP (relational OLAP)

Es un conjunto de interfases de usuario y aplicaciones que le dan a una base de datos relacional un aspecto de base de datos multidimensional. Es decir, se simula un cubo dimensional en una base de datos relacional.

Los vendedores de esta clase de tecnología utilizan herramientas middleware OLAP para soportar las piezas faltantes al utilizar una base de datos relacional para un datawarehouse. Estas herramientas optimizan la utilización de la base de datos y adicionan la lógica de navegación y de agregaciones,

HOLAP (hybrid OLAP)

Combina características de las tecnologías MOLAP y ROLAP.

Procesos básicos de un Datawarehouse

Una vez vistos los componentes de un datawarehouse, podemos analizar los diferentes procesos de un esquema de datawarehousing:

Extracción: la etapa de extracción es el primer paso para obtener datos hacia un datawarehouse. Extraer significa leer y entender los datos de los sistemas fuente y copiar las partes que se necesiten al área de staging intermedio de datos para su posterior uso.

Transformación: una vez que los datos son extraídos existen numerosos procesos de transformación que se les pueden aplicar. Algunos son:

1. Limpiar los datos corrigiendo problemas de tipeo, resolviendo problemas de dominio (tal como un nombre de ciudad incompatible con un código postal), tratar elementos de datos faltantes y estandarizar los datos.
2. Purgar los campos seleccionados de los sistemas fuentes que no son útiles para el data warehouse.
3. Combinar diversas fuentes de datos, haciendo coincidir los valores claves o realizando matcheos por campos no clave.
4. Crear claves subrogadas para cada registros de dimensión de forma tal de evitar la dependencia en claves de los sistemas transaccionales.
5. Realizar agregaciones y sumarizaciones.

Carga e indexado: cargar el datawarehouse es replicar las tablas de dimensión y las de facts utilizando comunmente bulk loading (carga masiva). La carga masiva es importante, en contraste con la carga por registro la cual es más lenta. El datawarehouse o data mart debe entonces indexar los nuevos datos para la performance de las consultas.

Aseguramiento de la calidad: cuando el datawarehouse ha sido cargado, indexado y correctamente agregado, el último paso antes de la publicación es el aseguramiento de la calidad. Esto puede hacerse con reportes donde se verifiquen los totales.

Publicación: en este paso, los usuarios acceden a la información del datawarehouse.

ETL

Introducción

Las herramientas de extracción transformación y carga (ETL – Extraction-Transformation-Loading) , son piezas de software responsables de la extracción de los datos de distintas fuentes, su depuración, customización e inserción dentro de un datawarehouse.

En este capítulo nos focalizamos en el problema de la definición de las actividades del ETL.

Se calcula que el ETL y la limpieza de datos puede representar hasta el 80 % de los costos de el tiempo de desarrollo en un proyecto de datawarehousing. Si bien existen herramientas orientadas a estos procesos, debido a la complejidad y la larga curva de aprendizaje de estas herramientas, muchas empresas prefieren realizar desarrollos in-house para las tareas de ETL y depuración de datos.

En general, el diseño, desarrollo y puesta en marcha de procesos de ETL es subestimado en estos procesos. Los procesos de ETL necesitan de modelado, diseño y metodologías.

En la siguiente figura representamos el ámbito general de los procesos de ETL. En la capa inferior se representan los almacenamientos de datos que están involucrados en el proceso. En la parte izquierda se representan las fuentes originales de datos (generalmente bases de datos relacionales y archivos). Los datos de estas fuentes son extraídos por rutinas de extracción. Luego, estos datos son propagados al Area de Staging de Datos (DSA), donde se transforman y depuran antes de ser cargados al datawarehouse.

El datawarehouse es la parte derecha del gráfico, y contiene las tablas facts y de dimensiones.

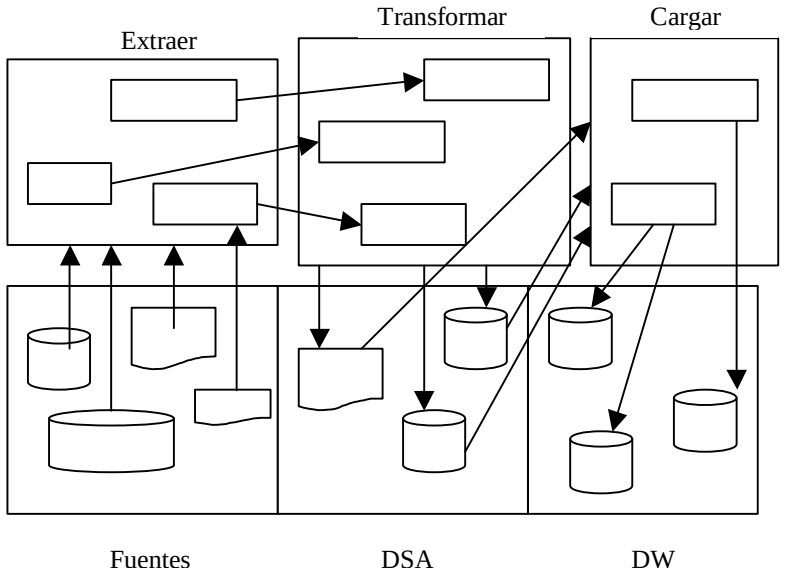


Gráfico 5 – Procesos de ETL

En las primeras etapas del diseño de un datawarehouse, el diseñador debe enfocarse a dos tareas principales que son prácticamente realizadas en paralelo. La primera de estas tareas implica la recolección de los requerimientos por parte de los usuarios. La segunda tarea de igual importancia para el éxito del proyecto, implica el análisis de la estructura y contenido de las fuentes de datos existentes y su mapeo al modelo del datawarehouse.

El diseño del proceso de ETL finaliza en la producción de un entregable fundamental: el documento de mapeo de los atributos de las fuentes de datos a las tablas del datawarehouse.

Producir este entregable implica varias entrevistas que resultan en la revisión y redefinición de los mapeos. Para esto hay que analizar las estructuras de datos de los sistemas fuentes. Esto implica dos tareas:

- Determinar las interrelaciones entre los atributos y los conceptos de los sistemas fuentes.
- Determinar cuales son las transformaciones necesarias que hay que realizar a los datos para cargarlos en el datawarehouse.

Conceptos y atributos

Atributos: Son un módulo granular de información. El rol de los atributos es el mismo que en el modelo de entidad-relación.

Conceptos: un concepto representa una entidad en las bases de datos de los sistemas fuente o en el datawarehouse. Las instancias de conceptos pueden ser archivos o tablas de las bases de datos fuentes, o dimensiones o facts del datawarehouse. Un concepto está formalmente definido por un nombre y un conjunto finito de atributos. En los términos de un modelo de entidad relación, un concepto es la generalización de entidades y relaciones; dependiendo del modelo utilizado, todas las entidades compuestas de un conjunto de atributos son instancias de la clase Concepto.

Un proceso de ETL puede tratar varias estructuras de almacenamiento físico como listas finitas de campos, incluyendo bases de datos relacionales, archivos COBOL o simplemente archivos ASCII.

Transformaciones (Transformations) y Restricciones (Constraints)

Dos grandes categorías de transformaciones de datos incluyen:

- Operaciones de filtrado o depuración de los datos, como el chequeo de violaciones de primary o foreign key.
- Sumarizaciones o agregación de los datos.

Formalmente, una transformación es definida por:

- Un conjunto finito de atributos de entrada.
- Un conjunto finito de atributos de salida.
- La transformación propiamente dicha.

Ejemplos de transformaciones son:

1. Filtrar los montos de venta menores a 100 $ y no cargarlos en el datawarehouse.
2. Transformar el campo fecha_venta que viene del sistema transaccional en formato AA-MM-DD a formato DD/MM/AAAA.
3. Eliminar aquellos registros provenientes de la base de datos A, cuyo campo ciudad no tenga un valor válido exista en la tabla ciudades de la base de datos B.
4. Sumarizar los datos de importe de venta que provienen del sistema fuente A en forma diara a mensual, si es que el datawarehouse solo guarda datos mensuales.

Formato de un documento de mapeo

Se mencionó anteriormente que el documento de mapeo es uno de los entregables fundamentales en el diseño de un datawarehouse. Una implementación simple de un documento de mapeo es una grilla que indique las tablas (conceptos) , atributos y transformaciones:

Concepto destino	Atributo destino	Transformación
D_Cliente	Id_cliente	Customers.id[1,4]
D_Cliente	ApeNom	Customers.surname : “, “ : Customers.name
D_Cliente	Fecha_nac	FormatearFecha(Customers.date, “YYYY-MM-DD”)
D_Cliente	Sexo	If (Customers.gender = ´1´) then ´M´else ´F´
D_Cliente	Nombre_Vendedor	1) Salesmen.surname : “, “ : Salesmen.name

Restricción: Customers.date > 1-1-1970.

Joins: 1) Customers.sales_id = Salesmen.sales_id

Cuadro 2 – Documento de mapeo

Este ejemplo simple indica que se van a poblar los atributos id_cliente, ApeNom, FechaNac, Sexo, Nombre_Vendedor de la dimensión d_cliente .

La restricción en este caso indica que se deben tomar del sistema fuente todos aquellos clientes cuya fecha de nacimiento sea mayor al 1 de Enero de 1970 (Restricción: Customers.date > 1-1-1970)..

Transformaciones

1) Customers.id[1,4]: El campo id_cliente se obtiene de tomar los primeros cuatro dígitos del campo id de la tabla Customers, que es una tabla del sistema fuente.
2) Customers.surname : ", " : Customers.name: el campo Apenom se obtiene de concatenar (operador ´:´) los campos surname y name de la tabla Customers. El formato del campo ApeNom sería entonces Perez, Juan.
3) FormatearFecha(Customers.date, "YYYY-MM-DD"): el campo fecha_nac se obtiene de formatear el campo date de la tabla customers a YYYY-MM-DD.
4) If (Customers.gender = ´1´) then ´M´else ´F´: el campo sexo se obtiene en función del valor del campo gender. Si gender es 1 entonces es M, caso contrario es F.
5) Salesmen.surname : ", " : Salesmen.name: el campo nombre_vendedor se obtiene de concatenar los campos surname y name de la tabla salesmen. Pero para poder acceder a la tabla salesmen se debe especificar la condición de join. En este caso, el join es Customers.sales_id = Salesmen.sales_id.

Calidad de datos:

Analicemos la transformación 4).. Como se puede ver, el proceso de ETL espera que del sistema fuente provenga un 1 o un 0. Ahora, que pasaría si algún registro del sistema fuente, no contiene esos valores y contiene, por el contrario, un nulo o cualquier otro valor. El proceso de ETL debe contemplar que puedan darse esos casos y corregirlos. En ese caso pueden tomarse diferentes acciones. Una de ellas sería enviar el registro con valor inválido a un archivo o tabla de rechazados para corregir el error y reprocesarlo posteriormente.

Para detectar estos casos de error existen herramientas en el mercado orientadas al análisis de calidad de datos. El detalle de dichas herramientas no está dentro del alcance del presente texto.

Modelado de un datawarehouse

Una vez presentado el datawarehouse en cuanto a su definición, utilidad y procesos, este apartado presenta las particularidades del mismo en cuanto a su estructura de base de datos.

Para esto analizaremos los dos componentes del modelo de un datawarehouse: el modelado lógico y el modelado físico:

Modelo lógico

Un modelo lógico es similar a lo que es un mapa cuando uno decide realizar un viaje. Uno necesita saber donde va y como llegar allí. Se necesita un plan, uno que sea visible y correcto.

El modelo lógico es un esquema de todas las piezas de información necesarias para entender los datos y como se relacionan con el negocio. Es una técnica fundamentalmente gráfica que resulta en un modelo de datos representando la definición, características y relaciones de los datos en un ambiente de negocios, técnico o conceptual. En términos simples, un modelo de datos lógico es un borrador de alto nivel de los datos.

Los modelos lógicos de datos son independientes de los dispositivos físicos de almacenamiento de datos. Esta es la clave del modelado lógico. La razón por la cual un modelo lógico de datos debe ser independiente de la tecnología es simplemente porque la tecnología cambia muy rápidamente. Lo que ocurre por debajo del modelo lógico de datos puede cambiar con la tecnología, pero el esquema de los datos debe permanecer igual.

Un modelo de datos lógico es un arreglo de los datos orientados al usuario general, opuesto al módelo físico, el cual arregla los datos pensando en el uso eficiente de la base de datos. El ámbito y complejidad de un modelo de datos lógico depende de los requerimientos de los usuarios y la disponibilidad de los datos fuente. Cuanto más sofisticados y complejos sean los requerimientos de reportes por parte de los usuarios, probablemente más complejo será el modelo lógico de datos.

Componentes de un modelo lógico

Un modelo de datos lógico es una representación de los siguientes conceptos derivados de los enunciados en el apartado anterior:

- Facts
- Atributos
- Jerarquías

Facts

Una de las primeras cosas que se deben hacer cuando se crea un modelo lógico de datos es determinar cuáles son las facts. Las facts brindan información al combinar una serie de atributos.

Conceptualmente, puede pensarse a las facts como medidas del negocio que son generalmente numéricas y agregables. La cantidad de unidades vendidas, el importe de las ventas, el inventario, el saldo de una cuenta, son algunos ejemplos de facts.

En un datawarehouse, las facts propiamente dichas son columnas de una tabla denominada "fact table". Los datos de estas facts pueden provenir de diferentes sistemas fuente y pueden tener diferentes niveles de detalle o agregación. Por ejemplo, se podría capturar en una fact table la venta diaria de una sucursal de un supermercado o tener esa misma información agregada mensualmente. Por supuesto, el nivel de agregación de la información es importante ya que determina el nivel de máximo detalle en el cual el usuario podrá ver la información.

Para aquellos que ya están familiarizados con SQL, las facts generalmente representan columnas numéricas en tablas en las cuales se realizan agregaciones tales como SUM y AVG.

Por ejemplo, en la siguiente sentencia SQL, la columna ORDER_AMOUNT en el datawarehouse es una fact:

```
SELECT EMP_NAME, SUM(ORDER_AMOUNT)
        FROM ORDER_FACT of, LU_EMPLOYEE le
        WHERE of.EMP_ID = le.EMP_ID
```

Atributos

Una vez que se determinan las facts, se deben identificar los atributos. Los atributos permiten contestar preguntas sobre una fact y proporcionan un contexto en el cual mostrar esas facts.

Por ejemplo, consideremos la fact Importe de las Ventas. Si a un usuario se le informó que la compañía tuvo ventas de 10.000 $, esto no es información demasiado útil. Para darle significado a ese número es necesario saber más sobre el origen del mismo, tal como:

- El período de tiempo en el cual se realizaron las ventas.
- Quien y cuanta gente contribuyó a la venta total.
- Qué productos fueron vendidos en cuál sector de la empresa o departamento.
- El ámbito de la venta, si fue nacional, regional, local o de un único local.
- La moneda que se usó para las compras.

Para contestar estos tipos de preguntas sobre las facts, los atributos proporcionan categorías y niveles de sumarización y clasificación de los datos. Los atributos se usan para contestar preguntas del negocio sobre hechos (o facts) ocurridos a distintos niveles de detalle. Por ejemplo, si los datos de las ventas están almacenados a nivel día, un atributo Mes permite ver los mismos datos sumarizados al nivel de mes.

Para aquellos ya familiarizados con SQL, los atributos generalmente representan las columnas no numéricas y no sumarizables de las tablas de una base de datos. Estas columnas se usan para clasificar y agrupar los datos fact.

Por ejemplo, en la siguiente sentencia SQL, la columna MONTH_ID del datawarehouse se correspondería al atributo Mes:

```
SELECT ms.MONTH_ID, m.MONTH_DESC, SUM(ms.TOT_SALES)
        FROM MONTHLY_SALES ms, MONTHS m
        WHERE ms.MONTH_ID = m.MONTH_ID AND
               ms.MONTH_ID in (200301, 200302,
      200303)
        GROUP BY ms.MONTH_ID, m.MONTH_DESC
```

Relaciones entre los atributos

Las relaciones entre los atributos son esenciales en el modelo de datos lógico. Sin relaciones, no habría interacción entre los datos. Las relaciones le dan significado a los datos proporcionando asociaciones lógicas de atributos basados en reglas de negocio.

Cada relación directa entre los atributos tiene dos partes: un padre y un hijo. Un hijo debe siempre tener un padre, y un padre puede tener múltiples hijos. El atributo padre tiene un nivel superior que el hijo. Por ejemplo, en una relación entre los atributos Año y Mes, el Año es el padre y el Mes es el hijo. Como podemos ver, un mes determinado pertenece a un único año y un año tiene múltiples hijos (12 para ser más precisos).

Atributos relacionados

Pueden existir tres tipos de relaciones entre atributos directamente relacionados.

- Uno a Uno: cada elemento del atributo padre tiene un solo hijo. Un ejemplo de esto es la relación entre Persona / Empresa y Número de CUIT. Una persona / empresa puede

tener un único número de CUIT y un número de CUIT pertenece a una única persona / empresa.

- Uno a muchos: cada elemento en el atributo padre se corresponde a varios elementos en el atributo hijo. Es el tipo de relación más común. Es el caso de Año y Mes.

- Muchos a muchos: cada elemento en el atributo padre puede tener múltiples hijos, y cada hijo puede tener múltiples padres. Es el caso de la relación entre Persona y Cuenta Bancaria. Una persona puede tener varias cuentas bancarias y una cuenta bancaria puede pertenecer a varias personas o cotitulares.

Jerarquías de atributos

Las jerarquías en un modelo de datos lógico son agrupamientos de atributos ordenados para reflejar la relación entre los mismos.

Un ejemplo de jerarquía que por lo general aparece en todos los datawarehouse es la jerarquía Tiempo:

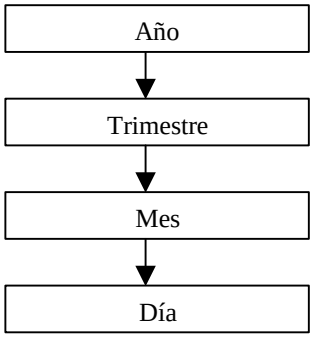


Gráfico 6 – Jerarquía tiempo

En un modelo de datos lógico, las jerarquías contienen atributos que están directamente relacionados uno con otro. Los atributos de una jerarquía no están directamente relacionados con atributos de otra jerarquía sino a través de las tablas de facts (se verá más sobre esto más adelante).

Esquema de jerarquías en un modelo de datos de un datawarehouse

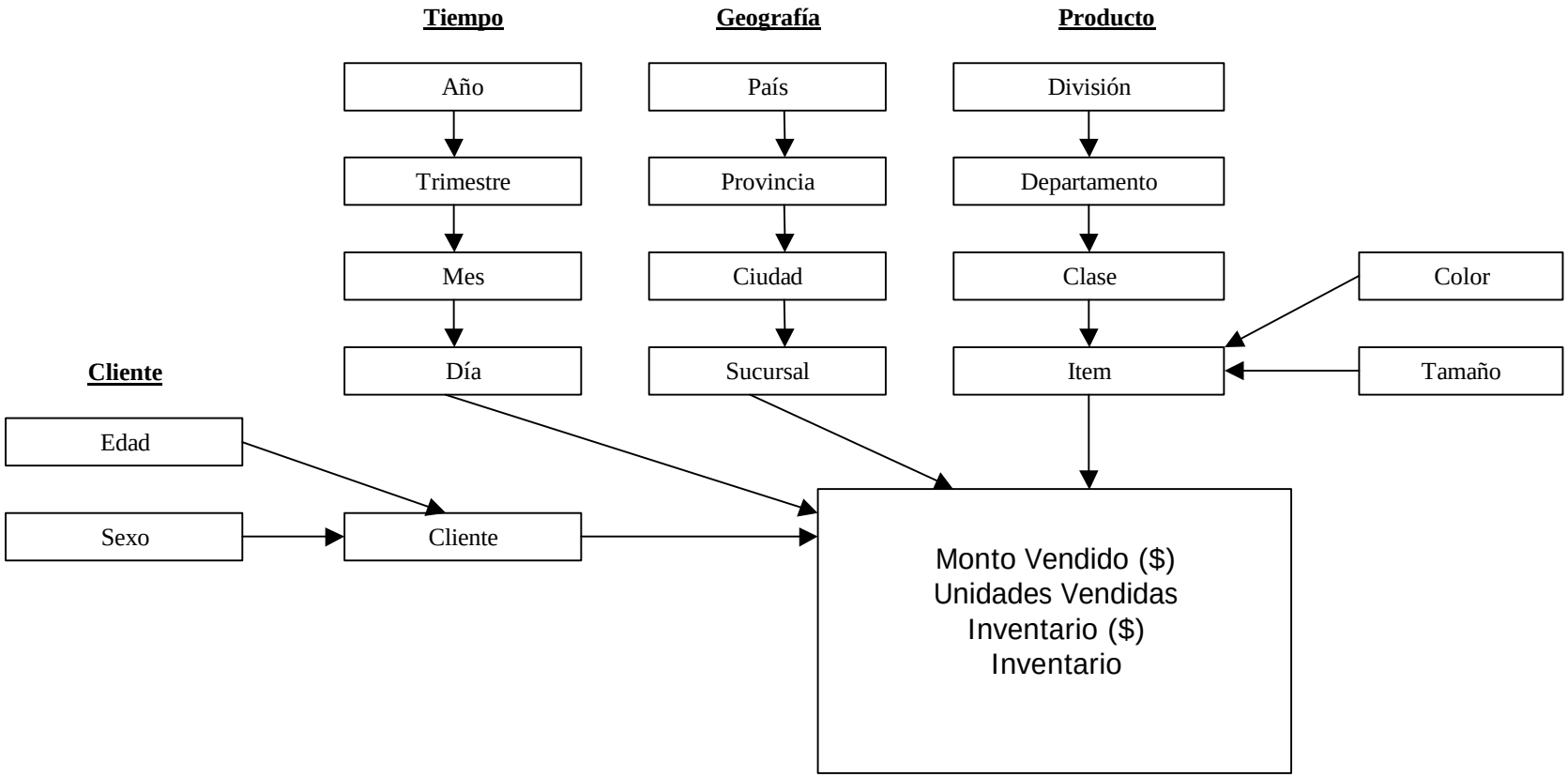


Gráfico 7 – Jerarquías en un modelo de datos de un datawarehouse

Con los conceptos ya vistos, al ver este modelo de datos se podrían responder las siguientes preguntas:

¿ Cuáles son los atributos ?
Algunos atributos son Sexo, Edad, Año, País, Clase, Item, Color, etc.

¿ Cuáles son las jerarquías ?
Las jerarquías de este modelo son Cliente, Tiempo, Geografía y Producto.

¿ Cuales son los hechos o facts ?

Los hechos o facts son Monto Vendido en $, Unidades Vendidas, Inventario en $, Inventario en unidades.

¿ Cuáles son algunas de las preguntas que le permitiría este modelo responder a un usuario ?

- ¿ Cuál fue el Monto Vendido en $ durante el año 2002 ?
- ¿ Cuál fue el Monto Vendido en $ durante Octubre, Noviembre y Diciembre del año 2002, de los productos A, B y C ?
- ¿ Cuál fue la cantidad de productos A de color amarillo y tamaño médium vendidos en la región norte a personas de entre 30 y 45 años de sexo masculino durante la primera quincena del mes de Marzo del año 2002 ?
- ¿ Cómo evolucionó el inventario de productos en cantidad de los mismos durante los últimos doce meses ?

Modelo físico

Un modelo físico está basado en el modelo de datos lógico. El modelo lógico concierne a objetos lógicos del modelo de negocios tales como día, ítem, sucursal o cuenta.

Mientras que el modelo lógico de datos dice qué facts y atributos crear, el modelo físico dice dónde van a ubicarse físicamente los datos de esos objetos. El esquema físico describe cómo los datos se almacenarán en el datawarehouse.

Dos componentes claves conforman el esquema físico de un datawarehouse: tablas y columnas. Las columnas y tablas de un datawarehouse físico representan facts y atributos de un modelo de datos lógico. Las filas de una tabla representan los elementos de los atributos y los datos fact propiamente dichos.

Columnas

Las columnas son campos en el datawarehouse que mantienen atributos y facts. Existen tres tipos de columnas:

1. **Columnas ID**: contienen códigos de identificación de elementos. Estos códigos son típicamente numéricos porque las computadoras pueden procesar los números mucho más rápidamente que el texto. Todos los atributos deben tener una columna ID.

2. **Columnas descriptivas**: contienen descripciones de texto de atributos. Una columna ID puede servir como un ID y una descripción a la vez. La mayoría de los atributos generalmente tienen una columna ID y al menos una columna descripción.

3. **Columnas fact**: contienen datos de hechos o facts.

Tablas

Existen tres tipos de tablas:

- Tablas de dimensión[9](dimension tables)
- Tablas de relación
- Tablas de facts

Tablas de dimensión o lookup

Son representaciones físicas de los atributos. Proveen la posibilidad de agregar datos a diferentes niveles. Funcionalmente, las tablas de dimensión proporcionan la información de un atributo a través de los datos almacenados en sus columnas ID y descripción.

Dependiendo en cómo se elija organizar el esquema físico, una tabla de dimensión puede almacenar información de uno o más atributos relacionados. Si una tabla solo almacena datos de un atributo se dice que es normalizada. Si una tabla tiene datos de múltiples atributos, se dice que es una tabla desnormalizada.

Tablas de relación

Mientas las tablas de dimensión almacenan información sobre uno o más atributos, las tablas de relación almacenan información sobre la relación entre dos atributos. Las tablas de relación contienen columnas de ID de dos o más atributos definiendo asociaciones entre ellos.

[9] También denominadas “lookup”.

Tablas Fact

Las tablas fact se usan para almacenar métricas. Dado que los atributos son los que dan significado a los valores de fact, las tablas fact incluyen los Ids de las columnas de atributos y las métricas. Las columnas ID de atributos incluídas en una tabla fact representan el nivel en el cual las facts en las tablas están almacenadas.

Por ejemplo, tablas fact que contienen datos de ventas e inventario podrían representarse mediante el siguiente diagrama:

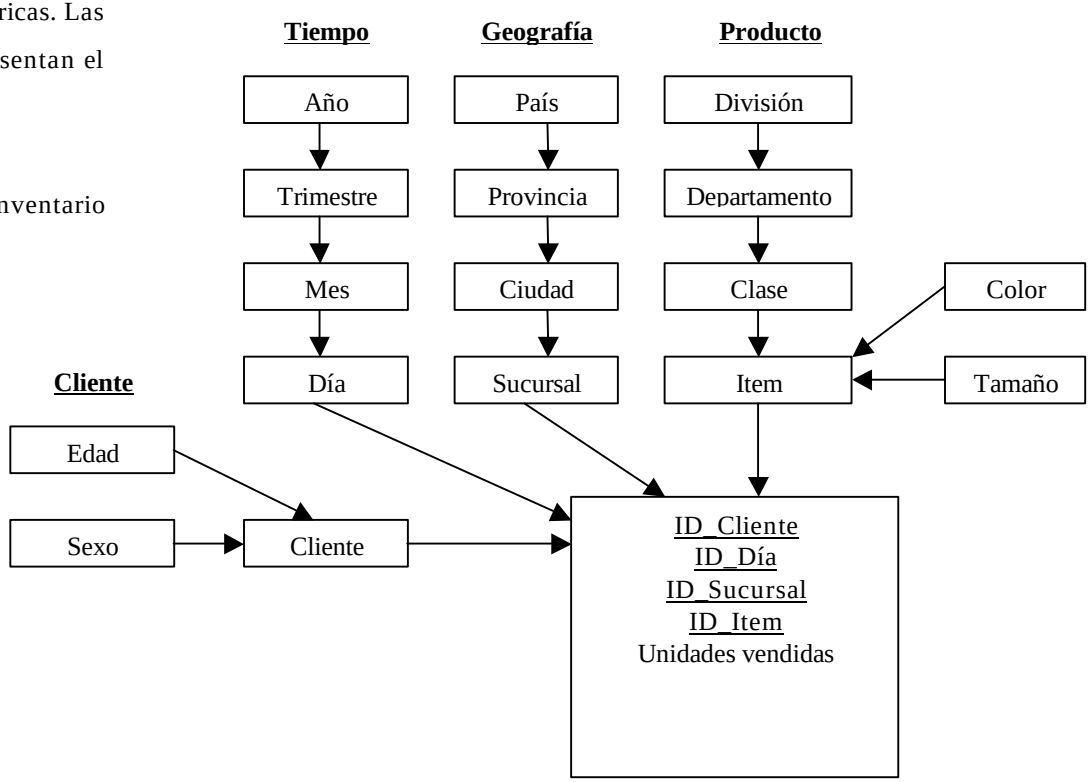


Gráfico 8 – El modelo anterior indicando la estructura de la fact table

Algebraicamente, podemos ver claramente mediante la estructura de la tabla fact que:
Unidades Vendidas = f (Cliente, Día, Sucursal, Item)

Diferentes tipos de esquemas

Existen muchas formas de estructura un datawarehouse y ningún método es por sí mismo correcto o equivocado. Cómo se elige la estructura del datawarehouse depende de la naturaleza de los datos, el espacio de almacenamiento disponible, y los requerimientos de los usuario. Típicamente, uno de los tipos de esquema o la combinación de ellos es utilizada para organizar el esquema físico para optimizar la performanse de las consultas. Estos tipos de esquema son:

- Altamente normalizado
- Moderadamente normalizado
- Altamente desnormalizado

En cada uno de estos esquemas existe básicamente una tabla fact base y cualquier número de tablas facts agregadas.

Esquema altamente normalizado

Tomemos como ejemplo la jerarquía tiempo. La implementación física de las tablas utilizando un esquema altamente normalizado podría ser:

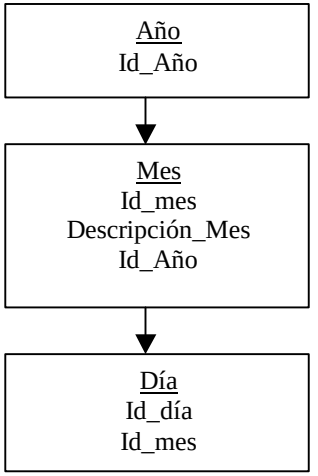


Gráfico 9 – Jerarquía Tiempo altamente normalizada

Esquema moderadamente normalizado

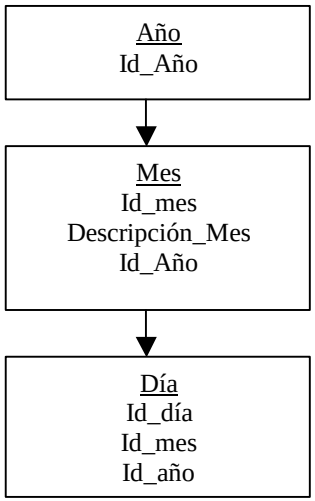


Gráfico 10 - Jerarquía Tiempo moderadamente normalizada

Esquema altamente desnormalizado

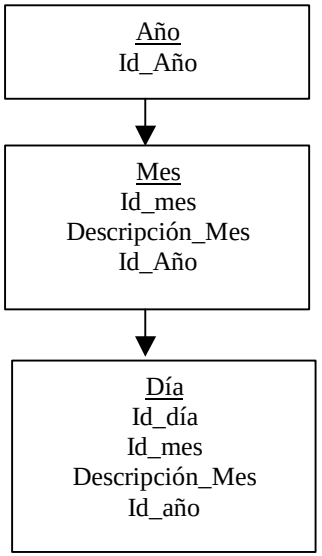


Gráfico 11 Jerarquía Tiempo altamente desnormalizada

Comparación entre los diferentes tipos de esquemas:

	Estructura de tabla de dimensión		
Esquema normalizado	Contiene un ID de atributo y columna de descripción así como la columna ID del padre.	Se minimiza el espacio de almacenamiento y se minimiza la redundancia de datos.	Requiere numerosos joins para obtener la información de las tablas de dimensión de más alto nivel.
Esquema moderadamente normalizado	Contiene un ID de atributo y columnas de descripción así como la columna ID de todos sus ancestros.	Reduce significativamente la cantidad de joins necesarios para relacionar los atributos dentro de la jerarquía.	Requiere algo de almacenamiento redundante.
Estructura altamente desnormalizada	Contiene el ID de un atributo y columnas de descripción así como las columnas de ID y descripción de todos sus ancestros	Elimina aún más los joins necesarios para recuperar las descripciones de los atributos.	Requiere más cantidad de espacio de almacenamiento.

Sumarización y agregación de una tabla fact

Suponiendo que se quiere modelar el datawarehouse de ventas de un supermercado y que este supermercado tiene 30 sucursales, con 50 puestos de caja promedio, cada puesto de caja procesa una venta cada 5 minutos promedio y cada venta es en promedio de 10 artículos. Cada sucursal abre 12 horas al día, incluyendo Sábados y Domingos.

Usando una estructura de fact table como la siguiente:

Código de Día
Código de Sucursal
Código de Caja
Código de Venta
Código de Producto
Precio venta
Unidades Vendidas

¿ Cuantos registros promedio / aproximadamente tendría esta tabla fact a lo largo de 30 días ?

Rta: Cantidad de ventas al día: 12 horas * 60 minutos / 5 minutos = 144 ventas

La cantidad de registros sería: 10 productos * 144 ventas diarias * 50 puestos de caja * 30 sucursales * 30 días = 64.800.000

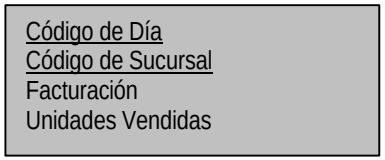
En un año, nuestro datawarehouse tendría 777.600.000 de registros !!!

¿ Qué pasaría si un usuario decide ver un reporte que le muestre la facturación total anual del supermercado en los últimos 5 años ? La consultaría se realizaría ejecutando una consulta sobre 3.888.000.000 de registros.

Lo que se debe hacer en este caso es agregar o sumarizar los datos a guardar en el mismo.

Teniendo en cuenta que estás son aplicaciones orientadas a la toma de decisiones (para nivel gerencial, directores, etc), lo más probable es que a un director o gerente no le interese saber que el día 3 de Abril, en la Sucursal 24, en la Caja 35, en la Venta con el ticket nro. 384.632 / 3 se vendieron 2 paquetes de azucar a 1.25 $ cada uno.

Quizás ese usuario se formule preguntas del tipo: ¿ cuál fue el día de mayor facturación del mes por sucursal ? A modo de ejemplo, una estructura de este tipo respondería esa pregunta:



Esta estructura contendría solamente 900 registros (30 días * 30 sucursales).
En 5 años contendría 900 regisros * 12 meses * 5 años = 54.000 registros.

Como conclusión, el nivel de detalle o de agregación de los registros debe estar en función del tipo de consultas que los usuarios del datawarehouse quieren responder. La agregación o sumarización de datos a este nivel son realizados por los procesos de ETL (extract, transform, load – extracción, transformación y carga) mencionados anteriormente.

Ejemplo de modelo físico de un DW

Supongamos que una empresa cuenta con información de sus ventas (cantidad de unidades, facturación y margen de cada una de las ventas realizadas). Y dicha empresa necesita mediante una herramienta emitir reportes por rango de fechas o tiempo (en sus diferentes niveles), producto y cuenta del comprador.

Un modelo de este tipo lo podemos representar de la siguiente manera:

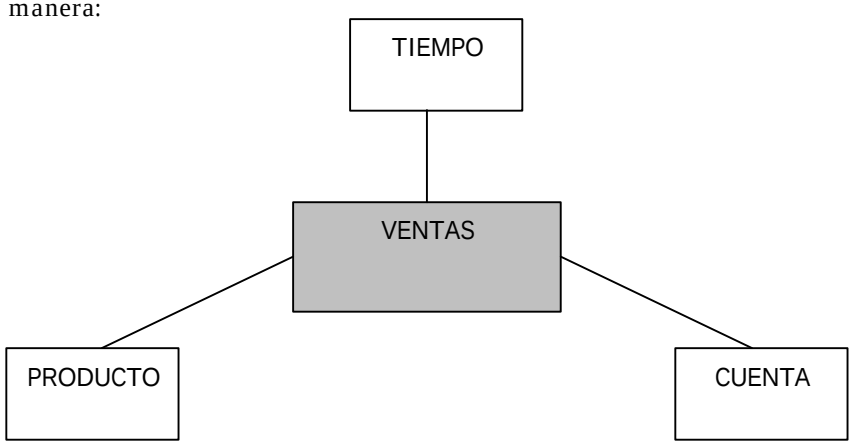


Gráfico 12 - Un ejemplo simple de modelo físico de un datawarehouse

Dimensiones

Como ya dijimos, las dimensiones son calificadores que le dan un significado a las medidas. Las dimensiones organizan los datos basados en el qué, cómo y cuando.

En este caso las dimensiones son Tiempo, Cuenta y Producto.

Ejemplos de preguntas que las dimensiones permitirían contestar con las dimensiones del ejemplo anterior serían:

- ¿ Qué cuentas tuvieron la facturación más alta el último año ?
- ¿ Cuál fue la ganancia por producto ?
- ¿ Cuántas unidades fueron vendidas para cada marca durante los últimos doce meses?

Ahora bien, tomemos la dimensión cuenta.

Supongamos que nuestra empresa tiene la información de las cuentas en una aplicación operacional (podría ser en un SAP, Peoplesoft, etc) en tablas con un grado alto de normalización:

Entidad Cuenta
Código de cuenta
Nombre de la cuenta
Código de Territorio

Entidad Territorio
Código de Territorio
Nombre del Vendedor
Código de Región

Entidad Región
Código de Región
Nombre de la Región
Gerente de la Región

Si bien esta estructura de datos es la adecuada para este tipo de ambiente operacional , ya que disminuye los datos duplicados y optimiza el uso de índices para búsquedas entre otras cosas, en un datawarehouse podría verse la siguiente estructura:

Dimensión Cuenta
Código de Cuenta
Nombre de la Cuenta
Código de Territorio
Nombre del Vendedor
Código de Región
Nombre de la Región
Gerente de la Región

El siguiente es un ejemplo de como quedaría esta estructura no normalizada con sus datos:

Código de cuenta	Nombre de la cuenta	Territorio	Vendedor	Región	Nombre de la región	Gerente de la Región
1	Siderca	100	M. Pérez	SE	Sudeste	P. Alba
2	Aluar	101	J. Sánchez	SE	Sudeste	P. Alba
3	Alpargatas	200	J. Sánchez	N	Norte	C. Carra
4	Ford	202	L. Martínez	N	Norte	C. Carra

Como ya afirmamos, en un esquema de datawarehousing, datos desnormalizados reducen la cantidad de joins necesarios para un query, y hace más fácil a los usuarios comenzar a realizar consultas a un nivel más alto y luego ir bajando hacia niveles de mayor detalle (esto por lo general se denomina drill down).

Cuanto mayor sea el nivel de detalle en la estructura de datos, mayor será el nivel de detalle con el cual el usuario podrá ver los datos.

Facts

Como ya dijimos, la tabla fact es el centro del modelo dimensional. Contiene una lista de todas las medidas y apunta a cada una de las claves del nivel más detallado de cada dimensión.

La granularidad de la tabla fact está determinada por el nivel más detallado de cada dimensión.

La fact de nuestro ejemplo se vería de la siguiente manera:

Código de Producto	Código de Cuenta	Código de Día	Unidades Vendidas	Facturación	Margen
1	2	32104	3	82.12	26.12
3	4	33111	2	171.15	98.56
1	3	32567	5	23.45	13.15

Podemos representar nuestro datawarehouse, ya en un modelo físico de la siguiente manera:

Gráfico 13 – Ejemplo de modelo físico de un datawarehouse

La tabla fact (en este caso Ventas) contiene una clave primaria compuesta por las claves primarias de las dimensiones. Cada atributo de la clave primaria de la tabla fact es a su vez clave foránea a alguna tabla de dimensión.

El tipo de esquema representado arriba se suele denominar “estrella” o “star schema”. Existe otro tipo de modelo denominado snowflake. Estos modelos se discuten en el siguiente apartado del presente texto.

Esquemas de modelado de datos

Star schema

Un star schema contiene una sola fact table y una sola tabla para cada dimensión. Cada fact, apunta a una tupla en cada una de las dimensiones.

El star schema no captura las jerarquías directamente. La performance se mejora reduciendo el número de tablas implicadas en la consulta. Este diseño simple hace a los datos menos complicados. Todas las tablas de dimensión hacen join a la tabla fact en una forma claramente definida. Cada tabla de dimensión joinea a la tabla fact por la misma columna cada vez, más allá del query.

Usando el modelo de la página anterior se podría determinar la facturación por producto para cada cuenta para cada día informando además el vendedor y su gerente de la siguiente manera:

```
SELECT codigo_producto, nombre_producto, codigo_cuenta,
fecha_orden, codigo_vendedor,    codigo_gerente
        FROM Ventas, Productos, Tiempo, Cuentas
        WHERE Ventas.codigo_prod = Productos.codigo_prod
AND
                Ventas.codigo_cuenta =
Cuentas.codigo_cuenta AND
                Ventas.codigo_dia = Tiempo.CodigoDia
```

Si se hubiese usado un grado de normalización mayor en la dimensión Cuentas, dividiendo a la Cuenta en 3 tablas (Cuenta, Territorio, Región) tal como se ejemplificó anteriormente, para obtener el mismo resultado hubiese sido necesario realizar la siguiente consulta:

```
SELECT codigo_producto, nombre_producto, codigo_cuenta,
fecha_orden, codigo_vendedor, codigo_gerente
        FROM Ventas, Productos, Tiempo, Cuentas, Territorio,
Region
        WHERE Ventas.codigo_prod = Productos.codigo_prod
AND
                Ventas.codigo_cuenta =
Cuentas.codigo_cuenta AND
                Cuentas.codigo_territorio =
                Territorio.codigo_territorio AND
                Territorio.codigo_region =
                Region.codigo_region
                Ventas.codigo_dia = Tiempo.CodigoDia
```

Imaginemos que nuestro datawarehouse tiene un registro por cada producto que se vendió en una venta determinada. Si vendemos 1 producto por día el impacto en la performace del query no sería importante. Pero si tenemos un alto volúmen de ventas, seguramente el agregar joins a la consulta impactará en la performance de la misma.

Facts Constellation (constelación de facts)

Podemos ver a una constelación de facts como múltiples star schemas (o múltiples estrellas) anidadas. En este caso, existen muchas tablas fact que comparten muchas tablas de dimensión. Para lograr esto, es clave tener las dimensiones representadas de manera uniforme.

Gráficamente, podemos representar una constelación de facts de la siguiente manera:

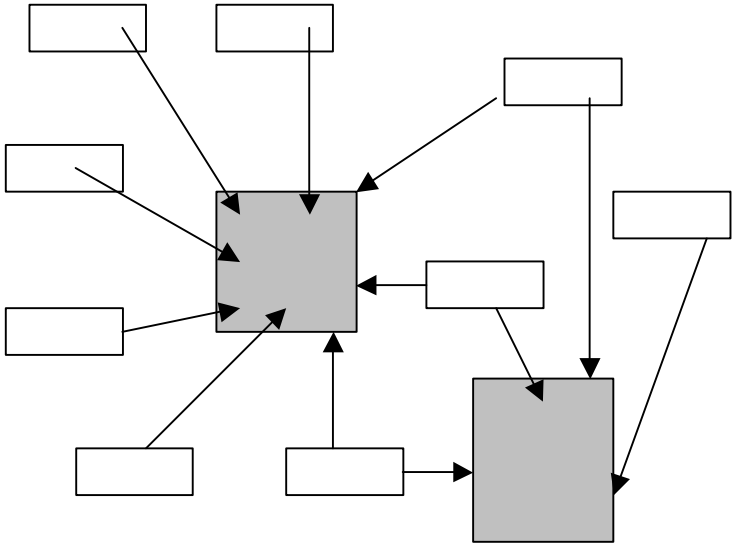


Gráfico 14 – Constelación de estrellas

Snowflake schema

Un snowflake schema representa la jerarquía de dimensiones directamente normalizando las tablas de dimensión.Un snowflake esquema es más fácil de mantener y reduce el espacio de almacenamiento a través de la normalización de sus tablas.

Algunos autores (por ej. Kimball) alegan que un schema snowflake reduce la efectividad del browseado de datos.

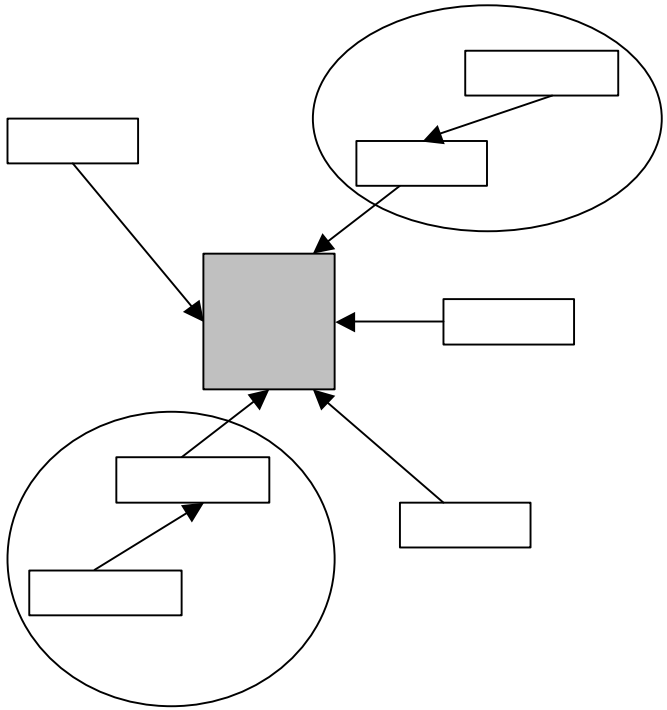


Gráfico 15 – Snowflake schema

Implementación física del Datawarehouse

Este capítulo trata sobre el soporte físico de la información del datawarehouse. Básicamente, nos referimos a la arquitectura que deben tener los servidores, y a la base de datos como estructura de almacenamiento.

Arquitectura de los servidores

La arquitectura de los servidores se determina en función del tamaño de la implementación (por ej. volumen de datos) sin dejar de considerando aspectos como la escalabilidad, la disponibilidad y el tiempo de respuesta.

Servidores de un solo procesador

Los servidores de un solo procesador presentan ventajas en cuanto a su administración, ya que administrar un solo procesador es más simple que hacerlo sobre un conjunto de los mismos. Pero el hecho de contar con un solo procesador tiene la desventaja fundamental que este se transforma en SPOF (single point of failure). Es decir, un problema en el único procesador impide que el sistema de datawarehousing continúe funcionando. Por otro lado, contar con un único procesador limita la capacidad de procesamiento y la escalabilidad (es decir, la posibilidad de que el datawarehouse crezca)..

SMP (Symmetric Multiprocessing - multiprocesamiento simétrico)

En este tipo de arquitectura, varios procesadores comparten memoria y los dispositivos de almacenamiento de disco. Esto incrementa la escalabilidad, ya que al detectar problemas de capacidad de procesamiento se pueden adicionar nuevos procesadores. Una empresa puede comenzar a utilizar su datawarehouse con una configuración mínima de 2 procesadores y luego escalar cuando sea necesario.

MPP (Massively Parallel Processing – Procesamiento masivo en paralelo)

En estos equipos, se conecta varios procesadores por medio de enlaces de banda ancha y alta velocidad. Podemos afirmar que MPP es un conjunto de servidores propios, generalmente SMP, con su propia memoria interna y dispositivos de almacenamiento, conectados por enlaces de alta velocidad. Es decir, cada nodo de una arquitectura MPP es un servidor completo.

Para poder utilizar esta arquitectura correctamente, las aplicaciones deben tener la posibilidad de ejecutarse en paralelo en partes separadas.

Este esquema es ideal para buscar información en grandes bases de datos. El motor de base de datos debe soportar procesamiento en paralelo (los vendedores más importantes de bases de datos generalmente cuentan con una versión para MPP). Además de esto, es importante el trabajo que se realiza en el tuning del motor de base de datos. Por lo general, las bases de datos distribuyen los mismos en distintos equipos (fragmentación).. El particionamiento que se aplique debe ser el adecuado para evitar que la carga sea despreja entre los procesadores.

NUMA (Non uniform memory access -Acceso de memoria no uniforme)

La arquitectura NUMA conecta múltiples nodos SMP en una sola gran máquina SMP mediante un banco de memoria único lógicamente (aunque distribuido físicamente) y un único sistema operativo.

NUMA combina los beneficios de administración de los equipos SMP con la capacidad de performance de las máquinas MPP.

Existen motores de bases de datos y aplicaciones que pueden moverse sin modificarldes de ambiente SMP a NUMA.

Bases de datos

Los motores de bases de datos relacionales estuvieron originalmente orientados a las estructuras de datos normalizadas. Pero los análisis de los datos de los datawarehouse (OLAP) requiere realizar varias operaciones de join sobre gran cantidad de registros. Un motor de base de datos relacional es más óptimo para una consulta del tipo "Traer el cliente con número de CUIT 20-24834739-3" que para consultas del tipo "Traer todos los clientes que tengan entre 30 y 40 años, que hayan comprado un inmueble mayor a los 100.000 $ en el último año y que tengan tarjeta de crédito VISA".

Lo que hicieron las empresas vendedoras de tecnología fueron añadir características al motor de base de datos original pensando en su uso OLAP. Algunos de estos motores de bases de datos soportan esquemas de indexados particulares.

Muchas de las herramientas de acceso a los datawarehouse explotan la naturaleza multidimensional del mismo. Algunos productos de bases de datos, tales como Essbase, implementan técnicas de almacenamiento y operadores que soportan estructuras de datos multidimensionales.

Elementos a considerar en bases de datos

Tablespaces

Las bases de datos generalmente manejan el concepto de tablespace. Una base de datos está integrada en varios tablespaces. Cada tablespace recibe un tamaño de espacio en disco. A cada objeto (por ejemplo, tablas o índices) que requiere espacio de almacenamiento se le asigna un tablespace. El tablespace es la conección entre las estructuras de la base de datos y el soporte físico de las mismas.

El primer objetivo de diseño de los tablespaces es separar los objetos que tienen diferentes requerimientos o funciones. Las tablas deben estar separadas de sus índices. Las tablas pequeñas deben separarse de las grandes. Dos tablas que generalmente tienen un join entre ellas deben estar en distintos tablespaces. Y en lo que a datawarehousing respecta, las tablas de dimensión deben estar separadas de las tablas fact. Por otro lado, tablas e índices que son estáticos no deben estar en el mismo tablespace que tablas dinámicas. Y tablas que son frecuentemente accedidas para consultas deben estar separadas de tablas que se acceden en forma aleatoria.

El segundo objetivo de diseño tienen que ver con el mantenimiento. En algunos motores de bases de datos (por ejemplo, Oracle), el tablespace es la unidad de backup y recuperación de datos. Entonces se debe planear los requerimientos de carga y de backup para determinar la estrategia de particionamiento.

Indices

Para entontrar los datos de interés en un query particular, las bases de datos tienen siempre la opción de leer todos los registros de una tabla. Para tablas pequeñas es a menudo el método de acceso preferido. En muchos casos, un índice es necesario para permitir a Oracle encontrar una fila o un conjunto pequeño de las mismas en una tabla. En otros casos, índices múltiples en varias columnas de una tabla pueden usarse en conjunto para determinar las filas que un query va a devolver.

Qué columnas de las tablas es necesario indexar se determinar al conocer los datos y los tipos de query que se van a realizar. La cláusula WHERE (y en algunos casos la cláusula ORDER BY) son las que determinan los índices que va a usar el query.

Monitoreando el uso del datawarehouse y los queries solicitados por los usuarios es posible identificar la necesidad de índices adicionales que pudieron no haber sido anticipados cuando el datawarehouse fue inicialmente implementado. Indices innecesarios, además de ser

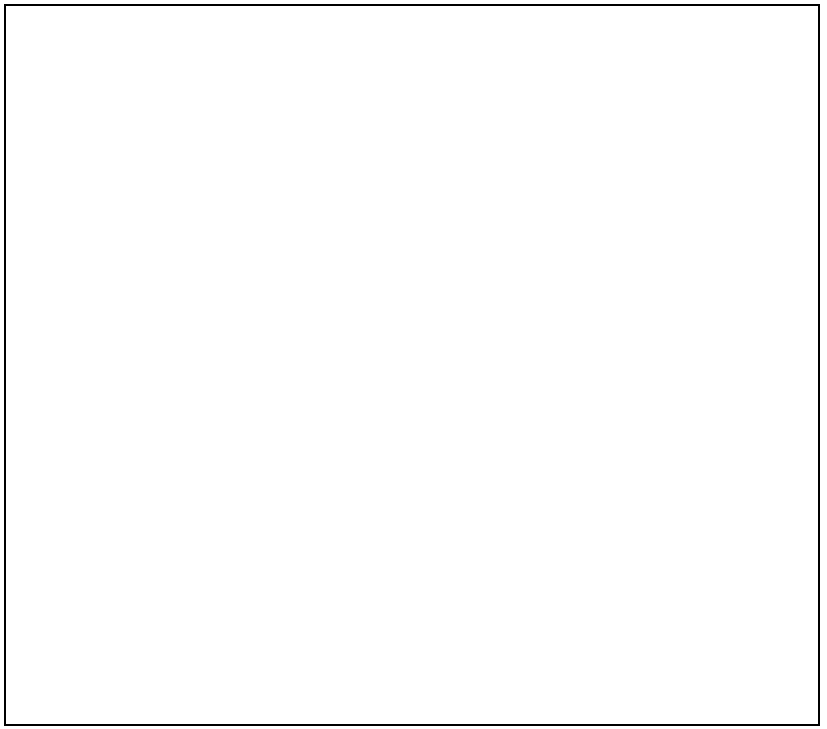
un desperdicio de espacio, generalmente no mejoran la performance de los querys.

Indices B-Tree

Los índices B-Tree son estructuras jerárquicas que permiten una búsqueda rápida para obtener la dirección (en algunas implementaciones de bases de datos llamadas ROWID) de una fila específica con un determinado valor en una tabla.

Un índice B-Tree está basado en un arbol balanceado que se procesa jerárquicamente para encontrar el valor de interés.

Si un usuario quiere encontrar información acerca del número de producto 21694, el motor de base de datos lee la raíz del árbol para determinar qué bloques de segundo nivel leer a continuación. Leyendo el bloque apropiado del segundo nivel se determina qué bloques del tercer nivel leer. Esto es así hasta llegar a los nodos hoja, que contienen los ROWIDs (direcciones) del bloque real de datos que contiene el dato que se quiere ver. En el ejemplo del siguiente gráfico, hacen falta cuatro lecturas lecturas de bloque lógico (3 en el índice y una en los datos) para llegar al dato propiamente dicho.



[10]

Gráfico 16 – B-Tree index

Por lo general, existen dos variedades de índices B-Tree. Un índice unique permite a lo sumo que una fila tenga un valor particular. El índice non-unique permite que varias filas contengan un mismo valor.

[10] Extractado de Essential Oracle 8i Datawarehousing – Gary Godge, Tim Gorman.

En un unique index, solo una fila apunta a cada valor del índice. Un nonunique index, puede tener diferentes punteros a diferentes filas de cada valor distinto. Como el número de filas direccionado por clave crece, la eficiencia de usar el índice cae. Cada valor de un índice B-TREE de baja cardinalidad, tales como el CODIGO_SEXO, que tiene dos valores (al menos en la mayoría de los diseños !), generalmente apuntaría a filas en el mismo bloque de la tabla. Resumiento, en este caso, hacer un scan de la tabla entera sería más eficiente que interactuar desde el índice hacia la tabla y viceversa.11 Esto es importante es estructuras de datos como las de un datawarehouse que no siempren se encuentran normalizadas.

Indices Bitmap

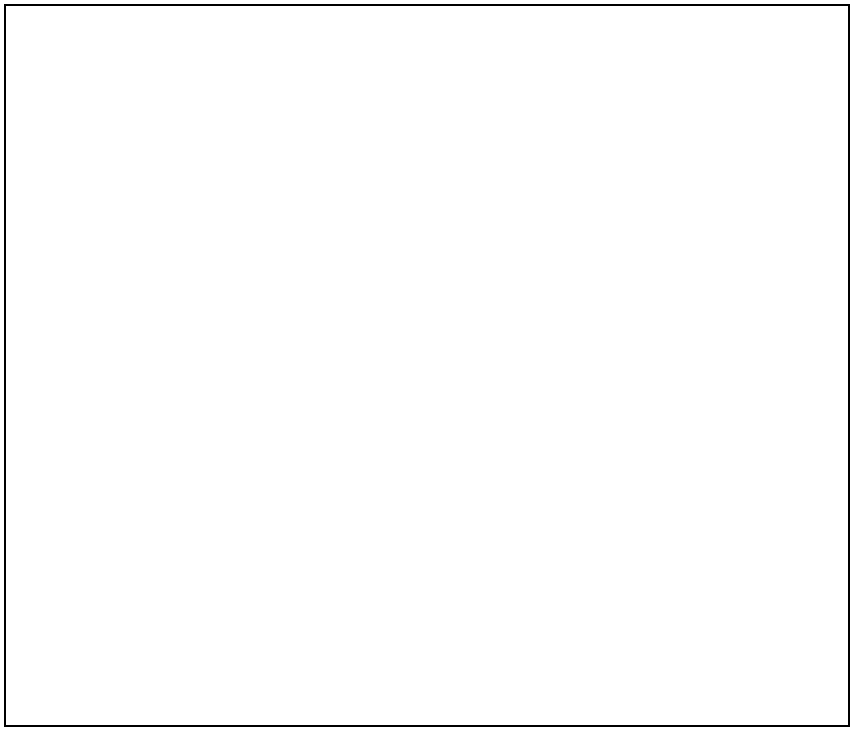
Los índices Bitmaps son apropiados en el caso en que solo un par de valores distintos ocurren en una columna (o serie de columnas). A estas columnas se las denomina "de baja cardinalidad". Un caso de columna de baja cardinalidad es el campo CODIGO_SEXO mencionado anteriormente. Los índices bitmap en columnas de baja cardinalidad son muy eficientes en cuanto al almacenamiento físico. Un bitmap separado se construye para cada valor distinto. Lógicamente, cada bitmap es tan grande como la cantidad de filas siendo indexadas.

Físicamente, un índice bitmap es idéntico en la estructura de jerarquía a un índice B-Tree, excepto en la forma que adoptan los nodos hoja. Mientras un índice B-Tree en sus hojas contiene un puntero a un ROWID, el índice bitmap contiene un conjunto de bits que incluye un bit correspondiente a cada fila en un rango continuo de fila de datos de una tabla. Cada fila será representada por un 1 si esa fila contiene un determinado valor, y un cero si no. Ciertos motores de bases de datos realizan compresión de los bitmaps de forma tal que una larga serie de ceros se puedan almacenar en menor espacio del originalmente requerido.

La columna SEX de la tabla EMPLOYEES del gráfico 17 es un excelente ejemplo de una columna candidata para un índice bitmap. Hay solo dos valores posibles dentro de la tabla, por lo que se necesitan crear sólo dos cadenas de bitmaps. Si se hubiesen construído índices bitmap en otras columnas de la misma tabla (tal como STATUS, que contiene otros 2 valores, DEPT y JOB_TITLE, que tienen un set de valores pequeños), entonces el motor de base de datos puede eficientemente combinar la información de estos cuatro índices bitmap para responder preguntas del tipo "¿ Cuantos

[11] Cuando la distribución de datos es despareja, un índice B-TREE en datos de baja cardinalidad podría ser adecuado. Es el caso de buscar aquellas personas cuyo CODIGO_SEXO = 'F' (femeninas) en la base de datos de quienes realizaron el servicio militar, asumiendo que son un pequeño porcentaje del total de los datos. Por otro lado, usar el mismo índice para encontrar todos los de sexo masculino sería totalmente ineficiente.

oficinistas part-time de sexo masculino están empleados en el departamento de marketing ? Los índices bitmap son entonces muy utilizados en datawarehousing, Queries de este tipo, los cuales buscan la intersección altamente selectiva de varios criterios de búsqueda que son individualmente poco selectivos (por la baja cardinalidad), son muy usados en sistemas de soporte a las decisiones (como un datawarehouse).



[12]

Gráfico 17 – Bitmap Index

Indices basados en funciones

Usar índices que encuentren filas que contienen un determinado valor (o rango de valores) dentro de un conjunto de columnas ha sido una técnica universal de los motores de bases ded datos durante los últimos 30 años. Ahora, ¿ qué pasa cuando el valor no está en la base de datos ? ¿ Qué pasa si se necesita encontrar todos los contratos de consultoría en los cuales los gastos hayan excedido en un 25% a lo facturado ? Supongamos que la tabla PROJECTS contiene una sumarización de los gastos y otra columna que muestra la facturación total. Pero esta tabla no tiene otra columna que almacene la facturación en función de los gastos (facturación dividido gastos). Una alternativa sería hacer un full-scan de la tabla y encontrar los proyectos buscados Esto podría hacerse para un query de una única vez. ¿ Pero qué pasarí si se quiere hacer este query frecuentemente ? Una alternativa sería agregar una nueva columna a la tabla PROJECTS que contenga el valor calculado de gastos / revenue. Esto funcionaría bien si nuevos registros se insertan en forma no frecuente. Sino, de otra manera sería necesario estar

[12] Extractado de Essential Oracle 8i Datawarehousing – Gary Godge, Tim Gorman.

permanentemente reconstruyendo y reorganizando las tablas para calcular este valor.

Algunos motores de bases de datos proporcionan otra alternativa. Si se determina la necesidad de buscar sobre filas donde alguna columna contiene un valor basado en una función de otras columnas, y no se quiere físicamente almacenar el valor calculado en la tabla, la alternativa es crear el índice basado en la función o expresión desaeada. Cada vez que una consulta utilice esta función o expresión en esta cláusula, se puede usar el índice basado en función para recuperar las filas más eficientemente.

Particionamiento de tablas

Las tablas de un datawarehouse por lo general contienen millones de registros de filas y pueden requerir muchos gigabytes de almacenamiento. Administrar segmentos de datos de este tamaño puede ser un problema para un DBA. Para solucionar esto, en muchos motores de bases de datos, una tabla puede particionarse. Cada partición puede contener un subconjunto del total de filas de la tabla. Los valores de ciertas columnas se pueden utilizar para hacer el particionamiento, aunque es común en muchas implementaciones de datawarehousing ver un particionamiento por un campo “fecha”. Esto permite que cada partición se cargue con datos de un período de tiempo (por ejemplo, un mes) y luego su estado se pase a read-only. Cuando los datos de una partición son demasiado viejos y no se utilizan en el análisis, la partición entera puede ser archivada o purgada fácilmente. Por otro lado, cada partición puede ser administrada en forma independiente, lo cual hace más simple a la administración total. Aunque particionar una tabla también resulta en mejoras de performance, las ventajas primarias de hacerlo radican en la facilidad de administración. El particionamiento de tablas es una necesidad en prácticamente todos los proyectos serios de datawarehousing.

ROLAP vs. MOLAP vs. HOLAP vs. DOLAP

Hoy por hoy, existen en el mercado diferentes filosofías de almacenamiento de los datos en una base de datos. Cada una de ellas tiene sus características particulares y no podemos afirmar que una sea mejor que la otra, sino en función de las necesidades de cada organización. Presentamos a continuación algunas de las ventajas y desventajas más reconocidas de estos esquemas.

MOLAP

Por un lado, está el OLAP ´original´ multidimensional (MOLAP) . En este esquema, los datos son precalculados y almacenados multidimensionalmente en cubos de datos. Es decir, los datos se toman del datawarehouse, se precalculan y se almacenan en cubos. El resultado es un tiempo de respuesta más rápido ya que los datos se encuentran precalculados y no necesitan ser recalculados ante cada requerimiento. Lo negativo de este esquema es que almacenar datos multidimensionalmente requiere mucho más espacio. Además, el esquema MOLAP suele requerir que sus bases de datos se precompilen para conseguir un rendimiento aceptable en las consultas, incrementando, por lo tanto, los requerimientos batch. Otra de las desventajas del esquema MOLAP es que es poco flexible dado que las dimensiones y agregaciones ya se encuentran predefinidas. Si un analista de negocios desease examinar una dimensión que no está definida en el modelo físico de la base de datos, un desarrollador necesitaría definir la dimensión en la misma y modificar las rutinas usadas para ubicar y reformatear los datos fuente, y luego un operador debería cargar los datos de la dimensión.

Otra consideración importante de este esquema es que los datos en la base de datos debe ser periódicamente actualizada. Este proceso de actualización debe ser scheduledo y administrado. Además, cada actualización necesita de un proceso de validación de los datos para asegurar la consistencia de los mismos. Finalmente, un operador debe alocar tiempo para crear los índices y agregaciones, tarea que puede consumir demasiado tiempo una vez que los datos hayan sido cargados. Las empresas generalmente necesitan invertir recursos significativos en implementar sistemas de bases de datos multidimensionales y monitorear en el día a día su operatoria. Esta complejidad agrega retrasos y costos en las implementaciones, y requiere que el área de IT se involucre activamente. Esto resulta en que el analista, que es generalmente un usuario de negocios, tenga una gran dependencia del área de IT. De esta manera, uno de los grandes beneficios de la tecnología OLAP (la habilidad de analizar la información sin la participación de los profesionales de IT) , es significativamente afectada.

ROLAP

Las implementaciones ROLAP (OLAP relacional) son similares en funcionalidad a las MOLAP. Sin embargo, estos sistemas utilizan un motor de base de datos relacional en lugar de un motor de base de datos multidimensional especializado. Esto da mayor escalabilidad dado que permiten manejar más grandes volúmenes de datos que un esquema MOLAP. Además, las implementaciones ROLAP generalmente tienen un mejor esquema de drilling (análisis de datos a diferentes niveles),. Las estructuras de datos utilizadas requieren, al igual que en el esquema MOLAP, de un tiempo inicial de diseño e implementación bastante grande. Adicionalmente, dado que las estructuras de datos son relacionales, para acceder a los registros detallados se debe utilizar SQL. De esta manera, el motor ROLAP debe realizar trabajos adicionales para realizar comparaciones, tales como comparar el importe de ventas del mes actual con el del mismo mes del año pasado.

HOLAP

Algunas empresas de tecnología venden soluciones capaces de acceder bases de datos relacionales directamente desde un motor de base de datos multidimensional, apareciendo así el concepto de HOLAP (Hybrid OLAP). Estas implementan el concepto de drilling, el cual automáticamente genera SQL para recuperar registros detallados para mayor análisis. Esto le da al usuario final la percepción de que está pasando de la base de datos multidimensional a la relacional.

El esquema HOLAP combina la performance y funcionalidad de una base de datos multidimensional con la capacidad de visualizar datos más detallados (drilling). Esto le da gran valor a algunas tipos de usuarios. Sin embargo, estas implementaciones están generalmente soportadas por un solo vendor de bases de datos y son nuevamente complejas de implementar y mantener.

DOLAP (Desktop OLAP)

El Mercado OLAP desktop resultó de la necesidad de los usuarios de correr consultas de negocios usando sets de datos relativamente pequeños extraídos de los sistemas productivos. Muchos sistemas OLAP desktop fueron desarrollados como extensiones de sistemas de reportes de producción, mientras otros fueron desarrollados en la época temprana de los sistemas client/server para utilizar las ventajas de la naciente PC. Los sistemas OLAP de desktop son populares y típicamente requieren poca inversión en IT para ser implementados. También proveen operaciones OLAP altamente móviles para aquellos usuarios que trabajan remotamente o viajan en forma permanente. Sin embargo, muchos están limitados a un solo

usuario y les falta capacidad para manejar grandes volúmenes de datos.

Ejemplos de herramientas OLAP

Algunas de las herramientas MOLAP existentes en el mercado son Hyperion Essbase, Powerplay Enterprise Server, SAS CFO Vision Comshare Decisión., Microstrategy, Informix Metacube , Cartesis Carat,
Oracle Express, Seagate Holos, Microsoft OLAP Services, BRIO, Business Objects y Cognos Powerplay.[13]

[13] Los ejemplos de esta sección no pretenden ser completos. La clasificación de los productos pueden variar con las evoluciones tecnológicas