UNIVERSIDAD POLITÉCNICA DE VALENCIA

Departamento de Sistemas Informáticos y Computación

Tesis doctoral

Un Sistema para el Mantenimiento de Almacenes de Datos

Presentada por:

Clemente García Gerardo

Dirigida por:

Dra. Matilde Celma Giménez

Valencia, España, Junio de 2008

Abstract

Data warehouses are databases designed to help organizations in the decision-taking process. Data warehouses combine data from several other database and transaction processing systems to make use of the data as a whole. So that the data warehouse represent the organization's reality they need to be updated periodically. That updating process may required quite a few resources and sometimes shut down the data warehouse so that analyst can not access it. Analysts require that data warehouse be up all the time so that data warehouse maintenance process is a critical point of the system. For that reason research about efficient strategies to improve data warehouse maintenance has recived special attention since this technology appeared.

Data warehouse maintenance is a three-phase process: Draw out of data from their sources, data transformation, and data warehouse updating.

This research work has to do with the data transformation phase and mainly with the updating one. For the transforming phase a system has been developed to perform data moderate cleaning activities, data format integration and data semantic integration. As to the updating phase two algorithms were defined and implemented to perform data warehouse updating on both incremental and line approaches. The use of those algorithms does not require that the data warehouse be down during the maintenance phase.

The algorithms used in the data warehouse updating phase are based on a multi-version strategy that let having an unlimited number of the updated data so that users may access the same data version while the warehouse is being updated. Those algorithms improve other existing literature algorithms and let data warehouse maintenance in either real time or batch.

Benefits of this research are quite significant due to it is improving the use data warehouse technology which is under huge growing fashion in all corporate processes mainly in the supply chain management process.

Resumen

Un almacén de datos es una base de datos diseñada para dar soporte al proceso de toma de decisiones en una organización. Un sistema de almacén de datos integra en un único repositorio, información histórica procedente de distintas fuentes de datos operacionales de la organización o externas a ella. Para que el almacén de datos sea en todo momento un reflejo fiel de la organización a la que sirve, debe ser actualizado periódicamente. Este proceso puede consumir muchos recursos, y en algunos casos inhabilitar el almacén de datos para los usuarios. En organizaciones donde el sistema debe estar disponible para los analistas en todo momento, el mantenimiento del almacén se convierte en un punto crítico del sistema. Por este motivo la investigación en estrategias eficientes de mantenimiento de almacenes de datos ha recibido la atención de los investigadores desde la aparición de esta tecnología.

El mantenimiento de un almacén de datos se realiza en tres fases: extracción de datos de las fuentes, transformación de los datos y actualización del almacén.
En este trabajo de tesis se han abordado, las fases de transformación y principalmente la fase de actualización. Para la fase de transformación se ha desarrollado un sistema que permite realizar tareas de limpieza moderada de los datos, integración de formato e integración semántica.

Pero, el trabajo principal se ha centrado en la fase de actualización, para ella se han definido e implementado dos algoritmos que permiten

realizar la actualización del almacén de datos de forma incremental y en línea, es decir evitando inhabilitar el almacén de datos durante el mantenimiento. Los algoritmos se basan en una estrategia multiversión, que permite mantener un número ilimitado de versiones de los datos actualizados, permitiendo de esta manera que los usuarios accedan a una misma versión del almacén mientras éste se está actualizando. Estos algoritmos mejoran otras propuestas existentes en la literatura, y permiten el mantenimiento en batch o en tiempo real, según las necesidades de la organización.

Los beneficios de esta investigación son evidentes, ya que la tecnología de almacenes de datos se extiende cada vez más en el mundo empresarial, siendo cada vez más los sectores comerciales, y las áreas geográficas donde se implantan este tipo de sistemas.

Resum

Un magatzem de dades és una base de dades dissenyada per a donar suport al procés de presa de decisions en una organització. Un sistema de magatzem de dades integra en un únic repositorio, informació històrica procedent de diferents fonts de dades operacionals de l'organització o externes a ella. Perquè el magatzem de dades sigui en tot moment un reflex fidel de l'organització a la qual serveix, ha de ser actualitzat periòdicament. Aquest procés pot consumir molts recursos, i en alguns casos inhabilitar el magatzem de dades per als usuaris. En organitzacions on el sistema ha d'estar disponible per als analistes en tot moment, el manteniment del magatzem es converteix en un punt crític del sistema. Per aquest motiu la investigació en estratègies eficients de manteniment de magatzems de dades ha rebut l'atenció dels investigadors des de l'aparició d'aquesta tecnologia.

El manteniment d'un magatzem de dades es realitza en tres fases: extracció de dades de les fonts, transformació de les dades i actualització del magatzem.

En aquest treball de tesi s'han abordat, les fases de transformació i principalment la fase d'actualització. Per a la fase de transformació s'ha desenvolupat un sistema que permet realitzar tasques de neteja moderada de les dades, integració de format i integració semàntica.

Però, el treball principal s'ha centrat en la fase d'actualització, per a ella s'han definit i implementat dos algorismes que permeten realitzar l'actualització del magatzem de dades de forma incremental i en línia,

és a dir evitant inhabilitar el magatzem de dades durant el manteniment. Els algorismes es basen en una estrategia multiversion, que permet mantenir un nombre il·limitat de versions de les dades actualitzades, permetent d'aquesta manera que els usuaris accedeixin a una mateixa versió del magatzem mentre aquest s'està actualitzant. Aquests algorismes milloren altres propostes existents en la literatura, i permeten el manteniment en batch o en temps real, segons les necessitats de l'organització.

Els beneficis d'aquesta investigació són evidents, ja que la tecnologia de magatzems de dades s'estén cada vegada més en el món empresarial, sent cada vegada més els sectors comercials, i les àrees geogràfiques on s'implanten aquest tipus de sistemes.

Esta tesis esta dedicada a mi esposa Martha Estela, a mis hijas Katia Daniela, Laura Rebeca y a mi hijo Raúl Clemente, quienes siempre tuvieron palabras o detalles para motivarme a terminar este trabajo.

Agradecimientos

Quiero expresar mi agradecimiento a la Dra. Matilde Celma Giménez, por haberme aceptado como su doctorante e introducirme al trabajo de investigación (etapa nueva para mí) y no podría dejar de agradecer la gran cantidad de horas dedicadas a la elaboración de esta tesis, así como todos los consejos que me dió, siempre los tendré presentes.

A los Drs. Paloma Martínez Fernández, Hendrik Decker, Francisco José García Peñalvo, Oscar Pastor López y Vicente Pelechano Ferragud, les agradezco el tiempo dedicado a la revisión del documento de tesis, ya que con sus observaciones se logró obtener un trabajo de mayor calidad.

A Leopoldo y Ricardo les doy las gracias, ya que en muchas ocasiones sus comentarios me permitieron encontrar soluciones a situaciones que percibía muy complejas y adicionalmente su convivencia me permitió hacer más agradable la vida académica que hacía muchos años no tenía.

A Nelly Condori Fenández e Isabel Diaz López les agradezco la oportunidad que me dieron de convivir con Uds. ya que desde siempre me recibieron con un trato de amigo de toda la vida.

A mis padres (Ramona y Clemente) y hermanos les agradezco su apoyo y buena vibra emitida para que llegara al final de este proceso.

A mis suegros (Guadalupe y Aurelio) les estoy eternamente agradecido ya que en mis ausencias de casa, debido a las diferentes estancias realizadas en el DSIC, me apoyaron con mis hijos haciendo la separación más llevadera, imposible olvidar en este apoyo a mi cuñada Amalia y a su esposo Miguel Badillo.

Les doy las gracias a las personas que trabajan en las diferentes dependencias e instituciones (DSIC, ITC, UAS, DGEST, UPV, ANUIES) que me apoyaron en la realización de los trámites administrativos que requería.

ANUIES	Asociación Nacional de Universidades e Instituciones de Educación Superior de la República Mexicana.
DSIC	Departamento de Sistemas Informáticos y Computación (UPV)
DGEST	Dirección General de Educación Tecnológica
ITC	Instituto Tecnológico de Culiacán
UAS	Universidad Autónoma de Sinaloa
UPV	Universidad Politécnica de Valencia (España)

Contenido

Capítulo 1. Introducción.

1.1 Orígenes de la tecnología de Almacenes de Datos.

La gestión de los datos en un computador ha experimentado una larga evolución desde sus orígenes hasta nuestros días. A principios de la década de los sesenta, el software de acceso a datos consistía en aplicaciones independientes, basadas en ficheros maestros almacenados en cinta magnética; lo que significaba un acceso secuencial a los datos. La aparición de los discos magnéticos en la década de los setenta representó un cambio cualitativo, éstos permitían el acceso directo a los datos (DASD, del inglés Direct Access Strorage Device), favoreciendo el desarrollo de nuevas organizaciones de ficheros. A partir de ese momento se produjo una acelerada evolución en la tecnología de acceso a datos que no ha parado hasta nuestros días [42].

Con el objetivo de conseguir una gestión integrada de los datos e independiente de la plataforma, que superase las limitaciones de los sistemas de ficheros clásicos, a principios de los setenta, se desarrollaron los sistemas de gestión de bases de datos (SGBD). Este tipo de sistemas experimentó también una rápida transformación, desde los pioneros sistemas jerárquicos hasta los actuales sistemas relacionales. La situación actual, en lo que a la gestión de los datos se refiere, se caracteriza por el uso extendido de la tecnología de bases de datos, concretamente de los sistemas de bases de datos relacionales. Estos sistemas son cada vez más robustos y fiables, y disponen de lenguajes e interfaces de acceso y manipulación cada vez más

amigables, lo que ha facilitado el acceso de los usuarios a esta tecnología.

Como fruto de esta ya larga evolución, se puede afirmar que, el uso de los sistemas de bases de datos, en las dos últimas décadas, ha provocado que las organizaciones almacenen en formato digital grandes volúmenes de datos con información histórica sobre la organización. Así, una vez satisfechas las necesidades de disponer de un sistema de información para la gestión del negocio, los usuarios exigen nuevas funcionalidades y prestaciones a sus sistemas, planteándose la necesidad de disponer de sistemas de información de apoyo a la toma de decisiones (DSS, Decisión Support Systems) que les permitan analizar el negocio, prever su evolución y decidir estrategias de futuro. Es el inicio del desarrollo de la tecnología de Almacenes de Datos (DW, Data Warehouse) y de las técnicas de análisis de datos conocidas como Minería de Datos (DM, Data Mining).

La aparición de los sistemas de almacenes de datos, ha abierto nuevos líneas de investigación y desarrollo en el área de Bases de Datos: nuevos modelos de datos, metodologías de diseño, optimización de consultas, técnicas de mantenimiento, herramientas de consulta y explotación, etc.

Las dos características más relevantes de este nuevo tipo de sistemas son: la adopción de un nuevo modelo de datos, el modelo multidimensional, y el uso de herramientas de análisis específicas, las herramientas OLAP (On Line Analytical Processing) [41].

Por otro lado las ventajas del uso de este tipo de sistemas para las organizaciones son obvias, les permiten mejorar sus servicios y productos y hacerlas más competitivas en un mercado cada vez más globalizado.

Un estudio hecho por Meta Group en 1998 (Meyer & Canon, 1998) detectó que el 95% de las compañías se planteaban ya la construcción de su sistema de almacén de datos. En el año 2000 se publicó en la referencia [69] un estudio de las ventas estimadas de productos relacionados con tecnología de almacenes de datos (Figura 1).

	1998	1999	2000	2001	2002
SGBD Relacional	900	1110	1390	1750	2200
Herramientas ETL	101	125	150	180	210
OLAP	2000	2500	3000	3600	4000

Figura 1 Ventas estimadas en millones de dólares (Informe, 2000)

La Figura 2 [78], muestra los 5 principales vendedores de herramientas de Business Intelligence (BI) en el mundo, en 2007.

Business Objects	894
SAS	679
Cognos	622
Hyperion/Oracle	529
Microsoft	480

Figura 2 Principales vendedores de BI (Informe, 2007)

1.2. Objetivos de la tesis.

El trabajo de tesis que se presenta, se enmarca en el área de Almacenes de Datos, su objetivo principal es:

La definición e implementación de un método para el mantenimiento de almacenes de datos.

Este objetivo, se refina en los siguientes objetivos específicos:

- Definición de un método para la integración de datos procedentes de distintas fuentes de datos operacionales. Se resuelve un caso particular.
- Definición de un método para la actualización del almacén de datos. Se proponen dos soluciones: un algoritmo en batch y un algoritmo en tiempo real.
- Implementación del método de actualización propuesto.
- Análisis y evaluación del método de actualización propuesto.
- Desarrollo de una herramienta que permita realizar el mantenimiento (integración y actualización) del almacén de datos.
- Aplicación a un caso de estudio.

1.3. Interés práctico del trabajo.

El valor práctico de este trabajo reside en:

- Se ofrece un entorno para el desarrollo de proyectos de almacenes de datos, basados en tecnología relacional.
- Se proporciona una herramienta para el mantenimiento del almacén de datos.

Para el contexto económico y social al que se van a transferir los resultados de la investigación, este trabajo va a representar un cambio importante, ya que significa la introducción a nivel nacional de la cultura sobre la importancia de los sistemas de apoyo a la toma de decisiones, empezando por la Universidad Autónoma de Sinaloa, beneficiaria directa de este trabajo.

1.4. Aportaciones de la investigación.

De acuerdo a los objetivos propuestos, las principales aportaciones de este trabajo de tesis son:

- La definición de un prototipo para la integración de datos procedentes de distintas fuentes de datos operacionales.
- La definición e implementación de un método para la actualización del almacén de datos.
- El desarrollo de una herramienta que permite realizar el mantenimiento del almacén de datos (incorpora los métodos anteriores).

1.5. Organización del documento.

El resto del documento se organiza como sigue:

En el Capítulo 2 se hace una presentación de los fundamentos teóricos sobre: Almacenes de Datos, Integración de Datos y Mantenimiento de Almacenes de Datos.

En el Capítulo 3 se describe nuestra propuesta sobre integración de datos, previamente se describen algunos problemas relacionados.

En el Capítulo 4 se revisan las políticas de mantenimiento de vistas materializadas y se realiza una revisión de los algoritmos existentes en la literatura sobre actualización de AD.

En el Capítulo 5 se describe nuestra propuesta para la actualización en línea de almacenes de datos.

En el Capítulo 6 se describe el sistema de mantenimiento de almacenes de datos que se ha desarrollado y se muestran los resultados de las pruebas realizadas.

En el Capítulo 7 se describe un caso de estudio relativo a la Universidad Autónoma de Sinaloa, para el que se ha aplicado nuestra propuesta.

En el Capítulo 8 se relacionan los problemas todavía abiertos en los que se puede continuar la investigación y se resumen las conclusiones del trabajo realizado.

Capítulo 2. Marco teórico.

2.1. Antecedentes.

Como ya se ha comentado en el capítulo anterior, durante las últimas décadas, un porcentaje significativo de datos corporativos han emigrado a bases de datos relacionales. Los sistemas relacionales son utilizados ampliamente en aplicaciones de gestión, estando especializados en el procesamiento transaccional (sistemas OLTP, On Line Transactional Processing). Se puede decir que actualmente la tecnología relacional constituye la tecnología más extendida para dar soporte a los sistemas de información organizacionales. Con el paso del tiempo, y en respuesta a las demandas de los usuarios, los fabricantes de sistemas relacionales han ido ofreciendo sus sistemas como herramientas para el desarrollo de sistemas de apoyo a la toma de decisiones, lo que se conoce como Almacenes de Datos. Un almacén de datos, es un sistema que almacena información histórica relativa a la actividad de una organización o negocio, con fines de análisis [1].

El término "almacén de datos" no es totalmente nuevo, ya que recuerda a un equipo de cómputo que a mitad de los años setenta extraía datos de las bases de datos de producción, los depuraba y los enviaba a una base de datos de usuario final [2]. IBM fue la primera compañía que acuñó el término "almacén de información" (information warehouse) en el año 1991. Mientras otros términos como "fábrica de información" (information factory) o "refinería de información" (information refinery) surgieron y desaparecieron, el

término “almacén de datos” (datawarehouse) es ahora un término ampliamente aceptado.

La primera generación de sistemas de almacenes de datos fue construida sobre ciertos principios establecidos por líderes de la industria. En la referencia [3] se reconoce a dos grandes pioneros en el área de Almacenes de Datos: Bill Inmon y Ralph Kimball. Estos dos científicos, han proporcionado las definiciones y los principios de diseño que la mayoría de los profesionales utilizan hoy en día. Aunque sus guías no sean seguidas exactamente, es común hacer referencia a la definición de almacén de datos de Inmon y a las reglas de diseño de Kimball.

Aunque los almacenes de datos han tenido una aceptación muy desigual por sectores comerciales y áreas geográficas, se puede decir que han captado la atención del mundo de los negocios. Se estima que hasta el año 1997 se gastaron en el mundo 15 billones de dólares en el desarrollo de sistemas de almacenes de datos (Menefee 1998), y que el mercado de las herramientas de procesamiento analítico en línea creció de 1 billón de dólares en 1996 a 4.3 billones de dólares en el 2004 [74] [75].

El uso de los sistemas de almacenes de datos ha puesto de relieve las ventajas e inconvenientes que el desarrollo de este tipo de sistemas reporta a las organizaciones.

Ventajas

1. Integrar datos históricos sobre la actividad de la organización (o negocio) en un único repositorio.
2. Analizar los datos del negocio desde la perspectiva de su evolución en el tiempo.
3. Prever tendencias de evolución del negocio.
4. Identificar nuevas oportunidades de negocio y tomar decisiones estratégicas.
5. Reducir los costes materiales y humanos en la toma de decisiones.

Inconvenientes

1. Riesgo de fracaso en la construcción del sistema, al subestimar los costes de captura y preparación de los datos.
2. Riesgo de fracaso en la construcción del sistema por cambios continuos en los requisitos de los usuarios.
3. Problemas con la privacidad de los datos.

En la última década, los almacenes de datos se han convertido en un nuevo campo de investigación dentro del área de Bases de Datos, atrayendo la atención de investigadores de áreas relacionadas [1]. En la referencia [53] se muestran algunos hitos importantes en la evolución de esta tecnología:

Inicio de los 90's

- Bill Inmon acuña el termino "almacén de datos".
- Gran interés de las empresas.
- Gran interés de los fabricantes de tecnología relacional.

- Desinterés del mundo académico.
- Reutilización de algunos tópicos clásicos en bases de datos: integración de fuentes heterogéneas, vistas materializadas, ...

Mediados de los 90's

- Inicio de un proyecto sobre almacenes de datos en la Universidad de Stanford.
- Interés del mundo académico.
- Aparición de herramientas comerciales dedicadas a sistemas de almacenes de datos.
- Publicación del artículo de revision "Research problems in Data Warehouse" de Jennifer Widom [68].

Año 2000

- Terminación del proyecto Europeo Data Warehouse Quality (DWQ).
- Sigue creciendo el interés por la investigación en el área de Almacenes de Datos.
- Numerosas herramientas comerciales están ya disponibles.
- Muchas empresas desarrollan su proyecto de almacén de datos.
- Publicación del artículo de revisión "Gulliver in the land of data warehousing: practical experiences and observations of a researcher" de Vassiliadis, en la conferencia Design and Management of Data Warehouse (DMDW'00) [69].
- Se observa una separación importante entre investigadores y profesionales.

- Los investigadores pasan por alto los problemas prácticos.
- Existe poca aceptación de los resultados de la investigación en el terreno industrial.

– Aparecen cerca de 20 artículos sobre el tema, publicados en congresos como VLDB (Very Large Data Bases), ACM SIGMOD (International Conference on Management of Data de ACM)

– Problemas y limitaciones en el momento:
  - Hay un déficit de "libros de texto" que describan una metodología estándar o extensamente aceptada [69], [79].
  - No hay estándares para metadatos.
  - No hay soluciones para el proceso de ETL (Extracción, Transformación y Carga).

Año 2003. Logros en investigación:

	Herramientas de implementación	Satisfacción de usuarios
Arquitecturas	Aceptable	Excelente
Modelado conceptual	Mala	Mala
OLAP	Excelente	Excelente
Lenguajes de consultas y procesamientos	Aceptable	Aceptable
Optimización	Mala	Aceptable
Aspectos físicos e indexación	Excelente	Excelente

Año 2004

En la referencia [61] se afirma que la investigación en Almacenes de Datos y herramientas OLAP (On Line Analytical Processing) ha alcanzado una madurez importante, aunque aún se observa una separación entre la investigación académica y los problemas del mundo real.

Aunque un almacén de datos es en esencia una base de datos, sus características especiales, de tamaño, explotación y mantenimiento han introducido nuevos retos en el área de bases de datos:

- Diseño de almacenes de datos
  - Definición del modelo multidimensional de datos
  - Estudio de los problemas de sumarizabilidad
  - Metodologías de diseño adecuadas

- Mantenimiento de almacenes de datos
  - Integración de datos
  - Transformación de datos
  - Técnicas de actualización

- Sistemas gestores de almacenes de datos
  - Estructuras de almacenamiento
  - Nuevos tipos de índices
  - Particionamiento de los datos
  - Técnicas de optimización

2.2. Almacén de datos.

Los datos que son de interés para una organización se encuentran, frecuentemente, dispersos en múltiples fuentes de información. Para que un usuario pueda acceder a esas fuentes de un modo integrado, hace falta construir un sistema que integre (física o lógicamente) los datos de esas fuentes. Sin esta integración sería necesario consultar independientemente cada una de las fuentes y luego combinar las

respuestas obtenidas. Una solución a este problema la proporcionan los sistemas de bases de datos federadas. Una base de datos federada es la integración lógica de distintas bases de datos que funcionan en servidores independientes (sin compartir recursos), y que pueden conectarse por una red local (LAN, Local Area Network) [43]. Una solución alternativa a este problema de la integración de fuentes de datos, la constituyen los sistemas de almacenes de datos.

Los almacenes de datos integran información procedente de múltiples fuentes de datos independientes en una única base de datos, funcionando como un repositorio de información histórica que puede ser consultado directamente por los analistas de la organización. El analista usa el almacén para detectar tendencias y anomalías dentro de las actividades del negocio, conocer el estado actual de áreas de interés de la organización, y tomar decisiones de futuro.
Usualmente el almacén de datos está separado de los otros sistemas y aplicaciones de la organización, en una base de datos propia. Una razón obvia para ello reside en el hecho de que en las organizaciones es frecuente que existan diferentes repositorios de datos, soportados por tecnologías diferentes, lo que dificulta el acceso integrado a la información. Otras razones son [2]:

- Los distintos objetivos de uso: los sistemas operacionales están orientados a los procesos (procesamiento de transacciones) mientras que los sistemas de almacenes de datos están orientados al análisis de los datos.
- Las distintas características de mantenimiento: los sistemas operacionales cambian constantemente, mientras que los sistemas

de almacenes de datos no son volátiles, sólo crecen incrementalmente.

Una definición clásica de almacén de datos, tomada de Inmon, es la siguiente:

Un almacén de datos es una colección de datos orientada al dominio, integrada, variante en el tiempo y no volátil, diseñada para dar apoyo al proceso de toma de decisiones en una organización [4].

Orientado al dominio significa que el almacén de datos está enfocado a los datos relacionados con un área de actividad del negocio. Ésta es la diferencia de enfoque con respecto a un sistema operacional que se diseña para dar soporte a los procesos básicos y cotidianos de la organización. La Tabla 1 muestra las diferencias entre los dos tipos de orientaciones en distintos contextos.

Operacional	Almacén de datos
Matrícula	Estudiantes
Prestamos	Clientes
Nóminas	Empleados
Tarjetas crédito	Clientes
Inventario	Productos
Ahorros	Clientes
Orientación al proceso	Orientación al dominio

Tabla 1. Dos tipos de orientación.

Integrada significa que los datos, independientemente de las fuentes de las que proceden, son almacenados en un único repositorio, unificando su formato (integración de formato) y unificando su

significado (integración semántica). La integración es un problema para muchas empresas, particularmente cuando existen muchos tipos de tecnología en uso. Por ello el proceso de integración exige costosos y largos procesos de limpieza, estandarización y agregación (resumen) de los datos.

Variante en el tiempo significa que los datos son asociados con un punto en el tiempo: diario, mensual, bimestral, trimestral, semestral, año fiscal, periodo de pago, etc. El almacén contiene grandes volúmenes de información histórica sobre las actividades de la organización y va variando en el tiempo, recibiendo periódicamente nuevos datos.

No volátil significa que los datos no cambian (no son actualizados) una vez que se añaden al almacén de datos. Cualquiera que use el almacén de datos tiene la seguridad de que la misma consulta producirá siempre los mismos resultados.

2.3. Sistema de almacén de datos.

En la Figura 3 se muestran los componentes y procesos que intervienen en un sistema de almacén de datos.

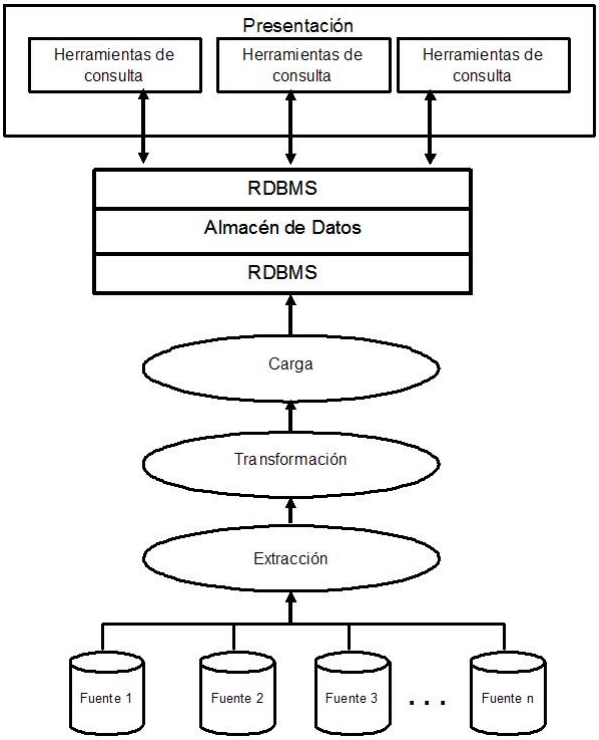


Figura 3. Sistema de almacén de datos.

Donde:

Fuentes: Representan los diferentes sistemas operacionales que suministran los datos al almacén de datos.

Extracción: Es el proceso que extrae datos de las fuentes operacionales para enviarlos al almacén de datos (**selección de datos**). Debe realizarse una selección de registros y campos de los sistemas operacionales, ya que no todos los datos de las fuentes son relevantes para el almacén de datos. Ejemplo: la Figura 4 ilustra una selección de datos de la fuente operacional; se han seleccionado dos campos del registro (*categoría* e *importe*) y sólo interesan los registros que en

categoría contengan como valor 1, 2 o 3 y que la fecha sea 30-09-2004.

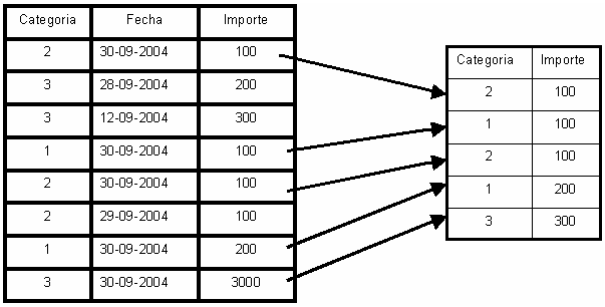


Figura 4. Extracción de datos.

Transformación: Es el proceso que prepara los datos de la manera adecuada, para ser incorporados al almacén de datos. El proceso de transformación se compone de las siguientes actividades: limpieza de datos, integración de formato, integración semántica, conversión de estructuras internas, integración de datos, resumen o agregación de datos.

- Limpieza de datos.

  Es necesaria porque los datos, procedentes de distintas fuentes, pueden contener errores y anomalías: inconsistencias, valores perdidos, violaciones de restricciones de integridad, etc. Para esta actividad, se pueden emplear herramientas de limpieza y/o herramientas de inspección de datos. Se puede hablar de dos tipos de limpieza:

  Limpieza moderada: Consiste en detectar anomalías comunes; por ejemplo, identificar que *Av.* y *Avenida* representan la

misma información o bien eliminar puntos, comas, comillas y otros signos de puntuación.

Limpieza intensa: Consiste en que el usuario proponga reglas para realizar la limpieza de datos; por ejemplo, el señor Juan López tiene una serie de concesiones con distintas direcciones, ¿debe considerarse como un mismo cliente o no?, esta decisión determina como se estructurará y se almacenará la información. Para trabajos de limpieza intensos, es conveniente utilizar herramientas que se han desarrollado para esas tareas. Existen dos grandes competidores: Enterprise/Integrator de Apertus Technologies y la herramienta Integrity Data Reengineering de Vality.

- Integración de formato: Una situación frecuente en las organizaciones consiste en que cada una de las fuentes operacionales está soportada por tecnología de diferente fabricante. Esto puede provocar que el mismo atributo sea de un determinado tipo en uno de los sistemas y de otro tipo en otro sistema. Por ejemplo, los números de cuenta bancaria o los números de teléfono pueden ser almacenados como un tipo alfanumérico en un sistema y como un tipo numérico en otro. La fecha puede ser almacenada en muchos formatos, “ddmmyy”, ”ddmmyyyy”, “yymmdd”, “yyyymmdd”, “dd mon yyyy”. Algunos sistemas almacenan los datos en miles de segundos, otros sistemas usan un entero que es el número de segundos a partir de un instante de tiempo, por ejemplo 1 de enero de 1900. Los atributos que almacenan datos de tipo dinero también son un problema, algunos sistemas almacenan el dinero como un valor

entero y esperan que la aplicación inserte el punto decimal, otros tienen el punto decimal ya incorporado. En sistemas diferentes, datos de diferente tamaño son usados para valores de tipo alfanumérico como el nombre, la dirección, la descripción del producto, etc.

- Integración semántica: La integración semántica hace referencia al significado de los datos. Ya que la información de un almacén de datos proviene de diferentes sistemas operacionales y éstos son usados por diferentes secciones de la organización, en cada sección se puede dar un significado distinto a los datos provocando confusión al analista. Es imprescindible, por lo tanto, que cada dato que se inserte en el almacén de datos tenga un significado preciso, que sea conocido por todos los usuarios. Para este fin, un almacén de datos debe disponer de un diccionario de datos que describa cada dato registrado en el almacén. Este diccionario es parte del almacén y es de hecho un repositorio de datos que describe los datos. El concepto de "datos acerca de datos" es referido usualmente como metadatos. En [65] se trata el problema de la integración semántica.

- Conversión de estructuras internas: Frecuentemente, los datos son estructurados de forma distinta, cuando pasan de un sistema operacional a un sistema de almacén de datos. En la Tabla 2, se muestra un ejemplo.

Sistema operacional	Almacén de datos
01 Registro de piezas 02 Clave 02 U/M 02 Descripción 02 Cantidad 02 Ultima entrada 02 Código de ensamblaje 02 Cantidad mínima …	01 Registro de piezas 02 Clave 02 Descripción 02 U/M 02 Código de ensamblaje 02 Cantidad …

Tabla 2. Conversión de estructuras.

- Resumen o agregación de datos: En los sistemas de almacenes de datos, es común realizar resúmenes de registros, aplicando funciones de agregación, cuando éstos pasan de los sistemas operacionales al almacén de datos. Ejemplo: en la Figura 5 se agregan los registros por categoría.

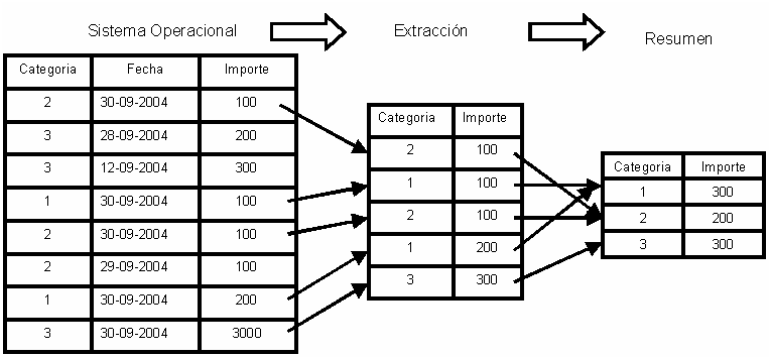


Figura 5. Agregación de datos.

- Integración de datos: Es frecuente que los datos (registros) que finalmente se van a insertar en el almacén de datos, se construyan a partir de otros registros de distintas fuentes. El proceso de

integración sigue una serie de reglas que se diseñan para garantizar que el dato que se va a cargar en el almacén es correcto.

Carga: Una vez que la información ha sido extraída de las fuentes y transformada, puede ser añadida al almacén de datos. Después de la carga inicial la estrategia de mantenimiento más frecuente consiste en actualizar el almacén periódicamente (diariamente, semanalmente, etc.).

Almacén de datos: Repositorio de datos que contiene la información que ha sido extraída de los diferentes sistemas operacionales. Este repositorio de datos puede ser consultado directamente por los analistas de la organización.

Presentación: El componente más externo en un sistema de almacén de datos es el componente de presentación, éste elabora una presentación amigable de los datos para los usuarios, ocultando la complejidad del esquema del almacén de datos.

2.4. Diseño de almacenes de datos.

Los conceptos ‘esquema en estrella’ (star scheme), ‘esquema en copo de nieve’ (snowflake scheme), ‘esquema en constelación’ (constellation scheme), ‘blizzard o tempestad de nieve’ (snowstrom scheme), ‘mapa de dimensión’ (dimension map), hipercubo (hypercube design), ‘modelo multidimensional’ (multidimensional model), han sido utilizados por distintos autores para abordar el modelado de almacenes de datos [46]. Independientemente de la

metodología seguida, todas ellos utilizan un modelo multidimensional de datos.

En un esquema multidimensional se modela una actividad de la organización que es objeto de análisis (*hecho*). El esquema multidimensional representa en el centro la actividad con sus indicadores relevantes (*medidas*) y en los extremos las *dimensiones* que caracterizan la actividad al nivel de detalle al que se ha decidido representar la actividad en el almacén. Cada dimensión está descrita por un conjunto de atributos organizados en jerarquías de niveles, atendiendo a las dependencias funcionales entre estos atributos (Figura 6). La representación gráfica de estos esquemas multidimensionales puede ser variada: cubos n-dimensionales donde, cada eje representa una dimensión y las celdas del cubo representan los hechos sobre la actividad; o esquema en estrella, donde en el centro de la estrella se representa la actividad con sus medidas y en cada punta de la estrella se representa una dimensión de la actividad.

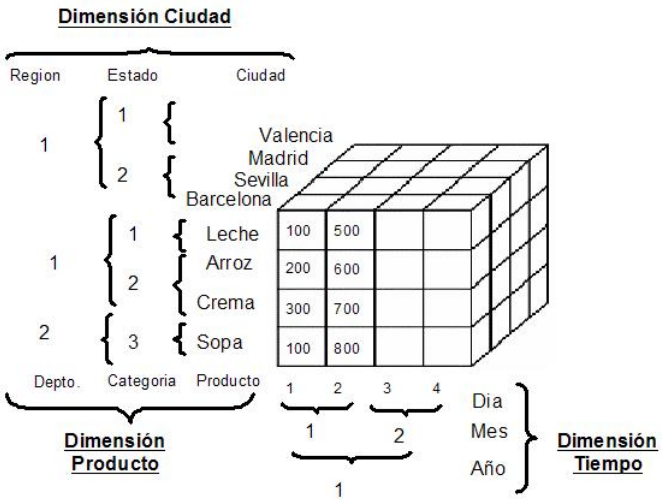


Figura 6. Esquema multidimensional en forma de cubo.

En la Figura 6 cada celda del cubo contiene medidas (importe de la venta, número de unidades vendidas, etc.) relativas a la venta de un producto (valor de la dimensión Producto), en una ciudad (valor de la dimensión Ciudad) en una fecha determinada (valor de la dimensión Tiempo). Un ejemplo de hecho sería [(Ciudad: Valencia, Producto: sopa, Tiempo: 19-10-2004), (100, 20)], que debe interpretarse como "Las ventas de *sopa* realizadas en *Valencia* el día *19-10-2004* fueron de 20 unidades por un importe de 100 pesos". Otra forma de representar un esquema multidimensional se muestra en la Figura 7.

Durante el análisis de datos sobre un almacén de datos, el analista usualmente consulta información agregada, rara vez consulta los hechos individuales al nivel de detalle al que están almacenados. Ello se realiza a través de las dimensiones: sus niveles y jerarquías.

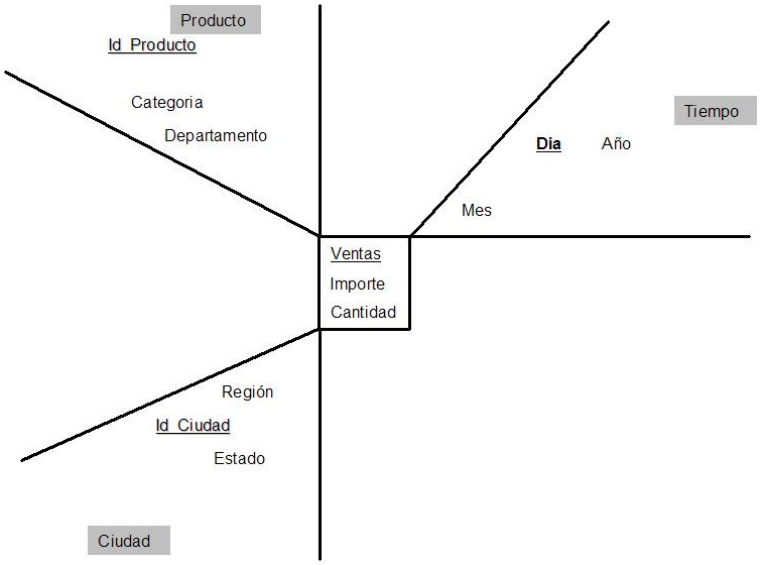


Figura 7. Esquema multidimensional en forma de estrella.

Como ya se ha comentado, los atributos de una dimensión no son independientes entre sí, existe siempre un atributo identificador (en el ejemplo: Día en la dimensión Tiempo, Id_producto en la dimensión Producto, y Id_ciudad en la dimensión Ciudad) y suele haber dependencias funcionales entre los atributos, lo que permite definir jerarquías. Una jerárquica de una dimensión es un conjunto ordenado de atributos que parte del nivel raíz (atributo identificador). Las jerarquías de una dimensión van a representan los diferentes niveles de agregación de las medidas durante el análisis de datos, y la vía o el camino para realizar operaciones de agregación (ROLL) y disgregación (DRILL) propias de las herramientas de OLAP. La Figura 8 muestra jerarquías en las tres dimensiones del ejemplo.

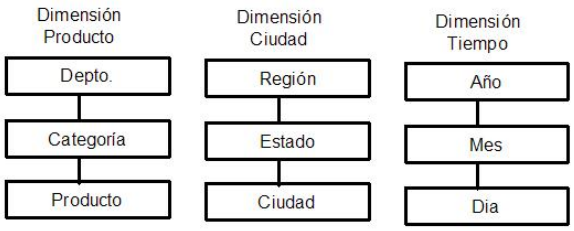


Figura 8. Jerarquías en las tres dimensiones.

Obsérvese que en la Figura 8 las tres dimensiones del ejemplo sólo cuentan con una jerarquía. En las dimensiones de los esquemas multidimensionales reales es usual que en una dimensión exista más de una jerarquía, como se ilustra en el ejemplo de la Figura 9.

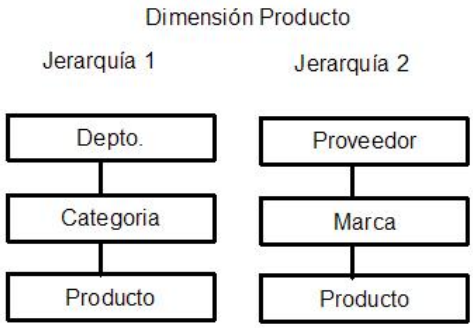


Figura 9. Dimensión Producto con múltiples jerarquías.

2.5. Herramientas OLAP.

El término OLAP, fue introducido en el año 1993 por E. F. Codd [47], para referirse a una nueva familia de herramientas para el análisis de datos. Las herramientas OLAP se basan en el modelo multidimensional de datos, es decir presentan a los usuarios una visión multidimensional de los datos, independientemente del servidor que soporte el almacén de datos.

Estas herramientas representan una mejora respecto a los tradicionales generadores de informes de los gestores de bases de datos. En cualquier caso, son herramientas que permiten explorar los datos almacenados y que no permiten extraer conocimiento de ellos, como harán las herramientas de Minería de Datos.

Las características básicas de estas herramientas son [50]:

- Ofrecer una visión multidimensional de los datos: hechos-dimensiones
- No imponer restricciones sobre el número de dimensiones
- Ofrecer simetría para las dimensiones
- Permitir definir jerarquías sobre las dimensiones
- Ofrecer operadores sobre consultas: operador de DRILL y operador de ROLL [49]:
  - Agregación (ROLL): permite cambiar (o eliminar) un criterio de agrupación en el análisis, agregando los grupos actuales, obteniendo así un informe más resumido (Figura 10).
  - Disgregación (DRILL): permite cambiar (o introducir) un nuevo criterio de agrupación en el análisis, con lo que se disgregan los grupos actuales, obteniendo un informe más detallado (Figura 11).
- Permitir expresar condiciones sobre las dimensiones y criterios de agrupación de los datos
- Ser transparentes al tipo de tecnología que soporta el almacén de datos: ROLAP (relacional) o MOLAP (multidimensional).

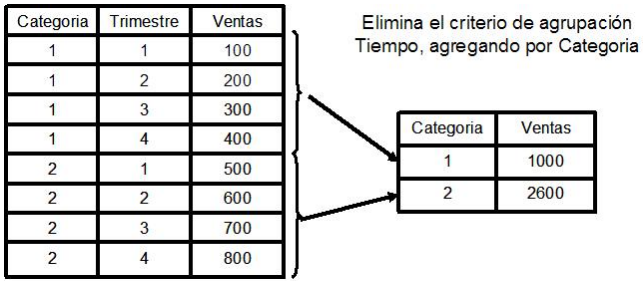


Figura 10. Análisis de datos con agregación (ROLL).

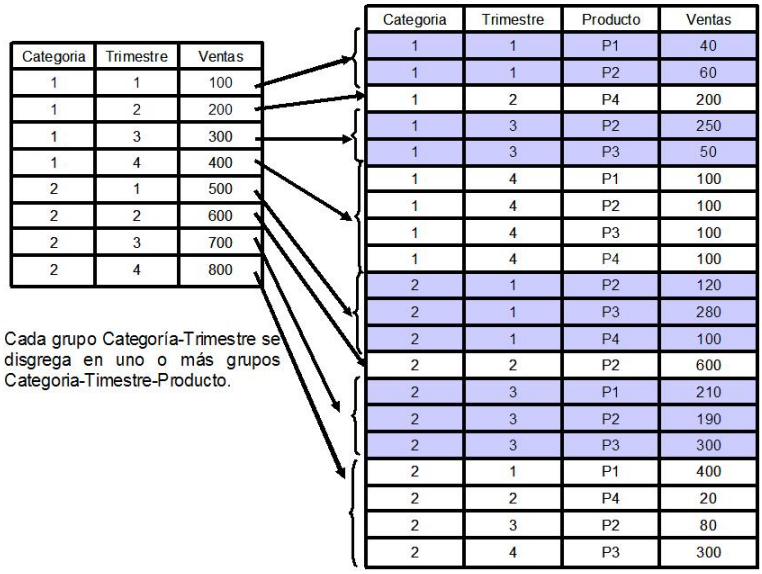


Figura 11. Análisis de datos con disgregación (DRILL).

Las herramientas OLAP ofrecen una visión multidimensional del almacén, que permite a los usuarios diseñar informes de una forma amigable y sencilla, seleccionando exclusivamente los atributos de las dimensiones y de los hechos que se desean ver en el informe. La herramienta debe tener definida la correspondencia entre el esquema multidimensional mostrado y el esquema real del almacén de datos,

siendo éste último transparente para el usuario. Además la herramienta ofrece facilidades para el análisis de datos: nuevas funciones estadísticas, operadores específicos: DRILL, ROLL, SLICE, DICE, definición de condiciones para filtrar los datos, etc. Todo esto se ilustra en la Figura 12.

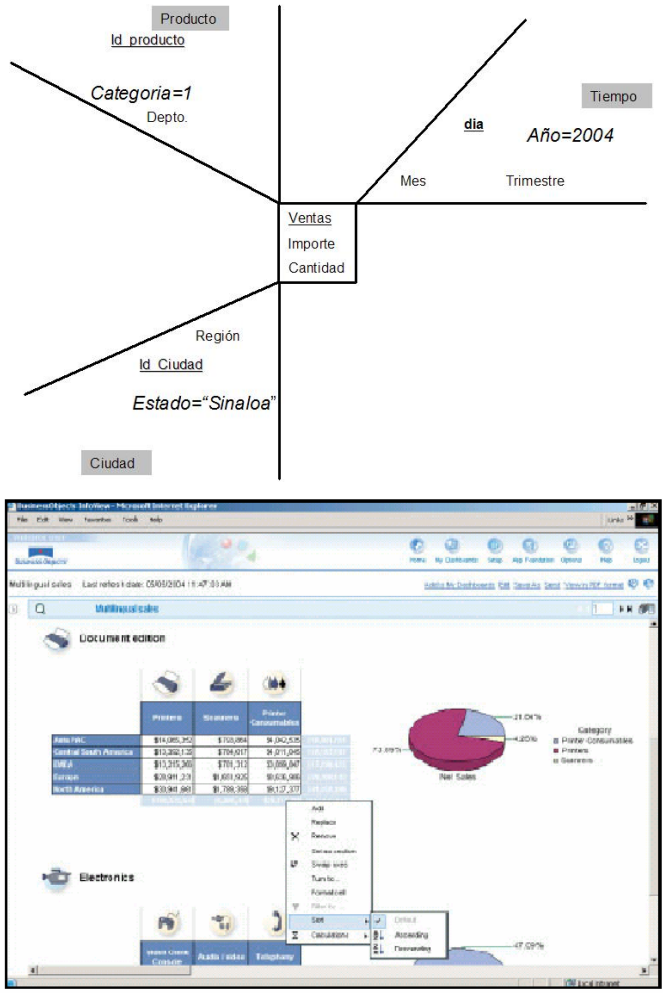


Figura 12. Análisis de datos con una herramienta OLAP.

2.6. Sistemas ROLAP y sistemas MOLAP.

Así como para las metodologías de diseño de almacenes de datos se ha consensuado y adoptado un modelo multidimensional de datos, respecto a la implementación (esquema físico) del esquema conceptual (multidimensional), se han seguido dos aproximaciones: los sistemas ROLAP y los sistemas MOLAP.

- Sistemas **ROLAP** (Relational On-Line Analytical Processing): se trata de usar gestores de bases de datos relacionales, con ciertas extensiones y facilidades nuevas. Ejemplos: Informix, SQL Server, Oracle. En estos sistemas, las herramientas OLAP son las encargadas de transformar el esquema relacional del almacén en un esquema multidimensional para el usuario, se ilustra en la Figura 13.
- Sistemas **MOLAP** (Multidimensional On Line Analytical Processing): se trata de construir gestores de propósito específico, basados en el uso de estructuras de almacenamiento de tipo matrices multidimensionales (Figura 13), que favorezcan el tipo de análisis que se hace en estos sistemas. Ejemplo: Hyperion Esbasse OLAP Server.

La principal ventaja de los sistemas MOLAP es que mantienen una correspondencia directa entre los datos almacenados y la vista que de ellos tiene el usuario (multidimensional). Se ha demostrado que las matrices multidimensionales son estructuras de datos muy adecuadas para el análisis de datos.

La ventaja principal de los sistemas ROLAP reside en la posibilidad de utilizar una tecnología ampliamente extendida y utilizada para la gestión de datos, los sistemas relacionales [41].

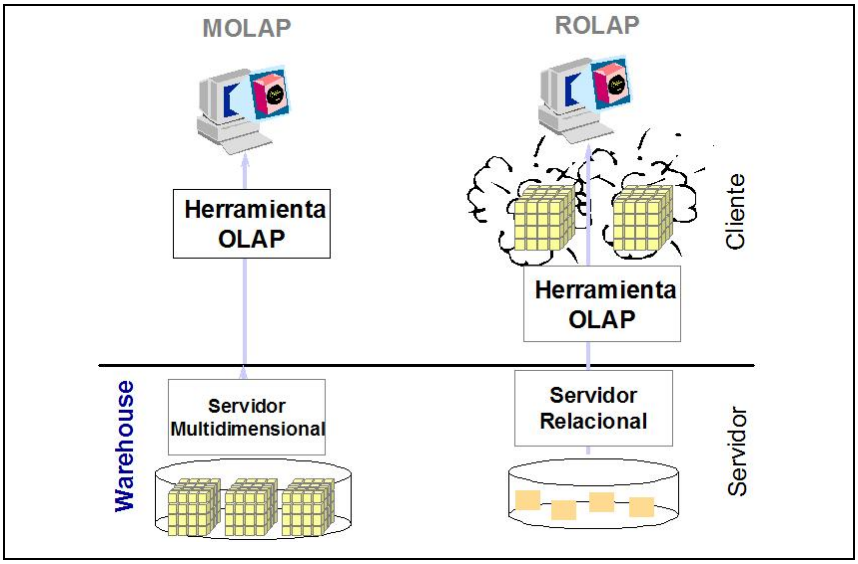


Figura 13. Sistemas MOLAP y sistemas ROLAP.

2.7. Mantenimiento de almacenes de datos.

Un almacén de datos almacena información histórica sobre las actividades de una organización. Esta información es recogida de los distintos sistemas operacionales de la organización. Para que el almacén sea consistente con la realidad que representa, debe ser actualizado periódicamente. Esta tarea no es sencilla, presentando numerosos problemas, es lo que se conoce como "mantenimiento de almacenes de datos".

Un sistema de almacén de datos puede verse como una jerarquía de datos, que tiene su origen en los datos almacenados en los sistemas operacionales (fuentes) y que termina con los datos almacenados en el

almacén de datos [44]. Esta jerarquía determina el flujo de datos que tiene lugar entre los sistemas operacionales y el sistema de almacén de datos. Este flujo se realiza por medio de los siguientes procesos:

a) Extracción de los datos
b) Transformación de los datos
c) Transporte y carga en el almacén

El subsistema o módulo encargado de realizar estos procesos se conoce como sistema de ETL (Extracción, Transformación y Carga) del inglés "Extract, Transform and Load".

El proceso de extracción de datos consiste en la búsqueda y recuperación de datos que son relevantes para el almacén, usando para ello los sistemas operacionales de la organización o fuentes de datos externas. Este proceso exige frecuentemente, un trabajo de programación ya que no existe un método estándar que permita hacerlo de manera automática debido a que las fuentes de datos suelen estar soportadas por distintas tecnologías. En este trabajo de tesis no se considerará el proceso de extracción y se analizarán sólo los procesos de transformación y carga de datos.

2.7.1. Transformación de datos.

En el desarrollo del sistema de información de una organización, es frecuente la creación de bases de datos independientes que son diseñadas para satisfacer los requisitos de las aplicaciones a las que sirven, generando problemas de heterogeneidad. En la referencia [9] se describen dos tipos de heterogeneidad:

- Heterogeneidad de formato: hace referencia a las diferencias entre definiciones locales, tales como tipo de datos, formato o precisión.
- Heterogeneidad semántica: hace referencia a las diferencias en el significado de los datos (variación en la manera en la que los datos con el mismo significado son representados y estructurados en diferentes sistemas). Como ejemplo, considérese una empresa donde en el área de producción se utiliza como unidad de medida el metro y en el área de ventas la unidad de mediada utilizada es la yardas. En esta situaciones, la información debe ser integrada semánticamente antes de ser registrada en el almacén de datos, para que las personas (analistas) que toman decisiones en la empresa, puedan disponer de los datos de una manera segura y puedan realizar análisis que conduzcan a una correcta y oportuna toma de decisiones [13], esta idea se ilustra en la Figura 14.

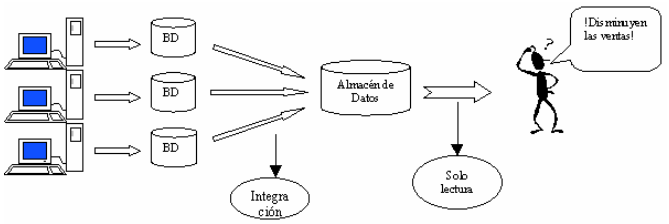


Figura 14. Consolidación de datos.

En la referencia [30] se señalan las tres causas principales que provocan la heterogeneidad semántica:

1. Distintas perspectivas: este es un problema de modelado que surge en la fase de diseño de las bases de datos. Diferentes grupos de diseñadores adoptan un punto de vista distinto cuando modelan la

misma realidad. Por ejemplo, el uso de diferentes nombres para referirse al mismo concepto.

2. Construcciones equivalentes: la variedad de lenguajes de modelado existentes, conduce a situaciones en las que un mismo concepto del mundo real es modelado de forma distinta en distintos ámbitos de aplicación.
3. Especificaciones de diseño incompatibles: diferentes especificaciones de diseño conducen a diferentes esquemas. Por ejemplo, en un esquema se puede especificar que un viajero sólo puede tener una reserva, mientras que en otros esquemas se puede decir que un viajero puede tener muchas reservas.

Arquitectura general para la integración de datos.

Las fuentes de datos (sistemas operacionales) se comunican con el almacén de datos a través de un wrapper/monitor (Figura 15), cuya función principal es detectar actualizaciones en las fuentes de datos y enviarlas al almacén de datos. El trabajo del módulo integrador es integrar los datos seleccionados de las diferentes fuentes, solucionar cualquier conflicto y propagarlo al almacén de datos [6].

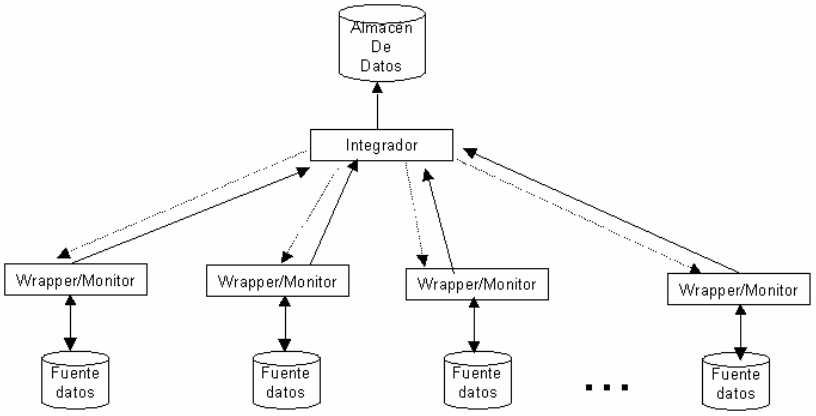


Figura 15. Arquitectura general para la integración de datos.

El proceso de integración de datos resuelve el problema que representa combinar datos residentes en diferentes fuentes, permitiendo proporcionar al usuario una vista unificada de los datos [11]. Los sistemas de integración de información (también conocidos como mediadores o agente recolector de información) proporcionan al usuario una interfaz uniforme, partiendo de una variedad de fuentes de información [12].

2.7.2. Actualización de almacenes de datos.

Para que el contenido del almacén de datos esté sincronizado con el contenido de las fuentes (sistemas operacionales), cualquier actualización en una fuente, que sea relevante para el almacén, debe ser transmitida al almacén para que éste sea actualizado. El desajuste temporal entre estos dos procesos de actualización, puede hacer que el almacén no esté sincronizado con las fuentes, es decir el almacén no contenga datos que serían derivables en un instante de tiempo, del contenido de las fuentes.

Ejemplo: considérese las relaciones r1(W,X), r2(X,Y) del sistema operacional y la relación V del almacén de datos definida como $V=\Pi_{W,Y}(r1 \bowtie r2)$ como se muestran en la Figura 16.

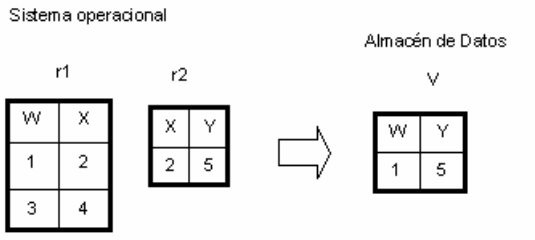


Figura 16. Derivación de la relación V a partir de las relaciones r1 y r2.

Si un analista consulta la extensión de la relación V, obtendrá la respuesta {[1, 5]}.

Ahora supóngase que se producen dos actualizaciones en el sistema operacional: insert(r1, [4,5]) e insert(r2, [4,5]). En la Figura 17 se muestra el estado de las relaciones r1 y r2 y de la relación V después de realizarse las actualizaciones en la fuente.

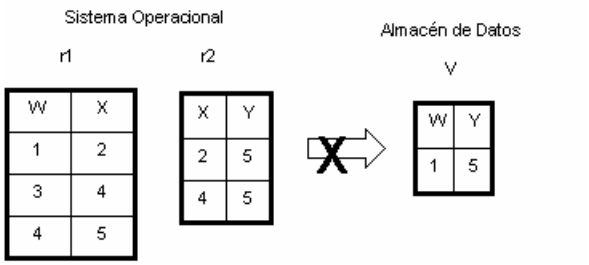


Figura 17. Relación V desincronizada con las relaciones r1 y r2.

Obsérvese que el estado de V representado en la Figura 16 y en la Figura 17 es el mismo, lo que indica una desincronización entre el sistema operacional y el almacén de datos, ya que las actualizaciones

que se han realizado en las relaciones r1 y r2 no se han reflejado todavía en la relación V. Cuando el mismo analista realiza de nuevo la consulta sobre la relación V (considerando el estado reflejado en la Figura 15) obtiene la misma respuesta {[1, 5]}, es decir, no obtiene la información más reciente que se puede derivar de las relaciones r1 y r2, aunque sí obtiene una información consistente con la lectura anterior. Se dice que el analista realiza una consulta "consistente" aunque el almacén de datos no está "sincronizado" con las fuentes.

La historia de un almacén de datos puede verse como una secuencia de estados[1] (o versiones) donde la transición de una versión a la siguiente se produce por la ejecución de una transacción de mantenimiento. Si un analista accede a la misma versión sus lecturas serán consistentes, independientemente de que la versión esté o no sincronizada con las fuentes.

El estado de la relación V sincronizada con las relaciones r1 y r2 sería el que se muestra en la Figura 18.

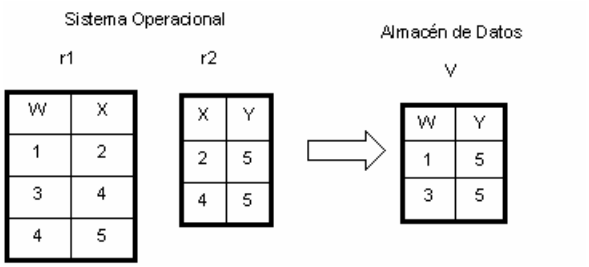


Figura 18. Relación V sincronizada con las relaciones r1 y r2.

[1] Obsérvese que el concepto de estado (o versión) del almacén no coincide con el concepto de estado de bases de datos.

Objetivos del mantenimiento de almacenes de datos.

Un objetivo deseable en el mantenimiento de almacenes de datos es conseguir que los analistas del sistema puedan ejecutar sus transacciones de consulta simultáneamente a la ejecución de las transacciones de mantenimiento del almacén, sin que unas transacciones bloqueen a las otras, debido a que requieran mucho tiempo de ejecución. Una forma de evitar este bloqueo consiste en permitir que los analistas realicen consultas inconsistentes durante su sesión de trabajo, debido a que durante la sesión el almacén es actualizado. Sin embargo, con frecuencia estas inconsistencias no son aceptadas, ya que los analistas desean ver datos consistentes durante una sesión de trabajo. Durante el análisis sería inaceptable tener diferentes resultados de una misma consulta. En base a lo anterior, podemos señalar los siguientes objetivos para las políticas de actualización de almacenes de datos:

- Conseguir que los analistas tengan acceso a un mismo estado (o versión) del almacén de datos durante su sesión de trabajo ("lectura consistente"), y que este estado esté lo más sincronizado posible (en el tiempo) con las fuentes.
- Reducir al máximo el tiempo que el almacén de datos se encuentra inhabilitado para los usuarios (analistas), debido al proceso de mantenimiento.

¿Cómo se puede garantizar a los analistas el acceso a un mismo estado ("lectura consistente") del almacén de datos?

El método más comúnmente usado en los sistemas comerciales de almacenes de datos, para garantizar la consistencia sin bloqueos consiste en realizar el mantenimiento del almacén durante la noche, tiempo en el que el almacén de datos esta inhabilitado para los analistas, es decir la transacción de mantenimiento y las transacciones de consulta de los analistas nunca se ejecutan al mismo tiempo. Este método se ilustra en la Figura 19.

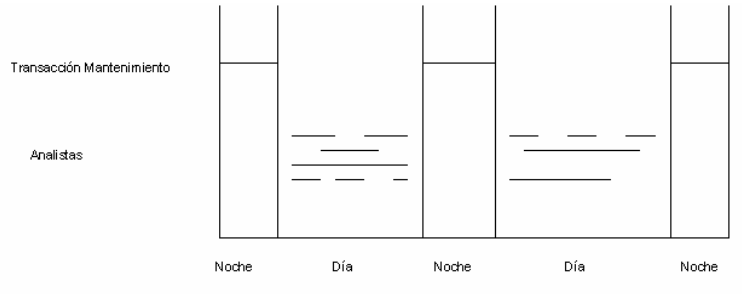


Figura 19. Método de actualización del almacén de datos, usado por los sistemas comerciales.

La Figura 19 nos muestra que la transacción de mantenimiento y las transacciones de los analistas jamás se ejecutan al mismo tiempo logrando con esto:

- Ejecución sin bloqueo.
- Los lectores tienen garantizado el acceso a un mismo estado del almacén de datos, es decir las lecturas son consistentes.

Esta política de mantenimiento presenta los siguientes problemas:

- En una organización distribuida geográficamente, puede no existir una franja de tiempo común en la que pueda inhabilitarse el almacén de datos.
- El tiempo disponible para la transacción de mantenimiento es limitado, y por lo tanto puede ser insuficiente.

Por ello va a ser necesario investigar otras políticas de mantenimiento, que cumplan con los objetivos antes marcados.

Métodos para la actualización de almacenes de datos.

La necesidad de mecanismos eficientes para actualizar almacenes de datos ha sido manifestada por distintos autores en la literatura sobre el tema:

Buneman y Clemons (1979), proponen vistas para el manejo de alertas (alerters), las cuales monitorizan una base de datos e informan a un usuario o aplicación si un estado de la base de datos, descrito por la definición de la vista ha sido alcanzado [20].

Garden et otros (1984), consideran una vista concreta (vista materializada) como un método candidato apropiado para el apoyo de consultas en tiempo real. Sin embargo desecharon este método debido a la ausencia de un algoritmo eficiente que mantenga la vista materializada actualizada con la base de datos relacional [20].

Horwitz y Tim Teitelbaum (1985), proponen un modelo para la generación de ambientes basado en lenguajes que usa una base de

datos relacional junto con atributos. Sugieren algoritmos para la actualización incremental de vistas [20].

Alexandron Labrinidis y Nick Roussopoulos (1998), proponen el algoritmo MAUVE, para la actualización incremental en línea de vistas, en él se usan técnicas de TIMESTAMPS que permiten accesos consistentes de lectura al almacén de datos mientras está actualizándose [51].

Dallan Quass (1999), en Inderpal Singh Mumick, presenta técnicas de mantenimiento incremental eficientes sobre una tabla resumen y múltiples tablas resumen [21].

Dallan Quass (1999), "Thesis: Materialized Views in Data Warehouse", presenta el algoritmo 2VNL y el algoritmo NVNL, para el mantenimiento en línea de vistas materializadas. Este algoritmo hace uso de redundancia horizontal para mantener dos versiones de la base de datos [22].

La Universidad de Stanford (2000) desarrolló un prototipo de sistema de almacén de datos que cuenta con cuatro algoritmos para el mantenimiento de vistas. Recálculo total, Sumarizado-delta con instalación basado en cursor, Sumarizado-delta con instalación batch, Sumarizado-delta con instalación de sobre-escritura [23].

Themistoklis Palpanas y otros (2002), proponen un algoritmo que realiza mantenimiento incremental que aplica a todas las funciones de

agregación incluyendo las que no son distributivas para todos los operadores [62].

Songting Chen, Bin Liu y Elike A. Rundensteiner (2004), describen el algoritmo TxnWrap, que permite realizar mantenimiento de vistas (usando multiversiones) definidas sobre fuentes de datos distribuidas. En este algoritmo se considera la actualización concurrente de los datos y los cambios de esquemas realizados [52].

Ki Young Li y Myoung Ho Kim (2005), proponen el mantenimiento de múltiples vistas reutilizando resultados intermedios que fueron usados para calcular otra vista materializada (los autores establecen que las vistas frecuentemente comparten expresiones comunes entre ellas) y con ello reducir el coste total de mantenimiento [77].

Songting Chen y Elike A. Rundenteiner (2006), describen el marco de trabajo llamado DyDa, en el cual desarrollan un método que resuelve las anomalías que se presentan cuando realizan el mantenimiento de vistas bajo actualizaciones concurrentes de los datos en las fuentes de datos o bien en los esquemas de las fuentes de datos [64].

Conclusiones del estudio

Las conclusiones más relevantes que se derivan del estudio realizado sobre métodos de actualización de almacenes de datos son las siguientes:

1. Existen muchos algoritmos para la actualización de almacenes de datos que trabajan fuera de línea, lo que provoca que el almacén de datos no esté disponible permanentemente para los usuarios. Esta situación representa un problema ya que muchas compañías internacionales no disponen de una franja de tiempo común donde se pueda inhabilitar el almacén de datos a los usuarios.

2. Otro problema que se detecta en la literatura sobre algoritmos de actualización de almacenes de datos es que una gran cantidad de ellos realizan la actualización del almacén en batch o lotes, lo que provoca que el almacén de datos no esté siempre sincronizado con las fuentes de datos.

3. Por último, existen algoritmos que realizan la actualización del almacén de datos en tiempo real, sin embargo el problema que se detecta es que sólo definen vistas Select-Project-Join, omitiendo en la mayoría de los trabajos el cálculo de las funciones de agregación usuales en las vistas materializadas de los almacén de datos.

En este trabajo de tesis se presentan dos algoritmos que realizan la actualización del almacén de datos en batch y en tiempo real, ambos permiten que el almacén de datos este disponible a los analistas las 24 horas del día, los 7 días de la semana y los 365 días del año y garantizan consultas consistentes ya que permiten tener un número ilimitado de versiones del almacén. El algoritmo de actualización en tiempo real permite realizar actualizaciones de vistas materializadas Select-Project-Join, con funciones de agregación tales como sum, count, avg, min y max.

Capítulo 3. Integración de datos.

En el contexto de los sistemas de almacenes de datos, se entiende por "integración de los datos" el proceso por el que datos procedentes de distintas fuentes son transformados y combinados antes de ser incorporados al almacén de datos.
La integración de datos es uno de los campos de investigación más antiguos en el área de base de datos. Surgió con la extensión del uso de los sistemas de bases de datos en las organizaciones (hacia 1960), debido a que el número de aplicaciones y dispositivos de almacenamiento crecían continuamente, y con ellos la necesidad continua de integrar datos [65].

En los sistemas de almacenes de datos se realiza un gran esfuerzo en la integración de datos, ya que estos sistemas son actualizados con datos que pueden proceder de fuentes muy variadas. Esta variedad en las fuentes aumenta la probabilidad de que haya datos sucios o datos erróneos. Recuérdese que los almacenes de datos son utilizados para la toma de decisiones, de modo que la calidad de sus datos es fundamental para evitar las conclusiones erróneas ("si basura se introduce, basura se obtiene") [63].

3.1. Métodos para la integración de datos.

En la referencia [14] se afirma que existen dos aproximaciones básicas para resolver el problema de la integración de datos: *procedimental y declarativa*. En el método procedimental, los datos son integrados en función de un conjunto de necesidades predefinidas de información,

sin recurrir a un conocimiento explícito de los esquemas (estructuras) de los datos integrados. Trabajos donde se usa esta aproximación aparecen en las referencias [15] y [16]. En la referencia [15] se presenta la implementación de un motor (máquina) basado en reglas, donde cada una de éstas es responsable de manejar un tipo de notificación de cambio y éste es implementado como un método. El método es llamado cuando el wrapper/monitor genera un aviso de cambio del tipo apropiado. El cuerpo del método realiza dos procesos: a) integra los datos seleccionados de las diferentes fuentes, solucionando cualquier conflicto, b) incorpora los datos al almacén de datos.

En el método declarativo el objetivo es modelar los datos de las fuentes por medio de un lenguaje apropiado para ello, y construir una representación unificada, que será utilizada cuando se hagan consultas al sistema. Trabajos donde se usa este método se pueden encontrar en las referencias [14], [31], [32] y [33]. En la referencia [32] se describe el sistema SIMS. En SIMS se define un modelo del dominio de la aplicación usando un lenguaje de representación del conocimiento que establece un vocabulario para describir los objetos del dominio, sus atributos y las relaciones entre ellos. El modelo del dominio se describe en el lenguaje Loom, miembro de la familia KL-ONE de los sistemas de representación del conocimiento. Los objetos básicos en Loom son las clases (también llamados conceptos). Un ejemplo de la definición de la clase Airport en el lenguaje Loom se muestra en la Figura 20:

```
(defconcept Airport
  :is-primitive (:and Port
                      (:all name Airport-Name)
                      (all runway-of Runway)
                      (:all altitude Number)
  :annotations((key(name))))
```

Figura 20 Definición de una clase Airport en el lenguaje Loom

Esta definición expresa que un Airport es una subclase de Port y hereda los roles de Port, y ésta cuenta con tres roles adicionales: name, runway-of y altitude. El término is-primitive es usado para indicar que ésta definición puede estar incompleta.

SIMS acepta consultas en un lenguaje uniforme de alto nivel, procesa las consultas de una manera transparente para el usuario y devuelve los datos solicitados. Las consultas en el SIMS no necesitan contener información que describa qué fuentes son relevantes para encontrar las respuestas o dónde están localizadas. Las preguntas no requieren indicar la concatenación de las fuentes a partir de donde la información será obtenida, es tarea de SIMS determinar cómo recuperar e integrar los datos necesarios para contestar a una pregunta de manera eficiente y transparente. La consulta mostrada en la Figura 21 requiere los códigos del país de los aeropuertos con pista de despegue con longitud de más de 10,000 pies. La línea número 1 especifica que todos los posibles valores de la variable ?contry-code deben ser regresados, de la línea 2 a la 6 establece restricciones que deben ser satisfechas en la recuperación de los valores deseados.

```
Line 1: (retrieve (?country-code)
Line 2: (:and (Airport ?aport)
Line 3: (country-code ?aport ?country-code)
Line 4: (runway-of ?aport ?rway)
Line 5: (structure-length ?rway ?rlength)
Line 6: (>= ?rlength 10000)))
```

Figura 21. Consulta en el sistema SIMS.

3.2. Trabajos relacionados.

En la referencia [8] se establece que en los últimos años se han divulgado muchos proyectos de investigación enfocados al el tema de la integración lógica de datos; entre estos sistemas se pueden citar Garlic [70], Yat [71], The Information Manifold [72], Tisimmis [35] y Disco [73]. El objetivo de estos sistemas es la explotación de varias fuentes de datos independientes, como fueran una única fuente de datos haciendo uso de un esquema global unificado. Un usuario realiza la consulta en términos del esquema global. Para ejecutar la consulta el sistema realiza las siguientes acciones:

- El sistema convierte la consulta en subconsultas expresada en términos de los esquemas locales.
- Las subconsultas se envían a las fuentes de datos locales.
- El sistema recupera los resultados y los combina para mostrárselos al usuario.

Los sistemas de integración lógica de datos pueden ser clasificados de acuerdo al método utilizado para relacionar las fuentes de datos locales con el esquema global unificado. En la referencia [17] y [66] se describen dos métodos de integración lógica de datos,

representados por una arquitectura basada en un esquema global y un conjunto de fuentes de datos. Las fuentes de datos contienen los datos reales, mientras que el esquema global proporciona una vista virtual integrada de las fuentes de datos. El primer método llamado GAV "vista global", del inglés "Global as View", exige que el esquema global esté expresado en términos de las fuentes de datos (especificar el esquema global en términos de los datos residentes en las fuentes). El segundo método llamado LAV "vista local", del inglés "Local as View", exige que el esquema global sea especificado independientemente de las fuentes de datos y las relaciones entre el esquema global y las fuentes sean establecidas definiendo cada fuente como una vista del esquema global. Para ilustrar estos métodos supóngase que se dispone de las relaciones mostradas en la Figura 22, S1 (productos del proveedor "A"), S2 (productos del proveedor "B") y S3 (clasificación de todos los productos disponibles). El atributo DesProd o NomProd representa la clave primaria en las relaciones. También se cuenta con el esquema global EG.

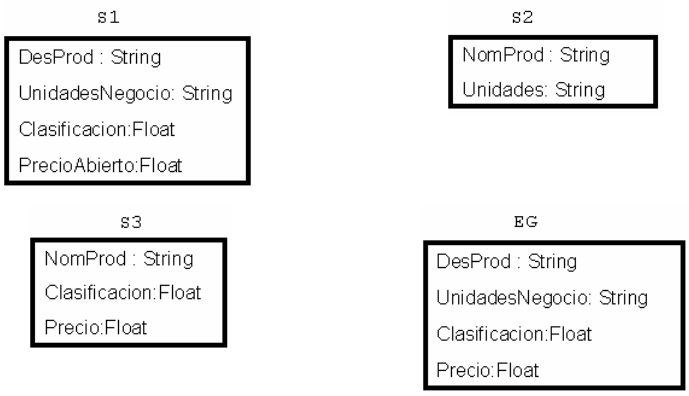


Figura 22. Ejemplo de GAV y LAV.

En el método GAV, una vista es definida en términos de las relaciones de las fuentes (Figura 23).

```
create view EG as
   select DesProd, UnidadesNegocio, Clasificacón, PrecioaAbierto as Precio
   From S1
Union
   select S2.NomProd as DesProd, S2.Unidades as UnidadesNegocio,
          S3.Clasificacion, S3.Precio
   from S2, S3
   where S2.NomProd=S3.NomProd
```

Figura 23. Esquema global definido en GAV.

En el método LAV por cada fuente de datos S una vista es definida en términos del esquema global (Figura 24).

```
create view EG1 as
   select DesProd, UnidadesNegocio, Clasificacón, PrecioAbierto as Precio
   From S1
create view EG2 as
   select NomProd as DesProd, Unidades as UnidadesNegocio,
   from S2
Create view EG3 as
   select NomProd as DesProd, clasificacion, precio
   from  S3
```

Figura 24. Esquema global definido en LAV.

Desde el punto de vista del modelado, el método LAV está basado en la idea de que el contenido de cada fuente "s" debe ser representado en términos de una vista "g" en el esquema global. Un caso notable de este tipo es cuando el sistema de integración está basado sobre un modelo de empresa. La idea es efectiva cuando el sistema de integración esta basado sobre un esquema global estable y bien establecido en la organización. El método LAV favorece la

extensibilidad del sistema; agregar una nueva fuente simplemente significa enriquecer el mapeo con una afirmación. Los sistemas presentados en la referencia [34] son ejemplos de sistemas LAV. Information Manifold expresa su esquema global en términos de descripción lógica y adopta el lenguaje conjuntivo como lenguaje de consulta.

Desde el punto de vista del modelado, el método GAV está basado en la idea de que el contenido de cada elemento "g" del esquema global debe ser representado en términos de una vista sobre la fuente, en ese sentido, el mapeo explícitamente dice al sistema cómo recuperar el dato cuando uno quiere evaluar varios elementos del esquema global. Esta idea es eficiente cuando el sistema de integración de datos está basado sobre un conjunto de fuentes estables. Al principio el método GAV favorece el procesamiento de consultas debido a que el sistema sabe como usar las fuentes para recuperar el dato, sin embargo la extensión del sistema con una nueva fuente es un problema; la nueva fuente puede necesitar tener un impacto sobre la definición de varios elementos del esquema global, cuya asociación de vistas necesita ser redefinida. Ejemplos de sistemas que hacen uso del método GAV se muestran en las referencias [35], [36] y [37].

En la referencia [54] se describe la primera implementación de la técnica de integración de datos llamada BAV (del inglés Both As View). Esta técnica tiene el poder expresivo de las técnicas de integración de datos GAV y LAV. Esta técnica fue implementada en el proyecto AutoMed. AutoMed es un sistema para expresar procesos

de transformación e integración de datos en ambientes de bases de datos heterogéneas.

3.3. Los metadatos.

Una definición clásica de metadatos, tomada de Jhon Poole, es la siguiente:

Los metadatos son datos que describen datos, o información acerca de ellos; o bien descripciones de estructuras de información [19].

El almacenemiento y el uso de metadatos permiten y facilitan:

- el manejo de los datos,
- el uso consistente de los datos,
- el entendimiento de los datos, y
- la explotación de volúmenes de información que son accesible en línea.

A pesar del creciente interés en el manejo de metadatos, su objetivo, requisitos y problemas todavía no están suficientemente estudiados. Esto es particularmente cierto en al área de almacenes de datos [18].

3.4. Propuesta.

En este trabajo de tesis se propone el desarrollo de un prototipo para el problema de la integración de datos, basada en el método procedimental, que usa metadatos para registrar la información necesaria para realizar el proceso de transformación de los datos durante la integración. La solución propuesta se ha implementado

como una función dentro del sistema de mantenimiento de almacenes de datos que se ha desarrollado. Para realizar la integración de datos se utilizan tres metadatos (Figura 25) que contienen información que permiten solucionar problemas de inconsistencias de los datos procedentes de las fuentes operacionales. Las inconsistencias que han sido consideradas son: conflictos de formatos, conflictos de operaciones y limpieza moderada de datos; permitiendo de esta manera cargar los datos al almacén sin conflictos. Se usará un SGBD relacional para almacenar los metadatos.

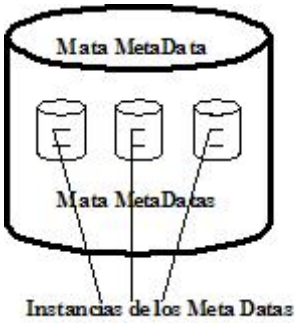


Figura 25. Metadatos usados para la integración de datos.

3.4.1. Definición del dominio.

En la propuesta sólo se contempla la integración de datos de sistemas operacionales que estén soportados por tecnología relacional y/o sistemas de archivos de registros, además se asume que los sistemas operacionales pertenecen a la misma organización.

3.4.2. Estrategia de la solución.

La integración de datos la realiza un proceso que hace uso de metadatos definidos. Con la ayuda del sistema desarrollado el administrador del almacén de datos dispone de un mecanismo general que le permite almacenar los atributos de las fuentes que requieren alguna transformación. Se contempla el uso de tres tipos de metadatos:

a) Formatos: la Tabla 3 muestra el esquema de una tabla que contiene la información de los atributos de los sistemas operacionales que necesitan algún cambio de formato antes de ser enviados al almacén de datos.

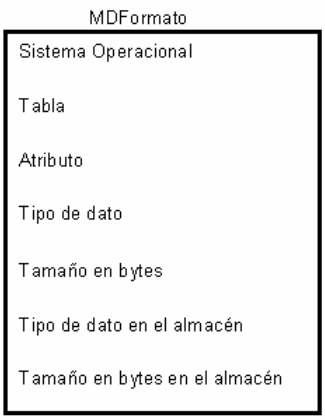


Tabla 3. Esquema del metadato de formato.

Donde:

- Sistema Operacional, Tabla y Atributo: son atributos de tipo alfanumérico que representan respectivamente el nombre del

sistema operacional, el nombre de la tabla y el nombre del atributo que necesita la transformación.

- Tipo de dato: es un atributo de tipo alfanumérico que representa el tipo de datos que tiene el atributo en el sistema operacional.

- Tamaño en bytes: es un atributo de tipo numéricos que representa el número de bytes que requiere el atributo en el sistema operacional.

- Tipo de dato en el almacén: es un atributo de tipo alfanumérico que representa el tipo de datos en el que debe transformarse el atributo del sistema operacional antes de ser incorporado al almacén.

- Tamaño en bytes en el almacén: es un atributo de tipo numérico que representa el número de bytes que requiere el atributo del sistema operacional transformado, para ser incorporado al almacén.

La Tabla 4 muestra un ejemplo de atributos de sistemas operacionales que necesitan una transformación de formato, como se indica en las dos últimas columnas de la tabla.

Sistema Operacional	Tabla	Atributo	Tipo de dato	Tamaño en bytes	Tipo de dato en AD	Tamaño en bytes en AD
Recursos Humanos	Padron	Localidad	Char	2	int	2
Recursos Humanos	Padron	Nombre	Varchar		Char	40
Recursos Humanos	Padron	Estatura	Char	5	Real	4
Nomina	Emision	Departamento	Long int	8	Char	6

Tabla 4. Ejemplo de atributos con cambio de formato.

b) Operaciones: la Tabla 5 muestra el esquema de la tabla que contiene la información de los atributos de los sistemas operacionales que requieren una transformación aritmética previa a su inserción en el almacén de datos, por ejemplo, un cambio de unidad de medida. Para ilustrar este tipo de integración considérese el siguiente ejemplo: sea un atributo A de tipo real, de una fuente de datos que representa una cantidad de dinero en dólares; durante el diseño del almacén se ha decidido que este atributo se almacene en el almacén de datos en euros. El sistema permite especificar la diferencia semántica y tomar en cuenta dicho conocimiento en el momento de insertar el atributo A en el almacén de datos.

MDOperación
Sistema Operacional
Tabla
Atributo
Expresión Aritmética

Tabla 5. Esquema del metadato de operación aritmética.

Donde:

Sistema Operacional, Tabla y Atributo: son atributos de tipo alfanumérico que representan respectivamente el nombre del sistema operacional, el nombre de la tabla y el nombre del atributo que necesita una transformación.

Expresión Aritmética: Es un atributo de tipo alfanumérico que representa la expresión aritmética que se realizará antes de ser incorporado el dato al almacén de datos.

La Tabla 6 muestra un ejemplo de atributos de sistemas operacionales que requieren una transformación aritmética antes de ser incorporados al almacén de datos. La fila primera de la Tabla 6 se interpreta de la siguiente manera: el valor del atributo "sueldo", de la tabla "Emisión", del sistema operacional "Nominas", debe dividirse por 100 y este resultado multiplicarlo por 0.77 antes de insertarse en el almacén de datos.

Sistema Operacional	Tabla	Atributo	Expresión aritmética
Nomina	Emision	Sueldo	Sueldo / 100 * 0.77
Almacen	Artuculos	Existencia	Existencia * ((Unidades +100) +0.75)

Tabla 6. Ejemplo de atributos con transformaciones aritméticas.

c) Limpieza de datos: la Tabla 7 muestra el esquema de la tabla que contiene información sobre los atributos de los sistemas operacionales que necesitan un cambio de valor.

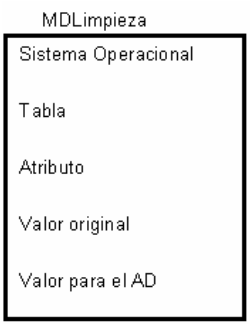


Tabla 7. Esquema del metadato de limpieza de datos.

La Tabla 8 muestra un ejemplo de atributos de sistemas operacionales que requieren que su valor sea analizado y en su caso cambiado. La fila séptima de la Tabla 8 se interpreta de la siguiente manera: si el atributo "Dirección", de la tabla "Padrón", del sistema operacional "Recursos Humanos", contiene el valor *Avenida*, éste debe de ser sustituido en el almacén de datos por el valor *Av*.

Sistema Operacional	Tabla	Atributo	Valor Original	Valor para el AD
Recursos Humanos	Padron	Sexo	H	1
Recursos Humanos	Padron	Sexo	Hombre	1
Recursos Humanos	Padron	Sexo	Masculino	1
Recursos Humanos	Padron	Sexo	M	2
Recursos Humanos	Padron	Sexo	Mujer	2
Recursos Humanos	Padron	Sexo	Femenino	2
Recursos Humanos	Padron	Direccion	Avenida	Av.
Adquisiciones	Compras	Descripcion	Sr	Sr.
Adquisiciones	Compras	Descripcion	senor	Sr.
Adquisiciones	Compras	Descripcion	Señor	Sr.

Tabla 8. Ejemplo de atributos con limpieza de datos.

Con este mecanismo de integración de datos se cumplen los dos requisitos que establecen Levitin y Redman para alcanzar el objetivo de "calidad de los datos" [26]:

- asegurar que el modelo de datos esté bien definido.

- asegurar que los valores de los datos sean exactos.

En la solución propuesta las tablas de metadatos desempeñan un papel fundamental, por lo que se presenta una definición más precisa de ellas.

Sean:

- $S_1, S_2, \ldots, S_n$, n fuentes de datos.
- t^i_j $(1 \leq i \leq n,\ 1 \leq j \leq m_i)$ las tablas de las fuentes de datos. (t^i_j representa la tabla j de la fuente i siendo m_i es el número de tablas de la fuente i).
- $a^{i,j}_k$ $(1 \leq i \leq n,\ 1 \leq j \leq m_i,\ 1 < k \leq l_{i,j})$ los atributos de las tablas de las fuentes de datos. ($a^{i,j}_k$ representa el atributo k de la tabla j de la fuente i siendo $l_{i,j}$ es el número de atributos de la tabla j de la fuente i).

Las tablas de los metadatos MDFormato, MDOperación y MDLimpieza incluyen en su esquema los atributos:

Sistema operacional: $\{S_1, S_2, \ldots, S_n\}$

Tabla: $\bigcup_{\substack{i:1..n \\ j:\ i..m_i}} t^i_j$

Atributo: $\bigcup_{\substack{i:\ 1..n \\ j:\ 1..m_i \\ k:\ 1..l_{i,j}}} a^{i,j}_k$

Ejemplo de una instancia del metadato Operación:
(“Nomina”, “Emision”, “Sueldo”, “Sueldo / 100 + (Sueldo * 0.75)”

La Figura 26 muestra la secuencia necesaria para lograr la transformación de los datos:

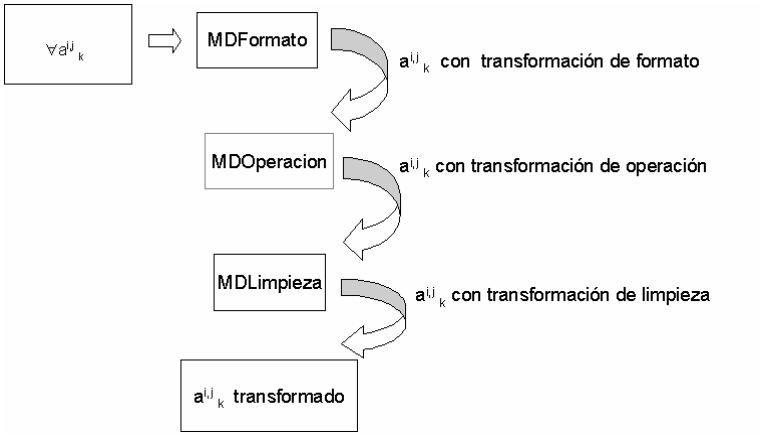


Figura 26. Secuencia de transformación de datos.

3.4.3. Sistema de administración de metadatos.

En este trabajo de tesis se ha desarrollado un sistema para el mantenimiento de almacenes de datos, en el que se incorpora el prototipo desarrollado para la transformación de los datos, y la actualización del almacén de datos. El sistema se ha desarrollado en el lenguaje de programación Visual Basic 6.

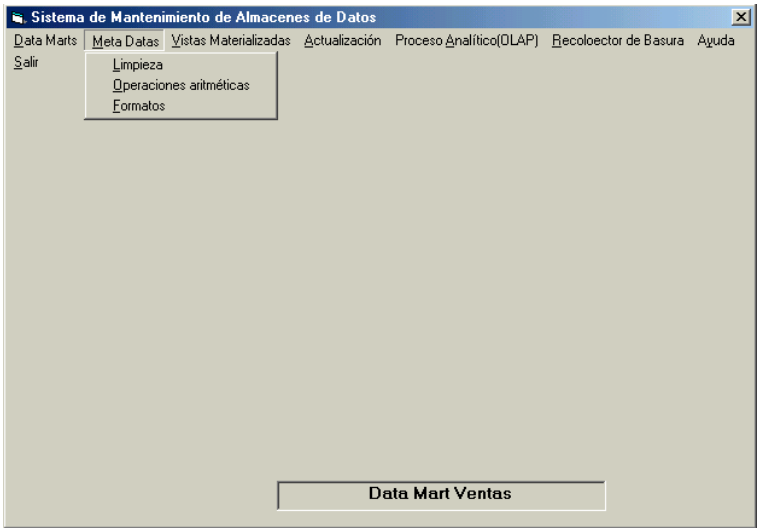


Figura 27. Módulo de administración de metadatos.

El administrador será el responsable de registrar los atributos de los sistemas operacionales que necesitan alguna transformación. El módulo de administración de metadatos muestra la interfaz que se ilustra en las figuras Figura 28, Figura 29 y Figura 30.

La interfaz para la administración de metadatos está dividida en tres secciones (Figura 28):

Sección A: permite seleccionar los atributos de los sistemas operacionales que necesitan alguna transformación.

Sección B: muestra los metadatos que ya están registrados en el sistema.

Sección C: permite indicar los cambios que se deben aplicar a los datos antes de incorporarlo al almacén de datos.

Cuando el administrador seleccione los dos primeros campos de la sección A, se muestran en la sección B, los atributos que ya se encuentran en el sistema, correspondientes al sistema operacional y a la tabla seleccionada. En este momento el administrador tiene dos opciones:

- Si el administrador necesita hacer una modificación, lo selecciona con el botón izquierdo del ratón y el metadato será enviado a la sección C, donde podrá borrarlo (pulsando el botón Eliminar) o hacer los cambios que desee (pulsar el botón Agregar para que se incorpore en la sección B). El botón de Registrar de la sección B, actualiza la instancia del metadato en el sistema.
- Si el administrador necesita incorporar un nuevo atributo al metadato, lo seleccionara en la sección A y en la sección C proporcionara el o los valores que puede tener en el sistema operacional y el valor o acción por el que debe ser sustituido en el Almacén de Datos, debe pulsar el botón Agregar, para que este atributo se incorpore a la sección B y una vez que haya terminado la captura de los atributos del sistema operacional y tabla seleccionada, se pulsa el botón de Registrar que se encuentra en sección B para que se creen las nuevas instancias del metadato en el metadato.

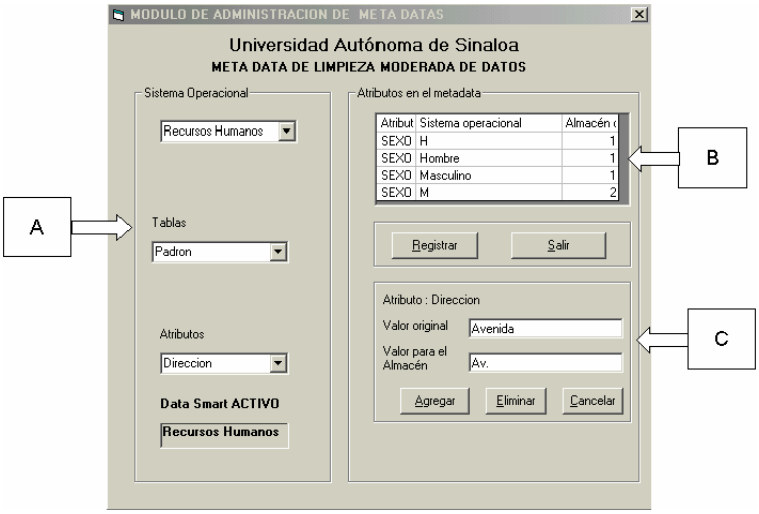


Figura 28. Administración de metadatos de limpieza.

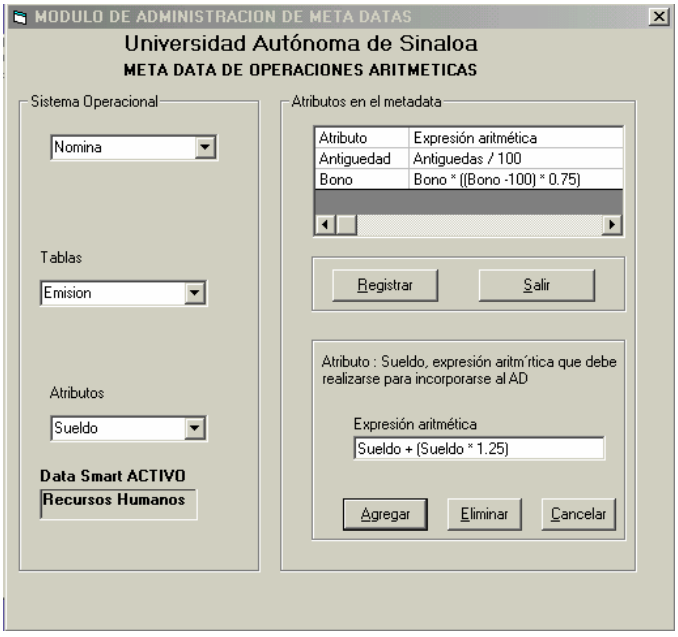


Figura 29. Administración del metadato de operaciones aritméticas.

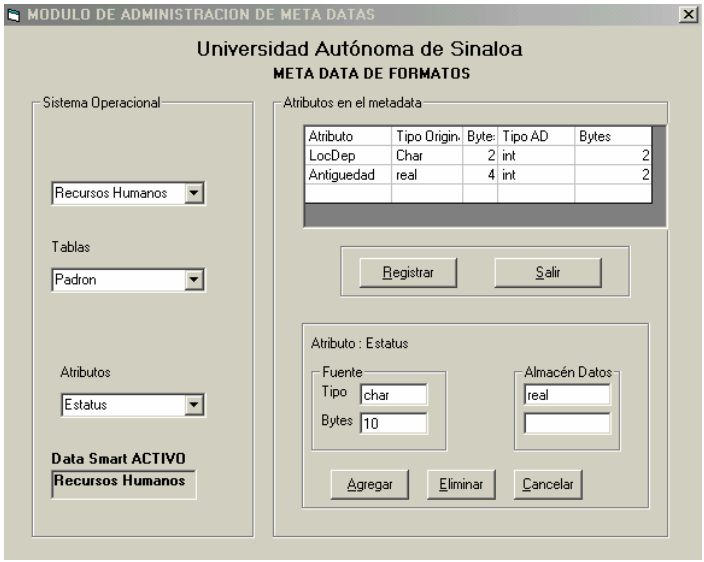


Figura 30. Administración del metadato de formato.

3.4.4. Algoritmo para la transformación de datos.

En la Figura 31 se ilustra el algoritmo que realiza la transformación de datos utilizando los metadatos presentados. Este algoritmo recibe como parámetros el nombre del sistema operacional, el nombre de la tabla, así como la tupla afectada.

Figura 31. Algoritmo para la integración de datos.

Este algoritmo recibe tres parámetros de entrada: (***T***) ***tupla***, (***SO***) ***sitema_operacional*** y (***TA***) ***tabla***. Donde ***T*** contiene la información de la tupla que se ha actualizado o insertado en el sistema operacional, ***SO*** representa el nombre del sistema operacional donde se realizó la actualización y ***TA*** representa el nombre de la tabla donde se realizó la actualización.

Por cada atributo de ***T*** que no forma parte de la clave primaria, se obtiene el ***nombre del atributo (NA)*** y ***el tipo de dato*** **(TD)** que se le definió, y se realizan las siguientes acciones:

1. Limpieza de datos

Usando el metadatos MDLimpieza se verifica si el atributo ***NA*** esta definido como atributo que requiere limpieza, se usa como clave primaria: ***SO, TA y NA***. Si se encuentra definido en el metadatos se realiza un proceso de sustitución de valor: el atributo **NA** de **T** es sustituido por el valor que se especifica en el metadata.

2. Transformación de formatos

Usando el metadatos MDFormatos se verifica si el atributo ***NA*** esta definido como atributo que requiere un cambio de formato, se usa como clave primaria: ***SO, TA y NA***. Si se encuentra definido en el metadatos se convierte el valor del atributo ***NA*** al tipo de datos especificado en el metadatos.

3. Realización de operaciones aritméticas

Usando el metadatos MDOperaciones se verifica si el atributo ***NA*** esta definido como atributo que requiere una operación aritmética, se usa como clave primaria: ***SO, TA y NA***. Si se encuantra definido en el metadatos se obtiene una expresión aritmética a la que se le sustituye el nombre de las variables que contiene con los valores respectivos de la tupla **T**. La expresión aritmética es convertida a expresión aritmética postfija y se evalúa.

Como resultado de este proceso la tupla ***T*** es transformada en la tupla ***T'***.

Capítulo 4. Mantenimiento de almacenes de datos.

La independencia de datos es una de las principales características de la tecnología de bases de datos. Según la arquitectura de niveles propuesta por ANSI/SPARC, deben existir esquemas de la base de datos a tres niveles de abstracción: lógico, físico y externo, y dos tipos de independencia entre estos tres niveles: lógico-físico y externo-lógico. En el modelo relacional, la independencia lógica se consigue por medio del mecanismo de vistas. Una vista (relación derivada) es una relación virtual definida a través de una consulta (instrucción SELECT) sobre relaciones del esquema lógico (relaciones básicas). Un esquema externo en una base de datos relacional se compone de un conjunto de vistas.

Las aplicaciones de las vistas, es decir la posibilidad de definir esquemas externos, son múltiples: asegurar la privacidad de los datos, independizar los programas de aplicación de cambios en el esquema lógico de la base de datos, ocultar la complejidad del esquema lógico a distintos conjuntos de usuarios, etc [40].

En sentido estricto, una vista es virtual (concepto tradicional de vista), pero en algunos casos, por razones de eficiencia, una vista puede ser materializada (Figura 32), lo que significa que su extensión se almacena explícitamente. En este último caso, cuando la base de datos cambia debido a actualizaciones realizadas sobre las relaciones básicas, las vistas materializadas deben ser actualizadas para que estén sincronizadas con las relaciones a partir de las cuales se definen.

Aunque una vista materializada siempre se puede actualizar volviendo a evaluar la expresión relacional que la define, esta solución resulta poco eficiente; por lo que la búsqueda de estrategias para la actualización de vistas materializadas en los sistemas de bases de datos, ha sido siempre un tema de investigación abierto en el campo de las bases de datos [20], [55], [56], [57] y [58]. A este problema se le conoce en la literatura como "mantenimiento de vistas materializadas" [24].

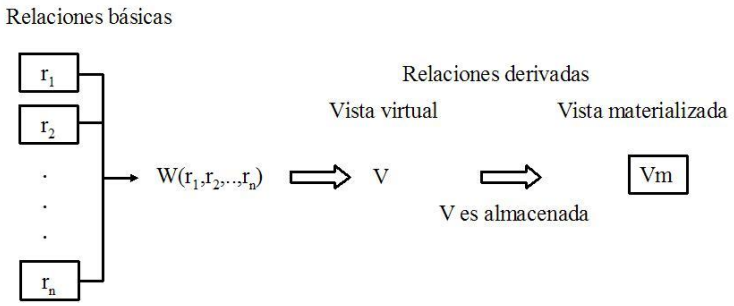


Figura 32 Relaciones derivadas (Vistas)

Desde un punto de vista teórico, y en un contexto relacional, un sistema de almacén de datos puede ser visto como un conjunto de vistas materializadas definidas a partir de las relaciones de las bases de datos operacionales que soportan el almacén. Desde esta perspectiva, el problema del "mantenimiento de un almacén de datos" puede ser visto como un caso particular del problema general del "mantenimiento de vistas materializadas", con lo que todos los resultados y avances de la investigación en esta línea pueden ser aprovechados, adaptándolos al nuevo contexto.

4.1 Concepto de vista.

Definición 1: En una base de datos relacional una vista es una relación virtual definida por una consulta a la base de datos.
La consulta puede expresarse en cualquier lenguaje relacional (Álgebra Relacional, Cálculo Relacional, SQL, …)

En el lenguaje estándar SQL, una vista se define por medio de la cláusula:
CREATE VIEW nombre_vista
[(atributo1, atributo2, …)] AS *instrucción_SELECT*
[WITH CHECK OPTION]

Definición 2: Una vista PSJ (Project-Select-Join) es una vista definida por una expresión relacional de la forma.

$$V = \Pi_{Proj}(\sigma_{Cond}(r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n))$$

donde Π_{Proj} representa el operador proyección sobre el conjunto de atributos *Proj* de las relaciones r_1, r_2, …, r_n; σ_{Cond} representa el operador selección sobre la condición lógica *Cond* definida sobre las relaciones r_1, r_2, …, r_n, y $\bowtie$ representa el operador de concatenación (NATURAL JOIN) entre dos relaciones.

Definición 3: Una vista materializada es una vista cuya extensión se almacena explícitamente.

La ventaja de las vistas materializadas es obvia, ya que para consultar la vista sólo hace falta acceder a los bloques de datos donde se

encuentra almacenada, sin necesidad de volver a evaluar la expresión relacional que la define. El uso de vistas materializadas reduce el coste de la consulta y mejora el rendimiento de los sistemas de bases de datos [27].

Cuando se usan vistas materializadas se debe garantizar a los usuarios la consulta en todo momento de una versión actualizada de la vista, es decir, el resultado de la consulta debe ser equivalente al que se obtendría evaluando sobre la base de datos la expresión relacional que define la vista. Por lo tanto, cada vez que una relación básica es actualizada, las vistas materializadas afectadas por la actualización, deberán ser actualizadas.

La instanciación del problema general del "mantenimiento de vistas materializadas" al caso particular de los sistemas de almacenes de datos, introduce nuevas dificultades. En la Figura 33 se quiere destacar como, en este contexto, una relación del almacén (vista materializada) puede estar definida a partir de relaciones pertenecientes a distintas bases de datos operacionales. Además la vista materializada usualmente estará almacenada en una base de datos distinta a las de las fuentes.

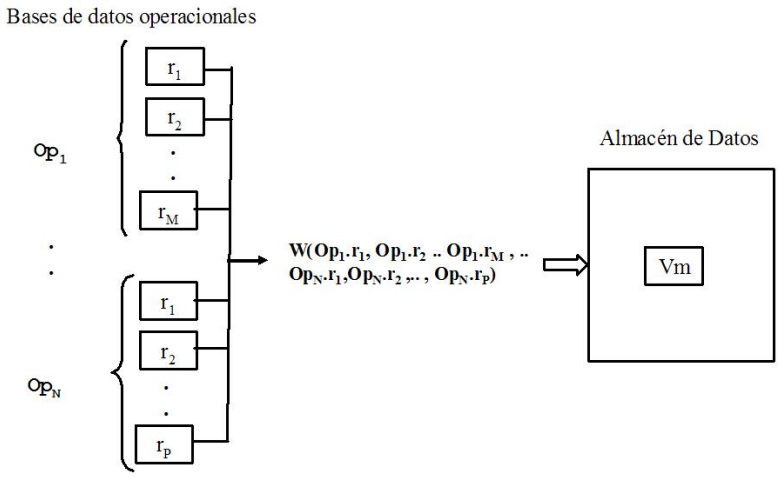


Figura 33. Vista materializada en un almacén de datos.

4.2. Mantenimiento de vistas materializadas.

En la referencia [28] se comenta el hecho de que existen muchos algoritmos para el mantenimiento de vistas materializadas que fallan en un contexto de almacenes de datos. Las causas de esta situación son muy variadas. Las fuentes operacionales pueden ser sistemas antiguos que no sean capaces de manejar vistas o que no conozcan la definición de las vistas materializadas del almacén. En este caso, las fuentes pueden informar al almacén de las actualizaciones que han tenido lugar, pero ignoran los datos adicionales que van a ser necesarios para calcular las actualizaciones que se deben realizar sobre el almacén. En este último caso, cuando la información sobre una actualización llega al almacén, éste puede necesitar consultar datos de algunas de las fuentes. Estas consultas serán evaluadas en las fuentes en un instante de tiempo posterior a la actualización, por lo que el estado de la fuente puede haber cambiado. Este desajuste temporal

entre la actualización de las fuentes y la actualización del almacén, puede conducir a inconsistencias en el mantenimiento de las vistas materializadas. El siguiente ejemplo ilustra este hecho (Figura 34).

Considérese una fuente de datos operacional con las relaciones básicas r1(w,x) y r2(x,y), con extensiones iniciales r1={[1,2]} y r2={[2,3]}, y un almacén de datos con una vista materializada $V = \prod_{w,y} (r1 \bowtie r2)$, de extensión inicial V={[1,3]}, definida a partir de ellas.
Supóngase que tienen lugar dos actualizaciones sucesivas en las relaciones básicas; U1=delete(r1,[1,2]) y U2=delete(r2,[2,3]), las siguientes acciones tendrán lugar.

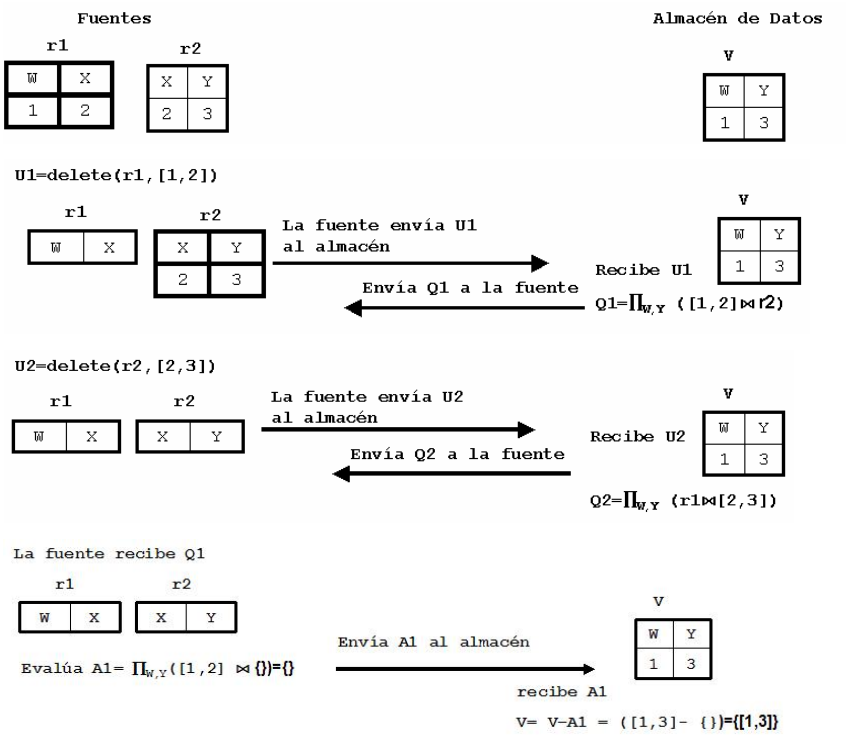
1. La fuente ejecuta la actualización U1=delete(r1,[1,2]) y envía U1 al almacén.
2. El almacén recibe U1 y envía la consulta $Q1=\prod_{w,y} ([1,2] \bowtie r2)$ a la fuente.
3. La fuente ejecuta la actualización U2=delete(r2,[2,3]) y envía U2 al almacén.
4. El almacén recibe U2 y envía $Q2=\prod_{w,y} (r1 \bowtie [2,3])$ a la fuente.
5. La fuente recibe Q1 y evalúa la respuesta $A1 = \prod_{w,y} ([1,2] \bowtie \{\}) = \{\}$.
6. El almacén recibe A1 y actualiza la vista, $V \leftarrow V - A1 \leftarrow [1,3] - \{\} = \{[1,3]\}$[2].
7. La fuente recibe Q2 y evalúa la respuesta $A2 = \prod_{w,y} (\{\} \bowtie [2,3]) = \{\}$.

[2] En la referencia [20] se establece que se utiliza la diferencia ya que la actualización a la relación base es un borrado.

8. El almacén recibe A2 y actualiza la vista, $V \leftarrow V - A2 \leftarrow [1,3] - \{\} = \{[1,3]\}$.

El estado final de la vista es inconsistente ya que las relaciones r1 y r2 están vacías y la vista contiene la tupla [1,3].

En las referencias [28], [45] se muestra que existen algoritmos que encuentran solución al problema anterior haciendo uso de algoritmos de compensación. Sin embargo en la referencia [59] se establece que cuando una vista del almacén está definida sobre más de una base de datos, el problema no puede resolverse con ninguno de los algoritmos mencionados.



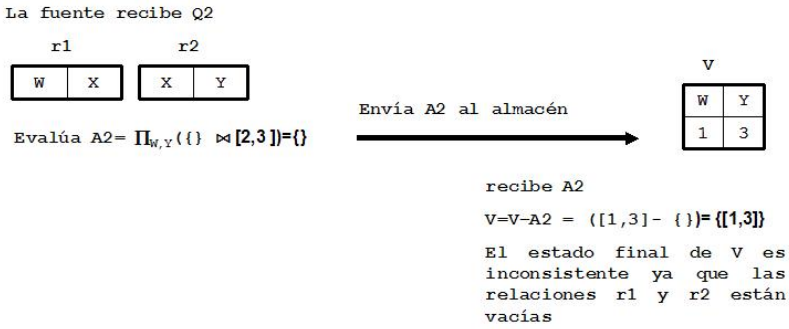


Figura 34. Inconsistencia en el mantenimiento de vistas materializadas.

4.3. Políticas de mantenimiento de vistas materializadas.

En la referencia [24] se establece que los algoritmos de mantenimiento de vistas materializadas pueden clasificarse en función de los siguientes tres criterios: "el cómo", "el cuándo" y "el contexto" en que se realiza el mantenimiento.

4.3.1. Criterio 1: ¿Cómo realizar el mantenimiento?

En la Figura 35 aparecen clasificadas las estrategias en las que se combinan distintas respuestas a la pregunta de *cómo* realizar el mantenimiento de vistas materializadas.

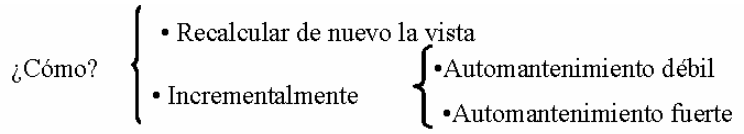


Figura 35. Criterio 1: ¿Cómo actualizar el almacén?

- Recalcular de nuevo la vista

Una vista materializada puede actualizarse, en cualquier instante de tiempo, volviendo a evaluar la expresión relacional que la define y almacenando de nuevo su extensión.

- Incrementalmente

Sea V una vista definida sobre las relaciones básicas R_1, R_2, ..., R_n.

Definición 4: Mantenimiento incremental. La vista materializada V se actualiza incrementalmente, si para cualquier actualización en cualquiera de las relaciones básicas, δR_i, se calcula la actualización de la vista, δV, provocada por δR_i (Figura 36).

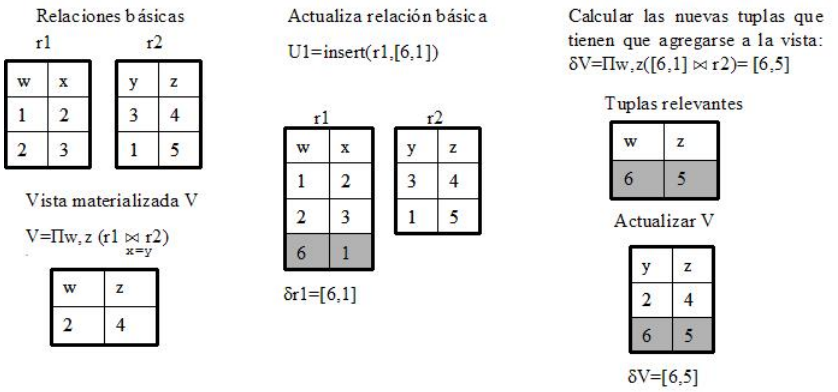


Figura 36. Mantenimiento incremental.

Cuando una vista en el almacén de datos, puede ser mantenida incrementalmente en respuesta a las actualizaciones producidas en las fuentes sin acceder a éstas últimas, se dice que la vista es automantenible**.**

Existen dos casos de automantenimiento: el automantenimiento fuerte y automantenimiento débil.

Definición 5: Automantenimiento fuerte. La vista V es automantenible (fuerte) si, frente a una actualización δR_i, δV puede ser calculada a partir de V y de δR_i. (Figura 37).

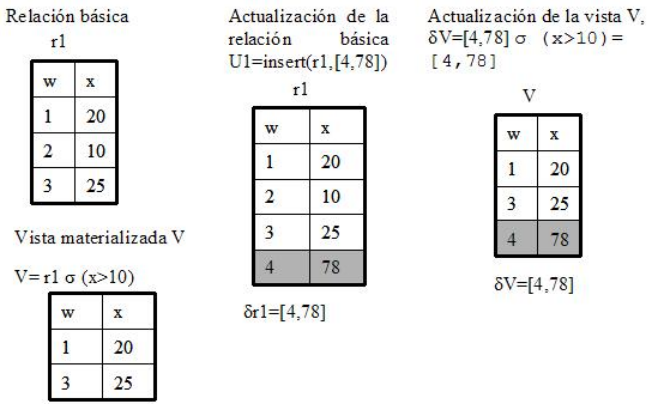


Figura 37. Automantenimiento fuerte.

Definición 6: Automantenimiento débil. La vista V es automantenible (débil) si frente a una actualización δR_i , δV pueden ser calculada a partir de V, δR_i y alguna información adicional mantenida en al almacén de datos. (Figura 38).

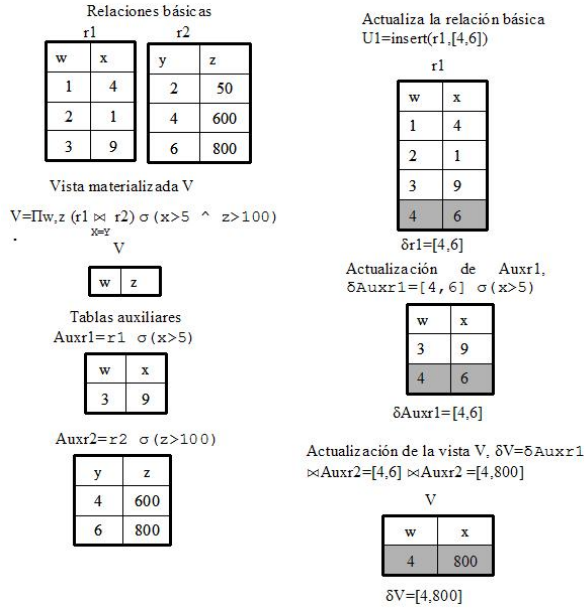


Figura 38. Automantenimiento débil.

Como se observa en los ejemplos, el mantenimiento de una vista requiere usualmente el acceso a datos que no están disponibles en la propia vista. Por ejemplo, en cualquier vista que incluya una concatenación, el mantenimiento de la vista requiere acceso a la base de datos. Ejemplo: para una vista $V = R_{R.A} \bowtie_{S.A} S$, cuando una inserción en la relación R es transmitida con un valor diferente para el atributo A con respecto a las tuplas que ya existen en R, para mantener V es necesario consultar S para descubrir que tuplas de S concatenan con la tupla insertada en la relación R. En el escenario de almacenes de datos acceder a la base de datos significa o bien consultar las fuentes de datos o bien duplicar las relaciones básicas en el almacén [29].

4.3.2. Criterio 2: ¿Cuándo realizar el mantenimiento?

La decisión sobre el instante de tiempo adecuado para realizar el mantenimiento de las vistas define este segundo criterio de clasificación (Figura 39). La actividad de mantenimiento puede ser disparada por diferentes eventos [60]: actualización de la base de datos, consulta de las vistas, petición explícita de mantenimiento o intervalos periódicos de tiempo. En la siguiente figura se muestran los instantes de mantenimiento más comúnmente utilizados.

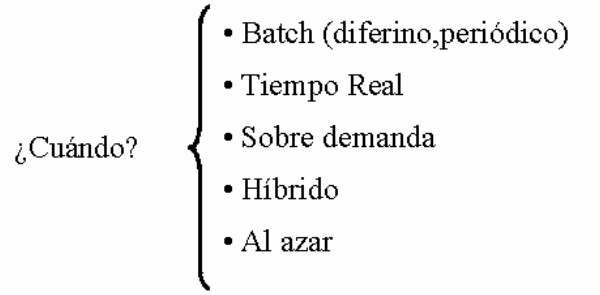


Figura 39. Criterio 2: ¿Cuándo actualizar el almacén?

- Batch (diferido, periódico): el mantenimiento se realiza en intervalos regulares de tiempo (cada hora, cada día, etc.). En este tipo de mantenimiento una vista puede estar temporalmente inconsistente con la base de datos. Este tipo de mantenimiento es comúnmente usado en los sistemas comerciales (Red Brick systems, Syncsort, EMC Enterprise Storage).
- Tiempo real (inmediato): el mantenimiento se realiza cuando se produce la actualización de una relación básica y esta actualización es relevante para las vistas.

- Sobre demanda: el mantenimiento se realiza cuando la vista es consultada. Este tipo de mantenimiento es también un mantenimiento diferido.
- Híbrido: distintos tipos de eventos disparan el mantenimiento. En la literatura se recomienda que este mantenimiento sea combinado con el mantenimiento periódico.
- Al azar: el mantenimiento se realiza en instantes de tiempo al azar.

4.3.3. Criterio 3: Contexto en el que se realiza el mantenimiento.

Uno de los objetivos de las políticas de mantenimiento en sistemas de almacenes de datos es tener el almacén de datos siempre disponible (24 horas al día y 7 días a la semana) para los analistas de la organización. Sin embargo, muchas políticas de mantenimiento no consiguen este objetivo. El tercer criterio de clasificación caracteriza este aspecto del mantenimiento. En la Figura 40 se muestran los contextos en los que se puede realizar el mantenimiento de almacenes de datos.

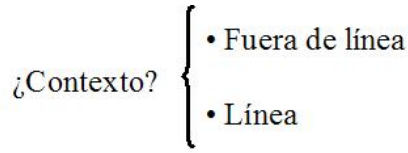


Figura 40. Criterio 3: Contexto en el que se actualiza el almacén.

- Fuera de línea: en este contexto el almacén de datos está inhabilitado a los analistas de la organización ya que el almacén de datos se está actualizando (no pueden realizarse consultas y actualizaciones concurrentemente).

- En línea: en este contexto el almacén de datos siempre está disponible a los analistas de la organización y la actualización del almacén puede realizarse en cualquier momento sin importar que se estén realizando consultas.

4.4. Mantenimiento incremental de vistas materializadas: una revisión de algoritmos.

En la referencia [25] se analiza el problema del mantenimiento incremental de vistas materializadas desde cuatro dimensiones básicas:

- Información: esta dimensión caracteriza la información disponible con la que cuenta el algoritmo de mantenimiento para actualizar las vistas. Es decir, si el algoritmo de mantenimiento tiene acceso a las relaciones básicas o a otras vistas mantenidas en el sistema.
- Modificación: esta dimensión caracteriza el tipo de actualizaciones que puede realizar el algoritmo de mantenimiento de vistas.
- Lenguaje: esta dimensión caracteriza la definición sintáctica de la vista; vistas SPJ (select-project-join), vistas con agregación, vistas con recursión u otros casos.
- Instancia: esta dimensión caracteriza si el algoritmo esta limitado a instancias restringidas de la base de datos o a instancias restringidas de las consultas.

Para ilustrar la dimensión información y la dimensión modificación, considérese la relación Piezas (NoPieza, Precio, Contrato), que contiene el precio para una pieza negociado en cada contrato. Una

pieza puede tener diferente precio bajo cada uno de los contratos realizados. Considérese también la vista materializada.

$PiezasCaras = \Pi_{NoPieza}\, \sigma_{(Precio>1000)}\, Piezas$

La vista contiene los números de pieza cuyo precio sea mayor de mil pesos en algún contrato (la proyección descarta duplicados). Cuando se insertan tuplas en la relación Piezas con un precio por pieza menor a mil pesos, la vista materializada no necesita ser actualizada, por otro lado, si insertamos la tupla <p1, 5000, c15> en la relación Piezas, diferentes algoritmos de mantenimiento de vistas se pueden diseñar dependiendo de la información disponible para determinar si la pieza 'p1' debe ser insertada o no en la vista materializada.

- Sólo la vista materializada está disponible: en este caso, se usa la vista materializada para determinar si existe el NoPieza que se está insertando en la relación básica Piezas. Si existe, no hay cambio en la vista materializada, en caso contrario se inserta en la vista el nuevo NoPieza.
- Sólo la relación básica Piezas está disponible: en este caso se usa la relación básica para verificar si existe alguna tupla con el mismo código y que cumpla la condición establecida para la vista materializada, si existe el cambio no es relevante y la vista no sufre modificación, en caso contrario se inserta en la vista el nuevo NoPieza.

Otro problema con respecto al mantenimiento de la vista PiezasCaras es responder a los borrados que se realizan en la relación Piezas. Supóngase que la tupla <p1,3000,c12> es borrada, 'p1' no se puede

borrar de la vista ya que pueden existir otras tuplas como <p1, 2000, c9> que oblign a que ‘p1’ esté en la vista materializada.

Para ilustrar la dimensión lenguaje y la dimensión instancia considérese la relación Precios (NoProv, NoPieza, Precio) con información sobre la lista de precios de los proveedores, y la vista PiezaProv = $\Pi_{NoPieza}$ Piezas $\bowtie_{(NoPieza=NoPieza)}$ Precios. La vista contiene los NoPieza que son suministrados por al menos un proveedor. Considérese usar solamente PiezaProv para realizar el mantenimiento de la vista en respuesta a la inserción de la tupla <p1, 5000, c15> en Piezas. Si PiezaProv ya contiene ‘p1’, la inserción no afecta a la vista, sin embargo si ‘p1’ no está en la vista, el efecto de la concatenación que define la vista hace imposible que se realice la actualización disponiendo sólo de la vista. Observe que PiezaProv es mantenible si la vista contiene ‘p1’ pero no de otra forma, así que la mantenibilidad de la vista depende de instancias particulares de la base de datos y de la actualización.

En la referencia [25] se presenta una clasificación de los algoritmos utilizados para la actualización de almacenes de datos en base a la dimensión información (información completa, información parcial, vistas automantenible, tiempo real), que serán ilustrados con ejemplos que muestran como se realiza el mantenimiento del almacén de datos.

4.4.1. Uso de información completa.

Todas las relaciones básicas y vistas materializadas están disponibles durante el proceso de mantenimiento.

Algoritmo "Counting" [78]

Considérese la relación básica r(X,V), tal que r(a,b) es cierto si existe un enlace desde el nodo origen "a" hasta el nodo destino "b". Sea la vista Link tal que Link(c,d) es cierto si "c" conecta con "d" usando dos enlaces, haciendo uso de un nodo intermedio.

Supóngase que la extensión de r es r = {[a,b], [b,c], [b,e], [a,d], [d,c]}, la extensión de la vista sería Link={[a,c,2],[a,e,1]}.

Link se calcula como:

$$Link(X,Y) = \Pi_{X,y} (r(X,V) \underset{V=W}{\bowtie} r(W,Y))$$

Link contiene un contador que indica el número de derivaciones para llegar de a→,c y de a→,e. El algoritmo counting puede manejar o no tuplas duplicadas.

Sea un conjunto de tuplas △(r) a ser insertadas en r. Las inserciones que necesitan ser realizadas a la vista Link, pueden ser calculados usando la técnica de diferenciación que se ilustra a continuación.

$$\triangle(Link) = \Pi x,y ((\triangle (r) (X,V) \underset{V=W}{\bowtie} r(W,Y)) \sqcup$$

$$\Pi x,y((r (X,V) \underset{V=W}{\bowtie} \triangle (r) (W,Y)) \sqcup$$

$$\Pi x,y ((\triangle(r) (X,V) \underset{V=W}{\bowtie} \triangle (r) (W,Y)))$$

Un programa T△ usa los cambios hechos a las relaciones básicas y produce como salida el conjunto de cambios △ ($Link_1$), ..., △ ($Link_K$), que necesitan ser hechos a las vistas. En el conjunto de cambios, las inserciones son representadas con contador positivo y las supresiones son representadas con contadores negativos. El valor contador de cada tupla es almacenado en la vista materializada y la nueva vista materializada se obtiene aplicando los cambios △ ($Link_1$),..., △ ($Link_K$) a las vistas almacenadas $Link_1$,...,$Link_K$. Los contadores con valores positivos son sumados y los valores con contadores negativos son restados. Una tupla con contador cero es borrada.

Ejemplo: △ (r) contiene la tupla {[d,c]} a ser insertada en r, en base a esto obtenemos △ (Link).

$$\triangle(Link) = \Pi_{x,y}((\triangle(r)(X,V) \underset{V=W}{\bowtie} r(W,Y)) \sqcup$$

d c a b
d c b c
d c b e concatenación ={}
d c a d
d c d c

$$\Pi_{x,y}((r\ (X,V) \underset{V=W}{\bowtie} \triangle(r)(W,Y)) \sqcup$$

a b d c
b c d c
b e d c
a d d c concatenación={[a,c]}
d c d c

$$\Pi_{x,y}((\triangle(r)\ (X,V) \underset{V=W}{\bowtie} \triangle(r)(W,Y)))$$

d c d c concatenación={}

Resultado de la unión es {[a,c]}, esto indica que la vista Link debe modificarse :

- Si la tupla [(a,c)] existe, incrementar el contador en uno.
- Si la tupla [(a,c)] no existe, insertarla con contador 1.

vista actualizada Link={[a,c,3],[a,e,1]}

Supóngase que △(r) contiene la tupla {[e,f]}, en base a esta inserción se obtiene △(Link).

$$\triangle(\text{Link}) = \Pi_{x,y}\left((\triangle(r)(X,V) \underset{V=W}{\bowtie} r(W,Y)) \sqcup\right)$$

e f a b
e f b c
e f b e concatenación ={}
e f a d
e f d c

$$\Pi_{x,y}\left((r(X,V) \underset{V=W}{\bowtie} \triangle(r)(W,Y)) \sqcup\right)$$

a b e f
b c e f
b e e f
a d e f concatenación={[b,f]}
d c e f

$$\Pi_{x,y}\left((\triangle(r)(X,V) \underset{V=W}{\bowtie} \triangle(r)(W,Y))\right)$$

e f e f concatenación={}

Resultado de la unión es {[b,f]}, esto indica que Link debe ser modificada :

- Si la tupla {[b,f]} existe, debe incrementar el contador en uno.
- Si la tupla {[b,f]} no existe, insertarla con contador 1.

vista actualizada Link={[a,c,3],[a,e,1],[b,f,1]}

Supóngase que △ (r) contiene {[d,c]} que será eliminada de la relación r. Obtenemos △ (r).

$\triangle(\text{Link}) = \Pi x,y\ ((\triangle\ (r\ )\ (X,V) \underset{V=W}{\bowtie} r(W,Y)) \sqcup$

d c a b
d c b c
d c b e concatenación ={}
d c a d
d c d c

$\Pi x,y((r\ \ (X,V) \underset{V=W}{\bowtie} \triangle\ (r\ )\ (W,Y)) \sqcup$

a b d c
b c d c
b e d c
a d d c concatenación={[a,c]}
d c d c

$\Pi x,y\ ((\triangle\ (r)\ \ (X,V) \underset{V=W}{\bowtie} \triangle\ (r\ )\ (W,Y)))$

d c d c concatenación={}

Resultado de la unión es {[a,c]} esto indica que la tupla [a,c] en Link debe ser modificada, decrementando el contador en uno y si el valor del contador queda en cero, la tupla debe ser borrado de la vista.

vista actualizada Link={[a,c,2],[a,e,1],[b,f,1]}

4.4.2. Uso de información parcial.

El mantenimiento de vistas puede realizarse usando un subconjunto de las relaciones básicas involucradas en la vista.

Sin uso de información: consulta independiente de la actualización.

Hay casos en que una modificación de las relaciones básicas no causa efecto en la vista. Esto se conoce como "consulta independiente de la actualización" o "actualización irrelevante". Todos los algoritmos proporcionan una verificación que determina si una actualización particular será irrelevante. Si la prueba es exitosa, entonces el estado de la vista permanece sin alterarse, sin embargo, si la prueba falla, algún otro algoritmo tiene que ser usado para realizar el mantenimiento de la vista. Uno de los algoritmos que existe en la literatura para detectar tuplas irrelevantes para vistas SPJ se describe a continuación y se puede consultar en la referencia [20].

Algoritmo "Efficiently Updating Materialized View" [20].

Considérese la vista definida por la expresión $V= \pi_X (\sigma C(Y) (r_1 \bowtie r_2 \bowtie .. \bowtie r_P))$, donde C(Y) es una expresión booleana. X y Y son conjuntos de variables denotando el nombre de los atributos que pertenece a alguna de las relaciones $r_1, r_2, , r_P$. Suponga que la tupla

t=(a1,a2,...,aq) es insertado (o borrado) de la relación r_k, definido sobre el esquema R_k, si la condición C(Y) modificada puede mostrarse que no es satisfecha a pesar del estado de la base de datos, entonces la inserción o borrado de la tupla t sobre la relación r_k no tiene efecto en la vista V (detectó una inserción irrelevante para la vista V).

Ejemplo: considérense dos relaciones r y s, definidas como r={A,B} y s={C,D} y una vista V definida como:

$$V=\pi_{A,D}\ (\ \sigma_{(A<10)\ ^\wedge\ (C>5)\ ^\wedge\ (B=C)}\ (r \bowtie s)\ \ )$$

En base a la definición de la vista V, la función C(Y) modificada es la siguiente:

C(A,B,C)=(A<10)^(C>5)^(B=C)

Ahora considere el siguiente estado inicial de las relaciones r={(1,2),(5,10),(12,15)}, s={(2,10),(10,20)} y la vista V={(5,20)}.

Suponga que la tupla (9,10) es insertada en la relación r, sustituyendo el valor (9,10) en las variables A y B en C(A,B,C)=(9<10)^(C>5)^(10=C). La condición de selección C(9,10,C) es satisfecha, lo que significa, que hay una instancia de las relaciones R y S conteniendo la tupla (9,10) y (10,δ), para algún valor de δ, tal que C(9,10, δ)=verdadero. Por lo tanto, insertar la tupla (9,10), en la relación r es **relevante** para la vista V. Ahora suponga que la tupla (11,10) es insertada en la relación r. Después de sustituir los valores (11,10) en las vriables A y B en C(A,B,C) obtenemos:

C(11,10,C)=(11<10) ^(C>5) ^ (10=C)

Se puede decir que C es insatisfecha a pesar del estado de la base de datos. Por lo tanto insertar la tupla (11,10) en la relación r, es **irrelevante** para la vista V.

El mismo argumento aplica para el borrado de tuplas, esto es, sustituir el valor de la tupla borrada en la condición de selección, si la condición de selección es insatisfecha a pesar del estado de la base de datos, entonces, la tupla borrada es **irrelevante**. Por otro lado, si sustituimos el valor de la tupla borrada, en la condición de selección y si esta condición es satisfecha, la tupla borrada debe eliminarse de la vista.

4.4.3. Vistas auto-mantenibles.

Las vistas pueden ser mantenidas usando sólamente la vista materializada y los cambios realizados en las relaciones básicas.

Algoritmo: "Self Maintenance of Multiple Views in Data Warehousing" [6]

Considérese un conjunto de m vistas materializadas V = $\{MV_1, MV_2, MV_3, ..., MV_M\}$ en el almacén de datos definidas sobre un conjunto de relaciones básicas $R=\{R_1,R_2, ...,R_N\}$.

Para realizar auto-mantenimiento de vistas múltiples, un conjunto de vistas auxiliares (AV) son almacenadas en el almacén de datos junto con el conjunto de MVs, de tal forma que juntas MV $\cup$ AV es auto-mantenible.

El esquema de mantenimiento de la vista (plan de vistas) se muestra en la Figura 41.

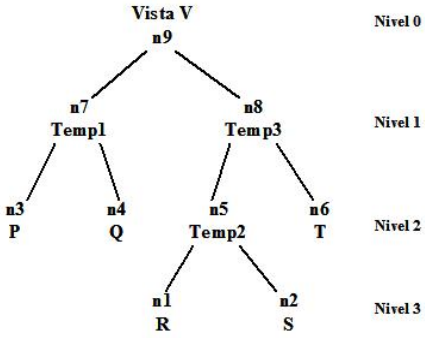


Figura 41. Esquema de mantenimiento de vistas.

El plan de vistas consiste de las relaciones básicas y los nodos intermedios. El nodo raíz del árbol es la vista materializada que necesita ser mantenida en respuesta a la actualización realizada en los nodos hojas (pueden ser relaciones básicas o conjunto de resultados intermedios). El nodo raíz del subárbol es un conjunto de resultados intermedios excepto cuando este es la raíz de la consulta completa del árbol.

El esquema de mantenimiento divide el árbol de consulta en subárboles individuales, se crea una AV para cada nodo hoja en el subárbol. El nodo raíz del árbol almacena tuplas en una MV. La vista V en el nivel 0 puede ser descrita en términos de los nodos del nivel 1 como Temp1 ⋈ Temp3. Algunas tuplas de Temp1 y Temp3 que no concatenan con la vista V son almacenadas en la respectiva AV para temp1 y temp3. Similarmente, Temp3 puede ser expresada como Temp2 ⋈ T, de esta manera las AV del nivel 2 almacenan tuplas de

Temp2 y T que no concatenan con Temp3. Similarmente, las AV_S para las relaciones básicas podrán almacenar tuplas que no concatenen con el siguiente nodo de nivel superior. Así, en cada nodo hoja de los subárboles, se tiene una AV que almacenan ciertas tuplas que pueden en el futuro estar el la MV del nodo raíz.

Cuando una actualización δ es puesta en la relación básica (por decir R), La ruta rutas que δ puede encontrar hacia el nodo raíz es:

$$a)\ \delta V = ((\delta \bowtie AV_S) \bowtie AV_T) \bowtie AV_{Temp1}$$

Ejemplo: Considere las relaciones básicas R(A,B), S(C,D), AV_R(A,B), AV_S(C,D) y la vista materializada MV= $R \bowtie (_{B=C}) S$ y suponga que los siguientes eventos se presentan. Inicialmente la relación R={[1,2],[3,4]}, S={[2,3],[5,6]}, MV={[1,2,2,3]),

Calculando la vista auxiliar

AV_R contiene tuplas que no están en Π_R (MV)

$AV_R = R - \Pi_{A,B} (R \bowtie MV)$

AV_R = {[1,2],[3,4]} - $\Pi_{A,B}$ ({[1,2],[3,4]}⋈ {[1,2,2,3]}

{[1,2],[3,4]} - {[1,2]}

AV_R={[3,4]}

Calculando la vista auxiliar

AV_S contiene tuplas que no están en Π_S (MV)

$AV_S = S - \Pi_{C,D} (S \bowtie MV)$

AV_S = {[2,3],[5,6]} - $\Pi_{C,D,}$ ({[2,3],[5,6]}⋈ {[1,2,2,3]}

{[2,3],[5,6]} - {[2,3]}

AV_S={[5,6]}

$\triangle_R$ [4,5] representa una inserción en la relación R

$T1 = \Pi_{\triangle_{R.A,}\ \triangle_{R.B,}\ MV^{OLD}.D}(MV^{OLD} \bowtie (_{MV^{OLD}.B=\triangle_{R.A}}) \triangle R)$

{[1,2,2,3]} 2=5 [4,5]

T1={}

X= $\triangle_R - \Pi_{A,B}$ (T1)

[4,5]-{}

X={[4,5]}

T2=$AV_S \bowtie (_{AVS.C=X.B})$ X

{[5,6]} 5=5 {[4,5]}

T2={[4,5,5,6]}

Calcula actualización de la vista materializada

$\triangle_{MV}$ = T1 ⊔ T2

{} {[4,5,5,6]}

$\triangle_{MV}$ ={[4,5,5,6]}

Calcula modificación a las vistas auxiliares

$\triangle_{AVR}$=X - $\Pi_{A,B}$ (T2)

{[4,5]}- {[4,5]}

$\triangle_{AVR}$={}

Calcula como queda la nueva vista materializada

MV^{NEW}=MV^{OLD} ⊔ △MV

{[1,2,2,3]} {[4,5,5,6]}

MV^{NEW}={[1,2,2,3],[4,5,5,6]}

Calcula la nueva vista auxiliar de la relación R

AV_R^{NEW}=AV_R^{OLD} ⊔ $\triangle AV_R$

{[3,4]} {}

AV_R^{NEW}={[3,4]}

Calcula la nueva vista auxiliar de la relación S

AV_S^{NEW}=AV_S^{OLD} - $\Pi_{B,D}$ (△MV)

{[5,6]} {[4,5,5,6]}

{[5,6]} - {[5,6]}

AV_S^{NEW}={}

∇R [1,2] representa un borrado en la relación R

Cambio a la vista auxiliar R

$\nabla_{AVR}=$

$\Pi\, \nabla_{R.A,}\, \nabla_{R.B}\, (AV_R^{OLD} \bowtie (\,_{(AV}{}^{OLD}{}_{R.A\,=\,\nabla R.A)} \wedge\, _{(AV}{}^{OLD}{}_{R.B\,=\,\nabla R.B)}\,)\, \nabla R)$

{[3,4]} 3=1 4=2 [1,2]

$\nabla_{AVR}=\{\}$

$X=\nabla R - \nabla_{AVR}$

[1,2] {}

X={[1,2]}

Cambio a la vista materializada

$\nabla_{MV}= MV^{OLD} \bowtie\, _{\{(MV}{}^{OLD}{}_{\cdot A=\nabla R.A)\,\wedge\, (MV}{}^{OLD}{}_{\cdot B=\nabla R.B)\}}\, X$

{[1,2,2,3] 1=1 2=2 {[1,2]}

[4,5,5,6]}

$\nabla_{MV}=\{[1,2,2,3]\}$

Calcula la nueva vista materializada

$MV^{NEW}=MV^{OLD} - \nabla_{MV}$

{[1,2,2,3] {[1,2,2,3]}

[4,5,5,6]}

$MV^{NEW}=\{[4,5,5,6]\}$

Calcula la nueva vista auxiliar R

$AV^{NEW}{}_R=AV^{OLD}{}_R - \nabla_{AVR}$

{[3,4]} - {}

$AV^{NEW}{}_R=\{[3,4]\}$

Calculando la nueva vista auxiliar S

$\nabla_{AVS}= \Pi_S(\nabla_{MV}) - \Pi_S(MV)$

{[2,3]} {[5,6]}

$\nabla_{AVS}=\{[2,3]\}$

$AV^{NEW}_S = AV_S^{OLD} \sqcup \nabla_{AVS}$

$\{\}$ $\{[2,3]\}$

$AV^{NEW}_S = \{[2,3]\}$

4.4.4. Mantenimiento en tiempo real.

La actualización de vistas materializadas en tiempo real se realiza a partir del establecimiento de un canal de comunicación entre el almacén de datos y los sistemas operacionales, de manera que cada vez que se produce una actualización en las relaciones básicas del sistema operacional, dicho sistema informa al almacén de datos de los cambios enviando las actualizaciones necesarias para mantener la consistencia con las vistas materializadas.

4.4.4.1. Algoritmo "Eager Compentsating Algorithm (ECA)" [28].

Es un algoritmo de mantenimiento incremental, basado en el algoritmo Efficiently Updating Materialized View, ECA cuenta con la característica de anticiparse a las anomalías que surgen debido al desacoplamiento entre las actualizaciones de las relaciones básicas y la actualización de las vistas.

Definición: Ui denota la actualización

Definición: el conjunto de consultas que no han recibido respuestas UQS, es el conjunto de todas las consultas que el almacén de datos ha enviado hacia alguna fuente pero aún no han recibido respuesta.

Definición: Q_i denota la consulta enviada por el almacén de datos, en respuesta a la actualización U_i.

Definición: V<U> denota la expresión vista V con la tupla U sustituida por las relaciones U´s.

Resultado = 0

W_up_i : recibe U_i ;

$Q_i = V<U_i> - \Sigma Q_{j \in UQS} Q_j <U_i>$

envía Q_i a la fuente

Dispara un evento S_qu_i en la fuente

W_ANS_i : recibe A_i ;

Resultado=Resultado + A_i ;

if UQS = 0 Then

{

MV = MV + Resultado;

Resultado=0

}

Considérense las relaciones básicas r1(W,X), r2(X,Y), r3(Y,Z) y la vista $V= \Pi_W (r1 \bowtie r2 \bowtie r3)$ y supóngase que los siguientes eventos tienen lugar. Inicialmente la vista materializada es V={}. Las relaciones r2 y r3 están vacías y r1={[1,2]}. Las actualizaciones se realizan primero en las relaciones básicas antes de que las consultas sean respondidas.

1. El DW recibe U1=insert(r1,[4,2])

DW envía $Q1=V<U1>= \Pi_W ([4,2] \bowtie r2 \bowtie r3)$

Q1 se encuentra pendiente de respuesta (PR)

2. El DW recibe U2=insert(r3,[5,3])

UQS[3]={Q1}

DW envía Q2=V<U2>-Q1<U2>

$Q2= \Pi_W (r1 \bowtie r2 \bowtie [5,3]) - \Pi_W ([4,2] \bowtie r2 \bowtie [5,3])$ (PR)

[3] Conjunto de consultas sin obtener respuesta

3. El DW recibe U3=insert(r2,[2,5])

UQS = {Q1, Q2 }
Q3=V<U3> - Q1<U3> - Q2<U3> =

Π_W (r1 ⋈ [2,5] ⋈ r3) - Π_W ([4,2] ⋈ [2,5] ⋈ r3) - Π_W ((r1-[4,2]) ⋈ [2,5] ⋈ [5,3])

4. El DW recibe A1={[4]}, resultado de la evaluación de Q1:

A1= Π_W ([4,2] ⋈ {[2,5]} ⋈ {[5,3]})

[{4, 2, 5, 3]} proyección w A1={[4]}

Resultado = {} + {[4]}={[4]}, UQS={Q2,Q3}

5. El DW recibe A2={[1]}, resultado de la evaluación de Q2:

A2=

Π_W ({[1,2],[4,2]} ⋈ {[2,5]} ⋈ [5,3]) - Π_W ([4,2] ⋈ {[2,5]} ⋈ [5,3])

{[1, 2, 5, 3] {[4, 2, 5, 3]}

[4, 2, 5, 3]}dif={[1, 2, 5, 3]}, proyeccion w

A2= Π_W dif = {[1]}

resultado=({[4]}+{[1]}={[1],[4]}, UQS={Q3}

6. El DW recibe A3 = {}, resultado de la evaluación de Q3:

A3=

Π_W ({[1,2],[4,2]}⋈ [2,5]⋈ {[5,3]}) - Π_W ([4,2], ⋈ {[2,5]} ⋈ {[5,3]}) –

{[1, 2, 5, 3] {[4, 2, 5, 3]}

[4, 2, 5, 3]}

{[1, 2, 5, 3]}

(({[1,2],[4,2]} – [4,2]) ⋈ [2,5] ⋈ [5,3])) ↓

{[1,2]} dif={}, A3=Πw dif={}

{[1, 2, 5,3]} ↑

{[1, 2 ,5, 3]}

El almacén de datos actualiza la vista: MV={} + Resultado=([1],[4])

Ejemplo de borrado con el algoritmo ECA

Supóngase que se tienen las relaciones básicas r1(W,X) y r2(X,Y) y la vista $V= \Pi_W (r1 \bowtie r2)$. r1={[1,2],[4,2]}, r2={[2,3]} y V={[1],[4]}.

1) El DW recibe U1=delete(r1,[4,2])
   DW envía $Q1=V<U1> = \Pi_W ((-[4,2]) \bowtie r2)$
2) El DW recibe U2=delete(r2,[2,3])
   UQS = {Q1}
   DW envía Q2=V(U2)-Q1(U2)
   $Q2= \Pi_W (r1 \bowtie (-[2,3])) - \Pi_W (-[4,2] \bowtie (-[2,3]))$
   $Q2= - \Pi_W (r1 \bowtie [2,3]) - \Pi_W ([4,2] \bowtie [2,3])$
3) El DW recibe A1={}, resultado de la evaluación de Q1:
   $A1= \Pi_W (-[4,2] \bowtie \{\})=\{\}$
   Resultado=Resultado+A1= {}+{}={}
4) El DW recibe A2=(-[4],-[1]), resultado de la evaluación de Q2

$Q2= - \Pi_W (\{[1,2], [4,2]\} \bowtie [2,3]) - \Pi_W ([4,2] \bowtie [2,3])$

{[1, 2, 3], {[4, 2, 3]}
[4, 2, 3]}

{-[1],-[4]} – {[4]}={-[1],-[4]}

Resultado= Resultado+A2={}+ {-[4],-[1]}={-[4],-[1]}

MV= ([1],[4])+RESULTADO={}

Ejemplo de *inserción* y *borrado* con el algoritmo ECA

Supóngase que se tienen las relaciones básicas r1(W,X) y r2(X,Y) y la vista $V= \Pi_W (r1 \bowtie r2)$. r1={[1,2],[4,2]}, r2={} y V={}.

1. El DW recibe U1=delete(r1,[4,2]))

   DW envía Q1=V<U1> = Π_W ((-[4,2]) $\bowtie$ r2)

   = - Π_W (([4,2]) $\bowtie$ r2)

2. El DW recibe U2=insert(r2,[2,3])

   UQS=[Q1}

   Q2=V<U2> - Q1<U2>

   Q2 = Π_W (r1 $\bowtie$ [2,3]) - Π_W ((-[4,2]) $\bowtie$ [2,3])

   Q2 = Π_W (r1 $\bowtie$ [2,3]) + Π_W (([4,2]) $\bowtie$ [2,3])

3. El DW recibe A1=-[4], resultado de la evaluación de Q1:

   A1= - Π_W ([4,2] $\bowtie$ [2,3])= {-[4]}

   Resultado={}+{-[4]}= {-[4]}

4. El DW recibe A2={[1],[4]}

   A2 = Π_W (r1 $\bowtie$ [2,3]) + Π_W (([4,2]) $\bowtie$ [2,3])

   A2 = Π_W ({(1,2)} $\bowtie$ [2,3]) + Π_W (([4,2]) $\bowtie$ [2,3])

   {[1,2,3]} [4,2,3]

   {[1] } + { [4]}= {[1],[4]}

   Resultado= {-[4]}+{[1],[4]}={[1]}

   MV=MV+Resultado = {}+{[1]}={[1]}.

4.4.4.2. Algoritmo “Strobe” [45].

El algoritmo Strobe procesa las actualizaciones en el orden que lleguen, realizando consultas a la fuentes cuando sea necesario. Sin embargo, las actualizaciones no son realizadas inmediatamente sobre la vista materializada; es decir, se generan una lista de acciones (AL) a ser realizadas sobre la vista. La vista materializada se actualiza solamente cuando se esta seguro que aplicando todas las acciones que se encuentran en AL, la vista será llevada a un estado consistente. Esto

ocurre cuando no hay consultas excepcionales y todas las actualizaciones recibidas han sido procesadas. Cuando el almacén de datos recibe una operación de borrado, éste genera una acción delete para la tupla correspondiente (unificando el valor de la clave) en la vista materializada. Cuando una inserción llega, el almacén de datos puede necesitar generar y procesar una consulta, usando el procedimiento source_evaluate(). Mientras una consulta Q está siendo respondida por las fuentes, otras actualizaciones pueden llegar al almacén de datos y las respuestas obtenidas puden omitir su efecto, Para compensar, se mantiene un conjunto de actualizaciones que se presentan mientras la consulta es procesada. Después de que las consultas son respondidas son compensadas y una acción de inserción generada para MV es colocada en AL.

Definición: el conjunto de consultas que no han recibido respuestas UQS, es el conjunto de todas las consultas que el almacén de datos ha enviado hacia alguna fuente pero aún no recibe respuesta.

Definición: por cada consulta Q en UQS, pending(Q) es el conjunto de todas las tuplas que han sido borradas por cualquiera de las fuentes.

Definición: Q_i denota la consulta enviada por el almacén de datos en respuesta a la actualización (inserción) U_i.

Definición: la operación key_delete(R,U_i) elimina de la relación R las tuplas cuyo atributo llave tenga el mismo valor de U_i.

Definición: V<U> denota la expresión vista V con la tupla U sustituida por las relaciones U´s.

Algoritmo Strobe

En cada fuente:

- Después de ejecutar la actualización U_i, envía U_i al almacén de datos.
- Sobre el recipiente de consultas Q_i, calcula la respuesta A_i, sobre ss[x] (estado actual de la fuente) y envía A_i al almacén de datos.

En el almacén de datos

- Inicialmente AL=<>, conjunto vació.
- Sobre recipiente de actualización U_i:
    - if U_i es ELIMINAR
        - $\forall Q_j \in$ UQS agregar pending (Q_j)
        - agregar key_delete(MV,U_i) a AL
    - if U_i es INSERTAR
        - Q_i=V<U_i> y asignar pending(Q_i)=0
        - A_i=source_evaluated(Q_i)
        - $\forall U_j \in$ pending(Q_i), aplique key_delete(A_i,U_j)
        - agregar insert(MV,A_i) a AL

- Cuando UQS=0 aplicar AL a MV como una transacción sencilla, sin agregar tuplas duplicadasa MV, inicalice AL=<>.

Considérense las relaciones básicas r1(W,X), r2(X,Y), r3(Y,Z) y la vista v= π_W (r1 ⋈ r2 ⋈ r3) y supónngase que los siguientes eventos se presentan. Inicialmente la vista materializada es V={}. La relación

r2 ={}, r1={[1,2]}, y r3={[3,4]}. Las actualizaciones se realizan primero en las relaciones básicas antes de que las consultas sean respondidas.

- El almacén de datos recibe U_1=insert(r2,[2,3])

  Q_1=r1 ⋈ [2,3] ⋈ r3 pending(Q_1)=0
  A_1=evaluar(Q_1)
  Q_1^1= r1 ⋈ [2,3]={[1,2]} ⋈ [2,3] = {[1,2,3]}
  A_1^1 = {[1,2,3]}
  Q_1^2 = A_1^1 ⋈ r3

- El almacén de datos recibe U2=insert(r3,[3,5])
  Q_2=r1 ⋈ [2,3] ⋈ [3,5] pending(Q_2)=0
  A_2 = evaluar(Q_2)
  Q_2^1= r1 ⋈ r2={[1,2]} ⋈ {[2,3]} = {[1,2,3]}
  A_2^1 = {[1,2,3]}
  Q_2^2 = A_2^1 ⋈ [3,5]

- El almacén de datos recibe U3=delete(r3,[3,4])
  Q_1 no ha respondido, agrega pending(Q_1, delete(r3,[3,4])
  Q_2 no ha respondido, agrega pending(Q_2, delete(r3,[3,4])
  agrega key_delete(MV,delete(r3,[3,4])
  A_1^2= {[1,2,3]} ⋈ [3,4]={[1,2,3,4]}
  A_2^2= {[1,2,3]} ⋈ [3,5]={[1,2,3,5]}
  U_3 ∈ pending(Q_1) agrega Key_delete(A_1,[3,4])
  U_3 ∈ pending(Q_2) agrega Key_delete(A_2,[3,4])
  Agrega insert(MV,A_2) a AL = insert(MV,{[1,2,3,5]})

- Cuando UQS=0 aplica AL a MV como una transacción sencilla, sin agregar tuplas duplicadas en MV, AL=<>.

Capítulo 5. Una propuesta para la actualización de almacenes de datos.

En el presente trabajo se proponen dos nuevos algoritmos (**αVNL**[4] y **αVNLTR**[5]) que permiten realizar la actualización del almacén de datos de forma incremental y en línea (las consultas de los analistas y la actualización del almacén se ejecutan concurrentemente sin bloqueos). Ambos algoritmos permiten controlar un número ilimitado de versiones de cada vista materializada del almacén, ofreciendo de esta forma consistencia en lectura a los analistas que están conectados. El algoritmo **αVNL** realiza la actualización en batch, mientras que el algoritmo **αVNLTR** la realiza en tiempo real de forma automantenible (débil). Estos algoritmos tienen como antecesores los algoritmos 2VNL y NVNL de Dallan Quass [22].

5.1 Algoritmos antecesores.

5.1.1 Algoritmo 2VNL [22].

El algoritmo 2VNL permite que los analistas tengan disponible el almacén de datos sin interrupción. El algoritmo es un algoritmo en línea que implementa un control de la concurrencia entre consultas y actualizaciones de tipo multiversión, muy apropiado para el mantenimiento de vistas materializadas en un contexto de almacenes de datos. El nombre, 2VNL, significa: *two-version* (se mantienen

[4] αV versiones ilimitadas , NL sin bloqueo
[5] αV versiones ilimitadas , NL sin bloqueo, TR tiempo real

simultáneamente dos versiones de las tuplas), *no-locking* (no existe bloqueo).

Características:

- En línea: permite ejecutar la transacción de mantenimiento y las transacciones de consulta de los analistas concurrentemente sin bloqueo (evita los inconvenientes de la espera producida por el bloqueo).
- Lectura consistente: permite a los analistas ver una única versión del almacén de datos a lo largo de su sesión de trabajo.
- Se mantienen dos versiones de las tuplas actualizadas.

La Figura 42 muestra la evolución del almacén cuando se utiliza el algoritmo 2VNL.

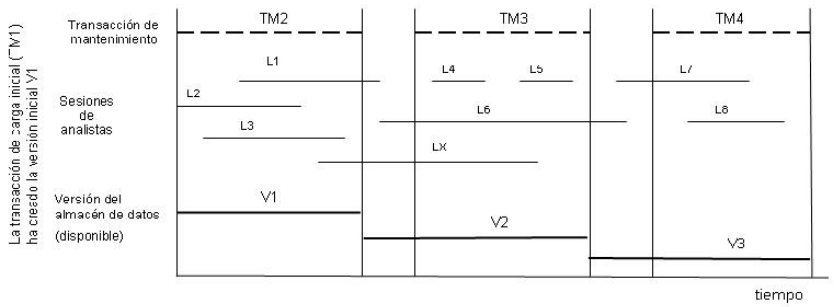


Figura 42. Evolución del almacén actualizado con el algoritmo 2VNL.

Interpretación de la Figura 42:

- La transacción de mantenimiento TM1 ha realizado la carga inicial del almacén de datos, creando la versión V1. A partir de ese momento sucesivas transacciones de mantenimiento crean las siguientes versiones del almacén.
- Las sesiones de los analistas L1, L2 y L3 leen la versión original del almacén de datos, versión V1, y se ejecutan concurrentemente

con la transacción de mantenimiento TM2 que está creando la primera versión actualizada del almacén de datos, versión V2. Nótese que TM2 finaliza antes de que L1 finalice su sesión. L1 termina la sesión con la versión del almacén con la que se inició (versión V1), aunque en el momento de finalizar ya existe una versión actualizada del almacén (versión V2).

- Las sesiones de los analistas L4, L5 y L6 leen la versión V2 del almacén de datos, que es la más actualizada en el momento en que se inicia su sesión, y se ejecutan concurrentemente con la transacción de mantenimiento TM3 que está creando una nueva[6] versión actualizada del almacén de datos, versión V3. Nótese que TM3 finaliza, es decir existe una nueva versión actualizada del almacén de datos, versión V3, antes de que L6 termine. L6 termina su sesión con la misma versión del almacén de datos con la que se inició (versión V2).
- La sesión de un analista LX sólo puede solaparse con una transacción de mantenimiento para asegurar una sesión de consulta consistente; si se solapa con más de una transacción de mantenimiento, su sesión expira ya que la versión con la que inició la sesión ya no existe.

Cada versión del almacén de datos, cada transacción de mantenimiento y cada sesión de usuario se identifican con un número único. La variable global del sistema *CurrentVN* se utiliza para asignar estos identificadores. Esta variable se inicializa con el valor 1 y se incrementa cada vez que una transacción de mantenimiento

[6] Cuando finaliza la transacción de mantenimiento TM3 deja de existir la versión original V1, en ese instante existen las versiones V2 y V3.

finaliza. Adicionalmente se requiere una variable global *MaintenanceActive*, que indica si una transacción de mantenimiento está activa en cada momento, esta variable se inicializa con el valor *false*.

La variable *CurrentVN* es utilizada por el algoritmo de mantenimiento de la siguiente forma:

- Para asignar un número de transacción a cada nueva transacción de mantenimiento. Cuando se inicia una transacción de mantenimiento se le asigna como identificador el valor de *CurrentVN*, y cuando finaliza la transacción el valor de esta variable se incrementa en 1. La transacción de carga inicial es la TM**1** (1 es el valor de inicialización de la variable *CurrentVN)*.
- Para asignar un número de versión a cada nueva versión creada del almacén de datos. La versión inicial será la versión V**1** (1 es el valor de inicialización de la variable *CurrentVN),* que será creada por la transacción de carga inicial TM**1**.
- Para asignar número de sesión a los analistas cuando se conectan al sistema. El número de sesión asignado a un usuario será *CurrentVN-1*.

Estas variables pueden ser representadas en una tabla de control, Control, como se muestra en la Tabla 9.

CurrentVN	MaintenanceActive
1	False

Tabla 9. Tabla Control (estado inicial).

Para usar el algoritmo 2VNL, el esquema de cada relación del almacén de datos debe ser extendido de la siguiente forma:

- Se añade un atributo *TuplaVN* que indica la última versión del almacén de datos en la que la tupla sufrió una actualización. Este atributo se modifica cuando una transacción de mantenimiento actualiza la tupla, recuérdese que la transacción de mantenimiento antes de iniciar la actualización del almacén de datos, accede a la tabla de control para consultar el valor de la variable global *CurrentVN*, éste será el número de la transacción de mantenimiento que se inicia así como el número de la versión que crea. Este valor será el utilizado para actualizar el atributo *TuplaVN*.
- Se añade un atributo *Operación* que indica la última operación realizada sobre la tupla, ésta puede ser *insert, delete,* o *update*.
- Se añade un conjunto de atributos *Pre-actualizados* que contienen el valor de los atributos actualizables antes de que se produjera la última actualización de la tupla. Los atributos actualizables de una relación son los que pueden ser modificados por una operación de actualización de la transacción de mantenimiento. Cuando la operación de mantenimiento es *insert* estos atributos tienen el valor *null* y si la operación es *delete* o update, contienen los valores de la tupla antes del borrado o la modificación.

Sea $A=\{A_1, A_2,..., A_n\}$ el conjunto de atributos de la relación R, y sea $A'=\{A_i, A_j,..., A_k\}$ el subconjunto de A que representa los atributos actualizables. El esquema extendido de R será:

$$\{\text{TuplaVN, Operación, } A_1, A_2,..., A_n, A^p_i, A^p_j..., A^p_k\}$$

Donde A^p_i denota el atributo *pre-actualizado* correspondiente al atributo A_i. En el peor de los casos, cuando todos los atributos sean actualizables, mantener dos versiones de las tuplas exige aproximadamente el doble de espacio de almacenamiento. Sin embargo es frecuente que muchos atributos del almacén de datos no sean actualizables, por ejemplo, los valores de agrupación (GROUP BY) que se usan para definir una tabla agregada no pueden ser modificados por una transacción de mantenimiento.

Cuando una transacción de mantenimiento inserta, borra o modifica una tupla varias acciones deben realizarse para mantener las versiones actual y pre-actualizada de la tupla:

- En algunos casos los valores actuales de los atributos son asignados a los atributos pre-actualizados, logrando con esto preservar la versión anterior de la tupla. Por ejemplo, cuando se realiza una operación de borrado (delete) o de modificación (update).
- En otros casos, a los atributos actuales se les asigna los nuevos valores especificados por la operación de mantenimiento. Por ejemplo, en una operación de inserción (insert) o de actualización (update).
- Al atributo *TuplaVN* se le asigna el valor de la variable global *CurrentVn*. El atributo *TuplaVN* indica la última transacción de mantenimiento que ha actualizado la tupla; es decir, el número de versión del almacén para el que la versión actual de la tupla fue creada.

– Al atributo *Operación*, se le asigna la operación lógica (insert, delete o update) realizada sobre la tupla por la transacción de mantenimiento. Una operación lógica realizada por una transacción de mantenimiento sobre la tupla puede no coincidir con la operación física realizada sobre la tupla por el algoritmo. Por ejemplo, cuando la operación lógica es un borrado, usualmente la tupla no es borrada físicamente del almacén, debido a que la versión pre-actualizada puede ser requerida por otros analistas conectados. Por último, el atributo *Operación* debe representar el efecto real de todas las operaciones realizadas por la transacción de mantenimiento sobre la tupla. Por ejemplo, si una transacción de mantenimiento inserta (insert) una tupla y luego modifica (update) la misma tupla, el efecto real sigue siendo una inserción.

Las acciones que deben realizarse sobre una tupla dependen de tres factores:

- de la operación de la transacción de mantenimiento,
- del valor del atributo *TuplaVN* en la tupla, y
- del valor del atributo *Operación* en la tupla.

Las figuras Figura 43, Figura 44 y Figura 45 muestran la operación física que se debe realizar sobre la tupla, para mantener la versión actual y pre-actualizada en cada caso. En estas tablas PV representa los valores de los atributos pre-actualizados de la tupla, CV los valores actuales de la tupla y MV los valores especificados en la operación de actualización.

	Operaciones previas: atributo *Operación* de la tupla		
	Insert	Update	Delete
TuplaVN < CurrentVN	Imposible	Imposible	Modificar tupla: PV:=Nulo CV:=MV TuplaVN:=CurrentVN Operación:=Insert
TuplaVN = CurrentVN	Imposible	Imposible	Modificar tupla: CV:=MV Operación:=Update
No existe la tupla	Insertar tupla: PV:=Nulo; CV:=MV TuplaVN:=CurrentVN Operación:=Insert		

Figura 43. Operación física sobre la tupla para INSERT

	Operaciones previas: atributo *Operación* de la tupla		
	Insert	Update	Delete
TuplaVN < CurrentVN	Modificar tupla: PV:=CV CV:=MV TuplaVN:=CurrentVN Operación:=Update	Modificar tupla: PV:=CV CV:=MV TuplaVN:=CurrentVN Operación:=Update	Imposible
TuplaVN= CurrentVN	Modificar tupla: CV:=MV	Modificar tupla: CV:=MV	Imposible

Figura 44. Operación física sobre la tupla para UPDATE

	Operaciones previas: atributo *Operación* de la tupla		
	Insert	Update	Delete
TuplaVN < CurrentVN	Modificar tupla: PV:=CV TuplaVN:=CurrentVN Operación:=Delete	Modificar tupla: PV:=CV TuplaVN:=CurrentVN Operación:=Delete	Imposible
TuplaVN= CurrentVN	Borrar la tupla	Modificar tupla: Operación:=Delete	Imposible

Figura 45. Operación física sobre la tupla para DELETE

Como ejemplo considérese un almacén de datos que contiene información sobre las ventas realizadas en una cadena de tiendas. El almacén contiene la vista materializada que se muestra en la Tabla 10.

Tabla 10. Esquema de la relación VentasDía.

El esquema modificado de la relación VentasDía para que el algoritmo 2VNL pueda trabajar se muestra en la Tabla 11.

Cuando se realiza la carga inicial, el atributo *TuplaVN* toma el valor 1, el atributo *Operación* el valor *insert* y el atributo *PreTotalVentas* el valor *null*. Al finalizar la carga inicial el atributo *CurrentVN* tiene el valor 2.

Tabla 11. Esquema modificado de VentasDía para el algoritmo 2VNL.

Para ilustrar como funciona el algoritmo 2VNL, supóngase que la relación VentasDia es cargada[7] con las tuplas que se muestran en la Tabla 12.

Tupla VN	Operacion	Ciudad	Producto	Fecha	Total Ventas	PreTotal Ventas
1	insert	Jose	Golf equip	14-10-04	10000	null
1	insert	Jose	Golf equip	15-10-04	1500	null
1	insert	Barkely	Raquetball	13-10-04	8000	null
1	insert	Novato	Pollerblades	14-10-04	12000	null

Tabla 12. Carga inicial de VentasDía.

[7] Se considera que la carga inicial es la transacción de mantenimiento 1.

Una vez finalizado el proceso de carga inicial de la relación VentasDía, la tabla de control tiene los valores que se muestran en la Tabla 13.

CurrentVN	MaintenanceActive
2	False

Tabla 13. Tabla de control después de la carga inicial.

En las tablas Tabla 14, Tabla 15, Tabla 17 y Tabla 18 se detalla como el algoritmo 2VNL actualiza la vista materializada VentasDía, suponiendo que las siguientes transacciones de mantenimiento tienen lugar:

a) La transacción de mantenimiento TM2, inserta la tupla <LA, Beisball, 15-12-04, 30000> y actualiza el importe de ventas de la tupla <Barkely, Raquetbaill, 13-10-04> a 55000.

TuplaVN	Operación	Ciudad	Producto	Fecha	TotalVentas	PreTotal Ventas
1	insert	José	Golfequip	14-10-04	10000	Null
1	insert	José	Golfequip	15-10-04	1500	Null
2	update	Barkely	Raquetball	13-10-04	55000	8000
1	insert	Novato	Pollerblades	14-10-04	12000	Null
2	insert	LA	Beisball	15-12-04	30000	Null

Tabla 14. VentasDía después de la transacción de mantenimiento TM2.

b) La transacción de mantenimiento TM3, actualiza el importe de ventas de la tupla <José, Golfequip, 15-10-04> a 78000.

TuplaVN	Operación	Ciudad	Producto	Fecha	TotalVentas	PreTotal Ventas
1	insert	José	Golf equip	14-10-04	10000	Null
3	update	José	Golfequip	15-10-04	78000	1500
2	update	Barkely	Raquetball	13-10-04	5000	8000
1	insert	Novato	Pollerblades	14-10-04	12000	Null
2	insert	LA	Beisball	15-12-04	30000	Null

Tabla 15. VentasDia después de la transacción de mantenimiento TM3.

c) Un analista con número de sesión 2 [8] que consulta la tabla en el estado anterior, lee las tuplas que se muestran en la Tabla 16.

Ciudad	Producto	Fecha	Ventas
José	Golfequip	14-10-04	10000
José	Golfequip	15-10-04	1500
Barkely	Raquetball	13-10-04	55000
Novato	Pollerblades	14-10-04	12000
LA	Beisball	15-12-04	30000

Tabla 16. Consulta VentaDía con número de sesión 2

d) La transacción de mantenimiento TM4, actualiza el importe de ventas de la tupla <LA, Beisball, 15-12-04> a 100000.

TuplaVN	Operación	Ciudad	Producto	Fecha	TotalVentas	PreTotal Ventas
1	insert	José	Golfequip	14-10-04	10000	Null
3	update	José	Golfequip	15-10-04	78000	1500
2	update	Barkely	Raquetball	13-10-04	5000	8000
1	insert	Novato	Pollerblades	14-10-04	12000	Null
4	update	LA	Beisball	15-12-04	100000	30000

Tabla 17 VentasDía después de la transacción de mantenimiento TM4

[8] Este usuario se conectó al sistema cuando la última versión actualizada del almacén era la versión 2, independientemente de que estuviese ya en marcha o no la transacción de mantenimiento TM3.

e) La transacción de mantenimiento TM5, actualiza el importe de ventas de la tupla <Jose, Golf equip, 15-10-04> a 155999.

TuplaVN	Operación	Ciudad	Producto	Fecha	TotalVentas	PreTotal Ventas
1	insert	Joé	Golfequip	14-10-04	10000	Null
5	update	José	Golfequip	15-10-04	155999	78000
2	update	Barkely	Raquetball	13-10-04	55000	8000
1	insert	Novato	Pollerblades	14-10-04	12000	Null
4	update	LA	Beisball	15-12-04	100000	30000

Tabla 18. VentasDia después de la transacción de mantenimiento 5.

f) Si el mismo analista con número de sesión 2, consultase la tabla VentasDía en el estado anterior, leería el siguiente conjunto de tuplas:

Ciudad	Producto	Fecha	Ventas
José	Golfequip	14-10-04	10000
José	Golfequip	15-10-04	78000
Barkely	Raquetball	13-10-04	55000
Novato	Pollerblades	14-10-04	12000
LA	Beisball	15-12-04	30000

Tabla 19. Consulta inconsistente de VentasDia con número de sesión 2.

Obsérvese que los resultados mostrados en la Tabla 19 son erróneos. Existe una inconsistencia con la lectura anterior de la tupla, <José, Golfequip, 15-10-04> ya que para una misma sesión de trabajo el analista lee valores diferentes para el total de ventas. Este hecho obliga a que las sesiones de los analistas sean expiradas cuando ya no se les puede ofrecer lectura consistente (la misma vista) del almacén de datos. Cuando termina la transacción de mantenimiento TM4 deben ser abortadas las sesiones de usuario con número de sesión 2.

La consulta anterior, de un usuario con número de sesión 2, se puede expresar por medio de una secuencia de instrucciones en SQL

(mostradas en la Figura 46 como parte de un programa en Visual Basic).

```
DIM Sesión AS Int
Sesión := <nro_sesión_de_usuario>
SELECT Ciudad, Producto, Fecha
        CASE WHEN :Sesión ≥ TuplaVN
        THEN  TotalVentas
        ELSE  PreTotalVentas
        END  AS Ventas
FROM  VentasDía
WHERE (:Sesión  ≥ tuplaVN AND Operación<>'delete')
                                    OR
          (:Sesión < tuplaVN  AND Operación<>'insert')
```

Figura 46. Procedimiento para consultar VentasDía actualizada con el algoritmo 2VNL.

Cuando un usuario inicia una sesión de consulta se le asigna un número de sesión que sirve para decidir qué versión del almacén debe acceder, siempre se le asigna la última versión actualizada del almacén existente en el momento de conexión. Así, si el número de sesión del usuario es mayor o igual que el valor que tiene el atributo *tuplaVN* en la tupla, lee los atributos que representan los valores actuales (TotalVentas), en caso contrario usa los atributos que representan los valores pre-actualizados (PreTotalVentas).

Para que este algoritmo realice la actualización del almacén, es necesario un trabajo previo, consistente en generar una tabla, *Actualizaciones*, que contenga las actualizaciones (después del proceso de transformación) de las fuentes que sean relevantes para el almacén. El algoritmo 2VNL necesita las tablas que se muestran en la Figura 47.

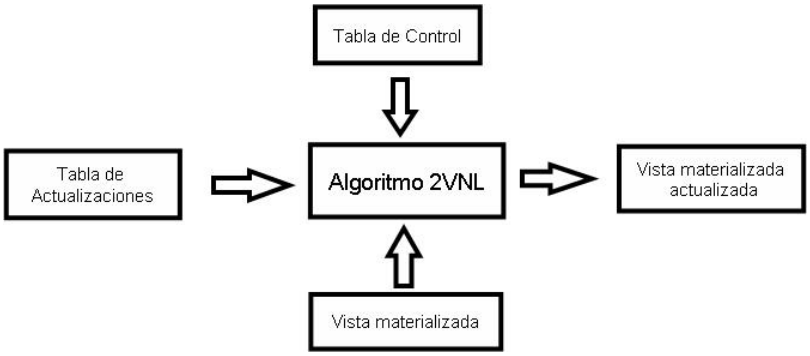


Figura 47. Tablas necesarias para uso del algoritmo 2VNL.

El esquema de la tabla *Actualizaciones* (con las tuplas relevantes para la vista) para el ejemplo de VentasDía se muestra en la Tabla 20.

Operación
Ciudad
Producto
Fecha
TotalVentas

Tabla 20. Esquema de la tabla Actualizaciones.

5.1.2. Algoritmo NVNL.

El algoritmo 2VNL evita el problema de mantener el almacén de datos inhabilitado para los analistas mientras se está actualizando, ya que permite ejecutar concurrentemente la transacción de mantenimiento y las transacciones de usuario. Sin embargo, este algoritmo presenta una limitación: la sesión de un usuario puede expirar debido a que sólo se mantienen dos versiones del almacén de datos. Para superar esta

limitación, se puede extender el algoritmo 2VNL al caso NVNL, en el que se mantienen N versiones del almacén de datos.

Incrementando a N el número de versiones se aumenta la franja de tiempo en la que las sesiones de los analistas pueden estar activas. Por ejemplo, supóngase que i es el intervalo mínimo que existe entre la ejecución de dos transacciones de mantenimiento consecutivas y m es el tiempo (duración) que dura la transacción de mantenimiento más corta, entonces el algoritmo 2VNL garantiza que las sesiones de analistas con una duración máxima m+i nunca expirarán, este hecho se muestra en la Figura 48.

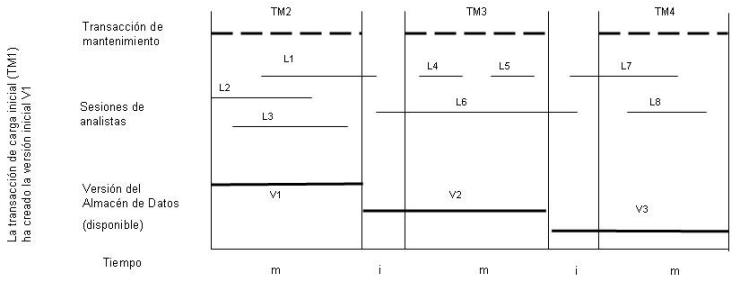


Figura 48. Evolución del almacén de datos actualizado con NVNL.

En el algoritmo 3VNL las sesiones de analistas con duración máxima de 2i + m nunca expirarán. En general NVNL garantiza que las sesiones de analistas con una duración máxima de (n-1) * (i+m) – m nunca expirarán. De esta manera n puede ser ajustada para el tipo de sesiones de analistas y de transacciones de mantenimiento esperado en el almacén de datos, para evitar la expiración de las transacciones de

usuario. Evidentemente, cuanto mayor es el valor de N, mayor es el coste de almacenamiento y el tiempo de ejecución del algoritmo de actualización.

El algoritmo NVNL necesita la misma tabla de control que usa el algoritmo 2VNL y el mismo tipo de extensión de los esquemas de las vistas materializadas del almacén, con el mismo significado para los atributos *TuplaVN*, *Operación* y los atributos pre-actualizados. Para el algoritmo NVNL el esquema de las relaciones se modifica de igual forma que para el algoritmo 2VNL, pero añadiendo N-1 conjuntos de atributos actualizables, uno para cada versión previa de la tupla mantenida por el algoritmo (1 versión actual + N-1 versiones anteriores).

Sea A = (A_1, A_2, ..., A_n) el conjunto de atributos de la vista materializada y sea A' = (A_i, A_j, ..., A_k) el subconjunto de atributos actualizables de A. El esquema de la versión actual es equivalente al esquema original ya que contiene los atributos que representan la clave primaria, los atributos no actualizables y los atributos actualizables. El esquema de la versión i-esima contiene los dos atributos de control $TuplaVN_i$, $Operación_i$ y los atributos actualizables (A_i, A_j,..., A_k), como se muestran en la Figura 49.

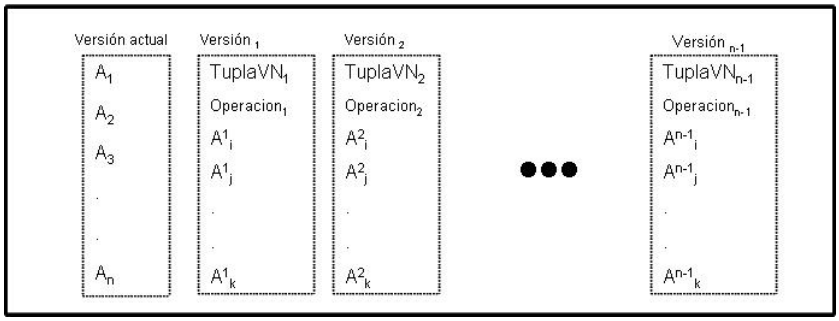


Figura 49. Esquema de la vista materializada en el algoritmo NVNL.

Cada conjunto de atributos con un mismo índice 1, 2, ..., N-1 representa una versión previa de la tupla, siendo la versión más reciente la representada por el conjunto de atributos actuales (si no hay transacción de mantenimiento activa) o por el conjunto de atributos con índice 1 (si hay transacción de mantenimiento activa). Por ejemplo, el esquema de la relación VentasDía para 4VNL se representa en la Tabla 21.

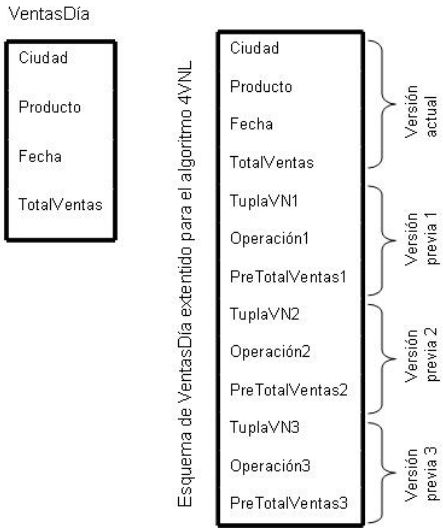


Tabla 21. Esquema de la vista materializada para NVNL (N=4).

En las tablas Tabla 22, Tabla 23 y Tabla 24 se detalla como el algoritmo NVNL (N=4) modifica una tupla de la vista materializada VentasDía, suponiendo que las siguientes transacciones de mantenimiento tienen lugar:

a) La transacción de mantenimiento TM3 inserta la tupla <José, Sport, 14-10-96, 10000>.

Ciudad	Producto	Fecha	Total Ventas	Tupla VN1	Operación1	PreTotal Ventas1	Tupla VN2	Operación2	PreTotal Ventas2	Tupla VN3	Operación3	PreTotal Ventas3
José	Sport	14-10-96	10000	3	insert	null						

Tabla 22. VentasDía después de la transacción de mantenimiento TM3. (NVNL N=4).

b) La transacción de mantenimiento TM5, actualiza el total de ventas de la tupla <José, Sport, 14-10-96> a 10200.

Ciudad	Producto	Fecha	Total Ventas	Tupla VN1	Operación1	PreTotal Ventas1	TuplaVN 2	Operación2	PreTotal Ventas2	Tupla VN3	Operación3	PreTotal Ventas3
José	Sport	14-10-96	10200	5	update	10000	3	insert	null			

Tabla 23. VentasDía después de la transacción de mantenimiento TM5. (NVNL N=4).

c) La transacción de mantenimiento TM6, borra la tupla <José, Sport, 14-10-96>.

Ciudad	Producto	Fecha	Total Ventas	Tupla VN1	Operación1	PreTotal Ventas1	Tupla VN2	Operación2	PreTotal Ventas2	Tupla VN3	Operación3	PreTotal Ventas3
José	Sport	14-10-96	10200	6	delete	10200	5	update	10000	3	insert	null

.

Tabla 24. VentasDía después de la transacción de mantenimiento TM6. (NVNL N=4).

Cuando un analista realiza una consulta al almacén la consulta se evalua según el procedimiento de la Figura 50.

```
DIM Sesión AS Int
Sesión := <nro_sesión_usuario>
IF Sesión ≥ TuplaVN1
THEN
   -- Se accede a los atributos de la versión actual --
ELSE
   IF Sesión < TuplaVN1 AND Sesión ≥ TuplaVNn-1-1
   THEN
       -- Se accede a la versiÓn j-esima de la tupla --

         FOR J=n-1 TO 1    /* localización de TuplaVNJ */
            IF TuplaVNJ > Sesión
            THEN
               -- Se accede a la versión j-esima --
               EXIT
            ENDIF
         ENDFOR
    ENDIF
ENDIF
```

Figura 50. Procedimiento para consultar VentasDía actualizada por el algoritmo NVNL (N=4).

En las tablas Tabla 25, Tabla 26, Tabla 27 y Tabla 28 se muestran resultados de consultas sobre VentasDia actualizada con el algoritmo NVNL (N=4). Se supone que las consultas de los usuarios tienen lugar cuando la versión actual del almacén es la versión V6, es decir la última transacción de mantenimiento que actualizó el almacén fue la transacción de mantenimiento TM6.

a) Un analista con número de sesión 6, que consulte la tupla <José, Sport, 14-10-96>, no leerá ningún valor, ya que la tupla ha sido borrada por la transacción de mantenimiento TM6.

Ciudad	Producto	Fecha	Total Ventas	Tupla VN1	Operación1	PreTotal Ventas1	Tupla VN2	Operación2	PreTotal Ventas2	Tupla VN3	Operación3	PreTotal Ventas3
José	Sport	14-10-96	10200	6	**delete**	10200	5	update	10000	3	insert	null

El usuario no leerá ninguna versión de la tupla. La tupla ha sido borrada por la última transacción de mantenimiento.

Tabla 25. Consulta de VentasDia con número de sesión 6.

b) Un analista con número de sesión ∈ {3, 4} que consulte la tupla <José, Sport, 14-10-96> leerá la tupla lógica que se muestra a continuación.

Versión que lee el analista

Ciudad	Producto	Fecha	Total Ventas
José	Sport	14-10-96	10000

Ciudad	Producto	Fecha	Total Ventas	Tupla VN1	Operación1	PreTotal Ventas1	Tupla VN2	Operación2	PreTotal Ventas2	Tupla VN3	Operación3	PreTotal Ventas3
José	Sport	14-10-96	10200	6	delete	10200	5	update	10000	3	**insert**	null

Tabla 26. Consulta de VentasDía con número de sesión 3 o 4.

c) Un analista con número de sesión 5 que consulte la tupla <José, Sport, 14-10-96>, leerá la tupla lógica que se muestra a continuación.

Versión que lee el analista

Ciudad	Producto	Fecha	Total Ventas
José	Sport	14-10-96	10200

Ciudad	Producto	Fecha	Total Ventas	Tupla VN1	Operación1	PreTotal Ventas1	Tupla VN2	Operación2	PreTotal Ventas2	Tupla VN3	Operación3	PreTotal Ventas3
José	Sport	14-10-96	10200	6	delete	10200	5	**update**	10000	3	insert	null

Tabla 27. Consulta de VentasDía con número de sesión 5.

d) Un analista con número de sesión 2 que consulte la tupla <José, Sport, 14-10-96>, no leerá ningún valor ya que el campo pre-actualizado para la versión que le correspondería leer es nulo.

Ciudad	Producto	Fecha	Total Ventas	Tupla VN1	Operación1	PreTotal Ventas1	Tupla VN2	Operación2	PreTotal Ventas2	Tupla VN3	Operación3	PreTotal Ventas3
José	Sport	14-10-96	10200	6	delete	10200	5	update	10000	3	insert	null

El usuario no leerá ninguna versión de la tupla. La tupla no existía en la versión V2.

Tabla 28. Consulta de VentasDía con número de sesión 2.

e) Un analista con número de sesión menor que 2, que consulte la tupla <José, Sport, 14-10-96>, no podrá leer y su sesión será abortada, se dice que su sesión ha expirado.

El algoritmo NVNL, que mantiene N versiones de las tuplas en el almacén, permite alargar el tiempo de las sesiones de consulta de los usuarios, pero sigue limitando su duración. Esta situación es un problema en algunos contextos de trabajo, por ello surge la necesidad de buscar otros algoritmos que superen esta limitación. Éste es el punto de partida, y la motivación de los algoritmos de mantenimiento propuestos en este trabajo de tesis.

5.2 Algoritmos propuestos.

5.2.1 Algoritmo ∝VNL.

En el algoritmo NVNL presentado en el apartado anterior, una vez se ha fijado el valor de N no puede modificarse, ya que esto requeriría cambiar el esquema de la vista materializada en el almacén de datos. Según el valor de N el algoritmo NVNL mantiene N versiones de las tuplas, garantizando a los analistas sesiones de consulta consistentes más duraderas. Para garantizar la consistencia, los valores establecidos para los parámetros i y m de las transacciones de mantenimiento, tienen que respetarse rigurosamente. En algunos contextos esto no es posible, por ejemplo en un almacén de datos en el que la transacción de mantenimiento se pude iniciar a cualquier hora y con una duración diferente en cada caso.

Siguiendo la lógica de mantenimiento de los algoritmos anteriores, se propone modificar el algoritmo NVNL para que mantenga un número ilimitado de versiones de las tuplas y garantice una lectura consistente

a los analistas sin hacer uso de los parámetros fijados en el algoritmo NVNL. Una situación de este tipo se ilustra en la Figura 51.

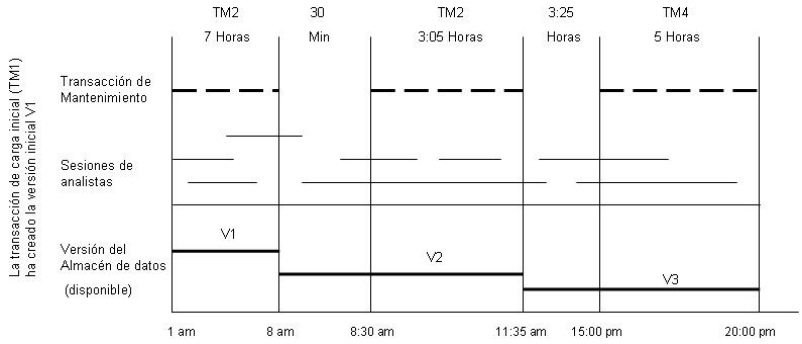


Figura 51. Evolución del almacén de datos actualizado con αVNL.

Obsérvese como en la Figura 51, las transacciones de mantenimiento (TM1, TM2 y TM3) tienen una duración diferente. También se observa como una transacción de mantenimiento puede iniciarse en cualquier momento una vez terminada la transacción de mantenimiento anterior (TM2 se inicia 30 minutos después de finalizar TM1 y TM3 se inicia 3 horas y 25 minutos después de finalizar TM2).

Ya que el algoritmo ∝VNL puede mantener un número ilimitado de versiones de las tuplas, será necesario un recolector de basura (controlado por el administrador del almacén de datos) para eliminar las versiones de las tuplas que ya no van a ser consultadas por ninguna sesión de usuario.

El primer paso que se debe realizar para usar el algoritmo ∝VNL es modificar el esquema del almacén para poder mantener las versiones

de las tuplas. En este caso no es posible extender el esquema de la vista materializada con N-1 conjuntos de atributos pre-actualizados, ya que el valor de N no se conoce previamente. Para resolver este problema, se definen para cada vista materializada, dos tablas que representarán la vista de la siguiente forma:

- Una tabla H formada por los atributos de la clave primaria de la vista, los atributos que no son actualizables y el atributo *ÚltimoVN* que indica el número de la última transacción de mantenimiento que modificó la tupla[9].
- Una tabla D cuyas filas representan distintas versiones de las tuplas. Esta tabla D esta formada por los atributos de la clave primaria de la vista original, los atributos *VnInicio*, *VnFin*, *OperaciónVN* y los atributos que son actualizables. *VnInicio* y *VnFin* son atributos de control que indican para qué sesiones de usuario es visible una versión de la tupla, y *OperacionVN* indica la operación lógica que generó la versión de la tupla.

Para ilustrar la definición del esquema de la vista materializada en nuestra propuesta, supóngase que se tiene la vista materializada que se muestra en la Tabla 29.

[9] Es decir, el número de la versión del almacén en la que fue modificada la tupla.

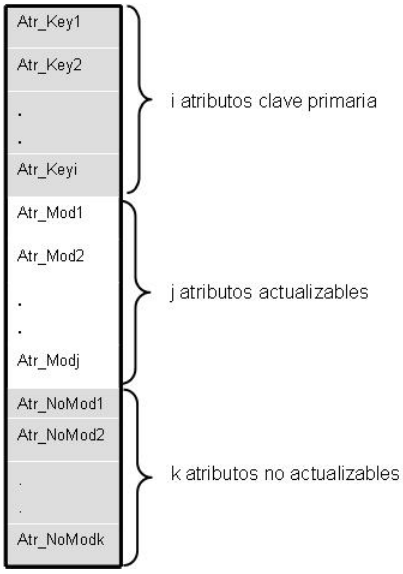


Tabla 29. Esquema de una vista materializada.

El esquema modificado de la vista se muestra en la Tabla 30.

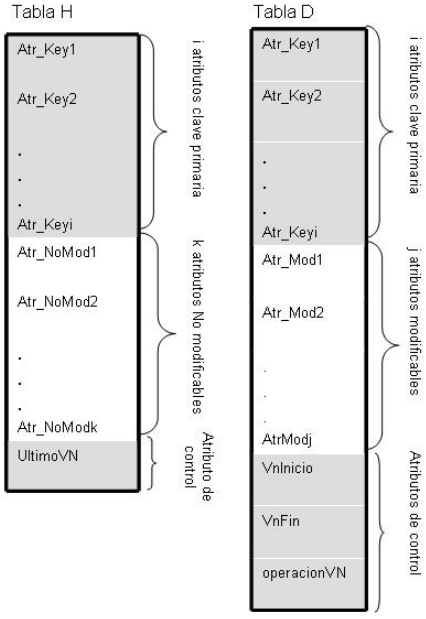


Tabla 30. Esquema modificado de la vista materializada.

Para usar el algoritmo ∝VNL, además del esquema modificado de la vista (tablas H y D), serán necesarias también, la tabla *Actualizaciones*, con las actualizaciones relevantes de las fuentes, y la tabla *Control* con las variables de control. La Figura 52 muestra las tablas necesarias en el algoritmo ∝VNL.

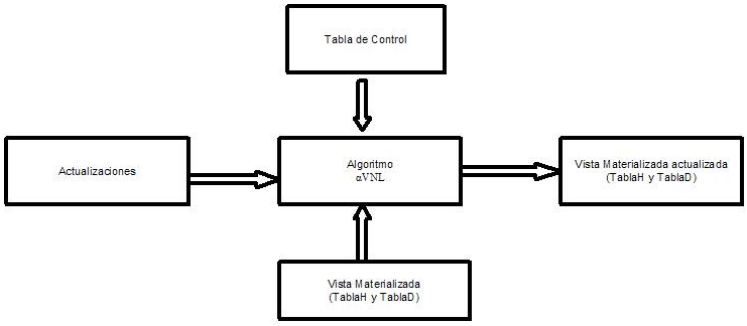


Figura 52. Tablas requeridas para ejecutar el algoritmo αNNL.

En la Tabla 31 se muestran la tabla de actualizaciones y la tabla de control.

Actualizaciones

Operación
Atr_Key1
Atr_Key2
.
.
Atr_Keyi
Atr_Mod1
Atr_Mod2
.
.
Atr_Modj
Atr_NoMod1
Atr_NoMod2
.
.
Atr_NoModk

i atributos clave primaria

j atributos actualizables

k atributos no actualizables

Control

CurrentVN
MaintenanceActive

Tabla 31. Esquema de las tablas: Actualizaciones y Control.

Algoritmo de mantenimiento ∝VNL.

Cuando una transacción de mantenimiento inserta, borra o modifica una tupla, varias acciones deben realizarse para mantener las versiones actual y pre-actualizadas de la tupla, como se muestra en el algoritmo de la Figura 53.

```
/* TDW: tupla de la tabla Actualizaciones
H y D: tablas que representan la vista materializada
Control: tabla de control  */

DIM Sesión, Inicio AS Int
SELECT * INTO t FROM Control;
Sesión:=t.CurrentVN;

/* Sea t una tupla de la tabla Actualizaciones:
CP(t): clave primaria de la tupla t
AtrNM(t): atributos no actualizables de t
AtrM(t): atributos actualizables de t
OP(t): atributo Operación de t */

SELECT * FROM H t WHERE CP(t)=CP(TDW);
IF No_Data_Found
THEN
   INSERT INTO H
   VALUES (CP(TDW), AtrNM(TDW), Sesión);
   INSERT INTO D
   VALUES(CP(TDW), AtrM(TDW), Sesión, 2147000000, 'insert')
ELSE
   SELECT * INTO tupla FROM D t
   WHERE CP(t)=CP(TDW) AND t.VnFin=2147000000;
   IF tupla.VnInicio=Sesión
   THEN
      UPDATE D t
      SET t.AtrM(t) = AtrM(TDW),
          t.OperaciónVN = OP(TDW)
      WHERE CP(t)=CP(TDW) AND t.VnFin=2147000000
   ELSE
      UPDATE D t
      SET t.VNFin=Sesión-1
      WHERE CP(t)=CP(TDW) AND t.VnFin=2147000000;
      INSERT INTO D
      VALUES (CP(TDW), AtrM(TDW), Sesión, 2147000000, OP(TDW))
   ENDIF;
   UPDATE H t
   SET t.UltimoVN=Sesión
   WHERE CP(t)=CP(TDW);
   UPDATE Control SET MaintenanceActive=true
ENDIF
```

Figura 53. Algoritmo αVNL.

Para ilustrar el funcionamiento del algoritmo ∝VNL, analícese el siguiente ejemplo. Sea el esquema de la relación VentasDía que se muestra en la Tabla 32.

Tabla 32. Esquema de la relación VentasDía.

El esquema de la relación VentasDia modificado para el algoritmo ∝VNL se muestra en la Tabla 33.

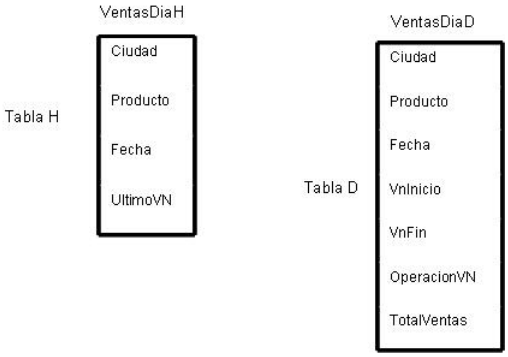


Tabla 33. Esquema de la relación VentasDía para αNVNL.

En las tablas Tabla 34, Tabla 35 y Tabla 36 se detalla como el algoritmo αNVNL irá registrando las actualizaciones en la vista materializada VentasDía, suponiendo que las siguientes transacciones de mantenimiento tienen lugar:

a) La transacción de mantenimiento TM3, inserta la tupla <José, Sport, 14-10-96, 10000>.

VentasDiaH

Ciudad	Producto	Fecha	UltimoVN
Jose	Sport	14-10-96	3

VentasDiaD

Ciudad	Producto	Fecha	VnInicio	VnFin	OperacionVN	TotalVentas
Jose	Sport	14-10-96	3	2147000000	insert	10000

Tabla 34. VentasDía después de la transacción de mantenimiento TM3.

(El número 2147000 del atributo VnFin representa un valor infinito)

b) La transacción de mantenimiento TM5, actualiza el total de ventas de la tupla <José, Sport, 14-10-96> a 10200.

Ciudad	Producto	Fecha	UltimoVN
Jose	Sport	14-10-96	5

Ciudad	Producto	Fecha	VnInicio	VnFin	OperacionVN	TotalVentas
Jose	Sport	14-10-96	3	4	insert	10000
Jose	Sport	14-10-96	5	2147000000	update	10200

Tabla 35. VentasDia después de la transacción de mantenimiento TM5.

c) La transacción de mantenimiento TM6, borra la tupla <José, Sport, 14-10-96>.

Ciudad	Producto	Fecha	UltimoVN
Jose	Sport	14-10-96	6

Ciudad	Producto	Fecha	VnInicio	VnFin	OperacionVN	TotalVentas
Jose	Sport	14-10-96	3	4	insert	10000
Jose	Sport	14-10-96	5	5	update	10200
Jose	Sport	14-10-96	6	2147000000	delete	10200

Tabla 36. VentasDia después de la transacción de mantenimiento TM6.

Con el estado de la relación VentasDíaD, mostrado en la Tabla 36, supóngase que un analista con número de sesión 5 desea leer la tupla <José, Sport, 14-10-96>. De las tres versiones de la tupla que existen en la tabla VentasDíaD, leerá aquella para la que 5 >=VnInicio y 5<=VnFin, el resultado obtenido se muestra en la Tabla 37.

Ciudad	Producto	Fecha	TotalVentas
José	Sport	14-10-96	10200

Tabla 37. Consulta de VentasDía con número de sesión 5.

5.2.2. Algoritmo ∝VNLTR.

El algoritmo **∝VNLTR** realiza la actualización del almacén de datos en línea, en tiempo real y es automantenible. La característica de automantenible está definida para un patrón de vistas materializadas definidas como agregaciones de relaciones básicas. Para asegurar esta característica, es necesario que para cada relación básica R que interviene en la definición de una vista V se defina una tabla auxiliar TA_R.

En la tabla auxiliar TA_R existen dos tipos de atributos:

- Atributos de Clave Primaria: son los atributos que están en el GROUP BY de la sentencia SELECT que define la vista.

– Atributos actualizables: son los que pueden cambian de valor y definen un atributo agregado de la vista.

Las tablas auxiliares serán utilizadas para almacenar los valores parciales de los atributos actualizables de la vista, que están definidos por funciones de agregación. El nombre de estos atributos se forma con el prefijo DW<*FunciónAgregación*>. En este trabajo se han considerado las funciones de agregación Sum, Count, Avg, Min y Max, de tal forma que los nombres de los atributos de estas tablas auxiliares se forman a partir del nombre del atributo de la relación básica de la forma y el correspondiente prefijo:

DWSum <nombre_atributo>,
DWCount <nombre_atributo>,
DWAvg <nombre_atributo>,
DWMin <nombre_atributo>, y
DWMax <nombre_atributo>.

Ejemplo 1: Sea una vista materializada V definida por agregación de las relaciones básicas R1 y R2 de la forma:

CREATE VIEW V AS
SELECT A_i, ..., A_j
$Ag(B_k)$, ..., $Ag(B_l)$, $Ag(D_k)$, ..., $Ag(D_l)$
FROM R1, R2
WHERE $A_i=C_i$ AND ... AND $A_j=C_j$
GROUP BY A_i, ..., A_j .

Donde R1 (A_1, ..., A_N, B_1, ..., B_M) (resp. R2 (C_1, ..., C_N, D_1, ..., D_M) es el esquema de la relación básica R1 (resp. R2), y A_1, ..., A_N (resp. C_1,

..., C_N) son los atributos identificadores, y B_1, ..., B_M (resp. D_1, ..., D_M) son los atributos actualizables.

V (A_i, ..., A_j, Ag(B_k), ..., Ag(B_l), Ag(D_k), ..., Ag(D_l)) es el esquema de la vista donde (A_i, ..., A_j) ∈ [A_1 ... A_N], B_k, ..., B_l ∈ [B_1 ... B_M], D_k, ..., D_l ∈ [D_1 ... D_M], y Ag representa cualquier función de agregación.

El esquema de las tablas auxiliares es:

TA_{R1} (A_i, ..., A_j, DWAgB$_k$, ..., DWAgB$_l$)

TA_{R2} (C_i, ...,C_j, DWAgD$_k$, ..., DWAgD$_l$).

Los datos almacenados en estas tablas auxiliares serán utilizados posteriormente, para el cálculo de las nuevas versiones de las tuplas de la vista. Además, para cada tabla auxiliar que contenga atributos con las funciones Min y/o Max, se necesita una tabla asociada con el esquema que se muestra en la Tabla 38.

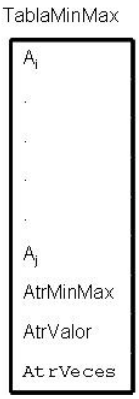


Tabla 38. Esquema de la tabla asociada a las tablas auxiliares.

Donde:

- Clave primaria: está formada por los atributos A_i, ..., A_j, AtrMinMax, y AtrValor, donde A_i, ..., A_j son los atributos que constituyen la clave primaria de la tabla auxiliar.

- AtrMinMax: contiene el nombre del atributo de la relación básica al que se le aplica la función Min o Max.
- AtrValor: contiene el valor mínimo (o máximo) del atributo en AtrMinMax.
- AtrVeces: contiene el número de veces que se encuentra el valor mínimo (o máximo) en el sistema operacional.

Las tablas asociadas contendrán información resumida de los valores de los atributos de las relaciones básicas a los que se les aplica una función Min o Max en la definición de la vista.

Para ilustrar el uso de las tablas auxiliares y las tablas asociadas, considérese el siguiente ejemplo. Sean las relaciones básicas: Docencia (Esc, CveEmp, Horas), Asesoria (Esc, CveEmp, Horas), y Inv (Esc, CveEmp, Horas), en un sistema universitario, donde las tres relaciones representan los servicios en horas (Horas) de docencia, de asesoría (gestión) y de investigación que los profesores (CvEmp) prestan en las distintas escuelas de la Universidad (Esc). La sentencia SELECT que se muestra en la Figura 54 define una vista del almacén de datos que contiene información con el número total de horas de docencia, asesoría e investigación prestadas por un profesor que se dedica a las tres actividades, así como el número mínimo de horas de docencia que imparte en una escuela y el número máximo de horas de investigación que realiza en una escuela.

```
CREATE VIEW V AS
SELECT D.CveEmp,SUM(D.Horas) AS HDoc,SUM(A.Horas)AS HAse,
       SUM(I.Horas)AS HInv,MIN(D.Horas)AS MinDoc, MAX(I.Horas)AS MaxInv
FROM  Docencia D, Asesoria A, Inv I
WHERE D.CveEmp=A.CveEmp AND D.CveEmp=I.CveEmp
GROUP BY CveEmp
```

Figura 54. Vista materializada V.

Para el mantenimiento de la vista V con el algoritmo ∝VNLTR, se deben definir en el almacén de datos las tablas auxiliares, las tablas asociadas y la vista materializada que se muestran en la Figura 55. Existen tres tablas auxiliares una para cada relación básica relevante para la vista, dos tablas asociadas (una para cada tabla auxiliar en la que aparece una función Min o Max) y dos tablas para representar la vista materializada.

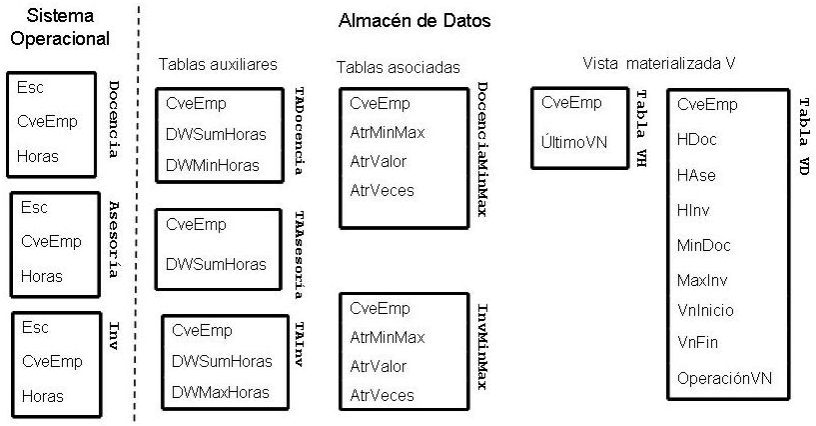


Figura 55. Tablas auxiliares, asociadas y vista materializada V.

El algoritmo necesita un Monitor/Wrapper (asociado a cada relación básica) que se ejecuta después de que se haya realizado una inserción o borrado en las relaciones básicas. Los monitores son los

responsables de actualizar las vistas auxiliares y de decidir si el almacén de datos debe actualizarse.

En esta propuesta no existe una transacción de mantenimiento definida como tal ya que las actualizaciones de los sistemas operacionales se reflejan inmediatamente en las vistas auxiliares (actualización en tiempo real), sin embargo para mantener las características del algoritmo ∝VNL**,** se considerará que una transacción de mantenimiento finaliza cuando se cumplen dos condiciones:

- un analista realiza una consulta y,
- antes de la consulta se ha generado desde las tablas auxiliares por lo menos una actualización del almacén de datos (Figura 56).

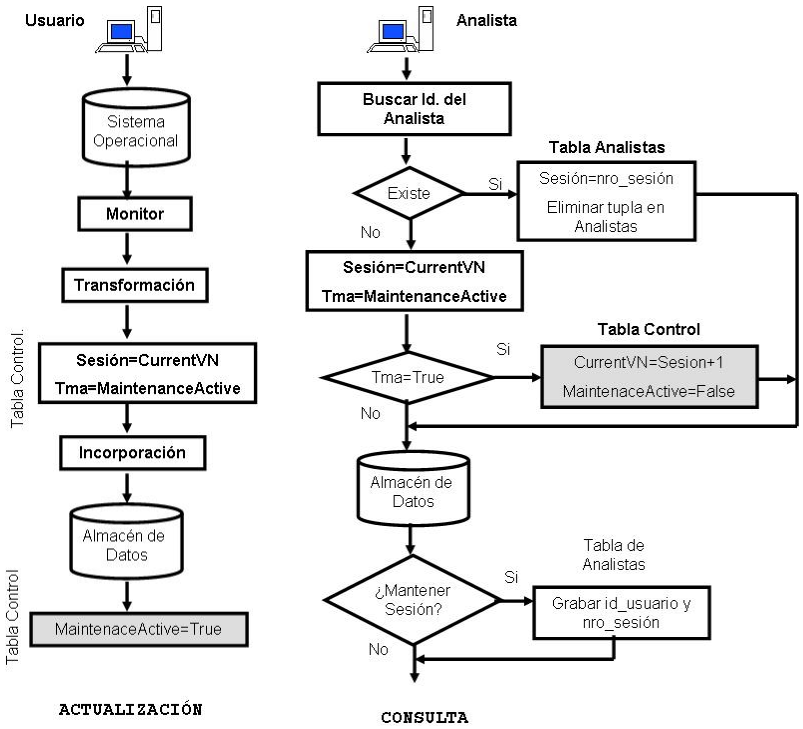


Figura 56. Sesiones de consulta y actualización del almacén.

Cuando un analista realiza una consulta al almacén de datos lo hace con un número de sesión que toma del atributo *CurrentVN* de la tabla de control, si el analista quiere mantener esa sesión de consulta para accesos posteriores sin importar que haya versiones más actualizadas del almacén, debe registrarse en la tabla de analistas (Tabla 39). Obsérvese en la Figura 56, que los analistas antes de acceder el almacén de datos obtienen el número de sesión, que puede asignarse de la tabla de analistas (se consulta una versión del almacén que previamente se había consultado) o de la tabla de control (se consulta la versión actual de almacén).

Tabla 39. Esquema de la tabla de analistas.

Cuando una operación de inserción o borrado (por simplicidad se omite la operación de modificación) tiene lugar en los sistemas operacionales dos procesos son necesarios para actualizar el almacén de datos:

1.- Actualización de las tablas auxiliares.

2.- Actualización del almacén de datos.

1.- Actualización de las tablas auxiliares.

Las acciones que deben realizarse para transmitir una actualización en una relación básica (R) a la tabla auxiliar TA_R correspondiente a R, para poder mantener el almacén de datos, son las siguientes:

a) Transformación de la tupla (t) de R (limpieza de datos, cambio de formato y operaciones aritméticas) en una tupla transformada (Tt).

b) Extracción de Tt de los valores de los atributos relevantes para actualizar la tabla auxiliar TA_R, obteniendo una nueva tupla (t'). La tupla t' estará formada por los atributos de agrupación A_i, ..., A_j y los atributos de agregación de la vista que proceden de R, es decir los atributos <*Atr*> tales que DWAg<*Atr*> aparece en la vista y <*Atr*> es un atributo de R. (En el ejemplo, t' = <CveEmp, DWSumHoras>).

c) Actualización de la tabla auxiliar TA_R:

Actaulización de TA_R en caso de INSERT:

```
/* Sea t' una tupla (transformada) de R:
CP_TA_R: atributos de la clave primaria de V
CP(t'): clave primaria de la tupla t'
AtrSUM_i: atributo de V con función SUM
AtrSUM_i(t): valor para t'del atributo de R que se agrega en AtrSUM_i de V
AtrCOUNT_i: atributo de V con función COUNT
AtrMIN_i(t): atributo de t' con función MIN en V
AtrMAX_i(t): atributo de t' con función MAX en V
*/
SELECT * FROM TA_R t WHERE CP(t)=CP(t');
IF No_Data_Found  /* la clave primaria de t'no existe en TA_R */
THEN
   INSERT INTO TA_R (CP_TA_R, DWSum<Atr>_i (i:1..n), DWCount<Atr>_i (i:1..n),
   DWMin<Atr>_i (i:1..n), DWMax<Atr>_i (i:1..n) )
   VALUES(CP(t'), AtrSUM_i(t')(i:1..n), 1 (i:1..n)
   AtrMin_i (i:1..n), AtrMax_i (i:1..n) );
   IF TA_R tiene atributos con prefijo DWMin
   THEN
   INSERT INTO TA_RMinMax (CP_TA_R, AtrSUM_i (i:1..n), AtrCOUNT_i (i:1..n))
   VALUES(CP(t´), DWMin<Atr>,1)
   ENDIF;
   IF TA_R tiene atributos con prefijo DWMax
   THEN
   INSERT INTO TA_RMinMax
   VALUES(CP(t´), DWMax<Atr>,1)
   ENDIF;
ELSE  /* la clave primariade t´ ya existe en TA_R */
   for each aggregate <Atr> in TA_R
      case <Atr> prefix of
         DWSum  : UPDATE TA_R
                   SET DWSum<Atr>=DWSum<Atr>+t´<Atr>
         DWCount: update TA_R set
                        DWCount<Atr>=DWCount<Atr>+1
         DWAvg  : update TA_R set
                       DWAvg<Atr>  =DWAvg<Atr> + t´<Atr>
            /* DWCount es requerido para calcular Avg */
         DWMin, DWMax :
           IF llave primaria de t´ and <valor Atr>
            existe en A_SST TA_R
            THEN
               UPDATE A_SST TA_R SET AtrVeces=AtrVeces+1
                Where llave de t' and
                Atr<valor>=t'.Atr<Valor>
            ELSE
               INSERT INTO A_SST TA_R (llave primaria t´,
                                 nom Atr>,<valor Atr>,1)
               /*actualiza DWMin y DWMax en TA_R*/
              IF DWMin<Atr> greater than t´<Atr>
               THEN
                  UPDATE TA_R SET DWMin<Atr>=t´<Atr>
                 WHERE llave t'
              ENDIF
               IF DWMax<Atr> less than t´<Atr>
               THEN
                  update TA_R set DWMax<Atr>=t´<Atr>
                  Where llave t'
               end if
            end if
      end case
   end for
end if
```

– Si la operación es un borrado en R (*delete)*:

Actaulización de TA_R en caso de delete:

```
for each aggregate <Atr> in TA_R
   case <Atr> prefix of
      DWSum  : update TA_R set
                   DWSum<Atr>=DWSum<Atr>-t´<Atr>
      DWCount: update TA_R set
                   DWCount<Atr>=DWCount<Atr>-1
      DWAvg  : update TA_R set
                   DWAvg<Atr>  =DWAvg<Atr> - t´<Atr>
      DWMin, DWMax :
         /*actualiza la tabla A_SST TA_R*/
         Update A_SST TA_R with AtrVeces=AtrVeces-1 Where
                llave t' and Atr<valor>=t'.Atr<Valor>
         if AtrVeces=0 then
            Delete from A_SST TA_R where llave primaria t´
            and Atr Veces=0 and NomCampo=DWMin<Atr>
            /*actualiza DWMin and DWMax en TA_R*/
            if <Atr>prefix = DWMin then
               Select Min(Atr Value)  into T from A_SST TA_R
               Where llave t' and NomCampo=DWMin<Atr>
               Update TA_R set DWMin<Atr>=T.<Atr Value>
               Where llave t'
            end if
            if <Atr>prefix=DWMax then
               Select Max(Atr Values) into T from A_SST TA_R
               where llave t´and NomCampo=DWMax<Atr>
               Update TA_R Set DWMax<Atr>=T.<Atr Value>
               Where llave t'
            end if
         end if
   end case
end for
```

d) Se verifica si la información almacenada en las tablas auxiliares es suficiente para actualizar el almacén de datos (concatenación de las tablas auxiliares), y se obtiene una tupla TDW con los atributos de la

clave primaria, los atributos actualizables, y los atributos no actualizables y la operación a realizar en el AD).

```
Select * into TDW DW<agregación><Atr> +
DW<agregación><Atr> +…+ DW<agregación><Atr> as acu
from TA_R1 ⋈ TA_R2 ⋈ … ⋈ TA_Rn where TA_R1llave=t'llave AND
 TA_R1llave=TA_R2llave … AND TA_Rn-1llave=TA_Rnllave
If TDW = Ø then
   return /* The Data Warehouse is not updated */
End if
if la operación en R es delete then
   if  TDW.acu = 0 then
      delete from TA_R1 Where TA_R1Key=t'.llave
      delete from TA_R2 Where TA_R2Key=t'.llave
       :
      delete from TA_Rn Where TA_RnKey=t'.llave
   Else
      select * into acumulado DW<agregación><Atr> +
      DW<agregación><Atr> +…+ DW<agregación><Atr> as Suma
      from TA_R1 where TA_R1.llave=t'.llave
      if acumulado.suma=0 then
          delete from TA_R1 where TA_R1Key=t'.Key
      end if
      :
      select * into acumulado DW<agregación><Atr> +
      DW<agregación><Atr> +…+ DW<agregación><Atr> as suma
      from TA_Rn where TA_Rn.llave=t'.llave
      if acumulado.suma=0 then
          delete from TA_Rn where TA_RnKey=t'.llave
      end if
      operación="insert"
       /* el borrado en R es un insert en el almacén */
   End if
End if
```

2.- Actualización del almacén de datos.

Para actualizar el almacén de datos, se recibe la operación de actualización del sistema operacional y una tupla con los atributos: ClavePrimaria, no actualizables y actualizables resultado del paso (d) del procedimiento de actualización de las tablas auxiliares. Este proceso consta de dos acciones:

```
/* TDW, tupla de la tabla actualización*/
Select * into S from Tabla Control
Sesion=S.CurrentVN
if Clave primaria(TDW) no existe en tablaH then
   insert into TablaH(llave primaria TDW,
   TDW<Atr> atributos no modifi, Sesion)
   insert into TablaD(Clave primaria TDW,TDW<Atr>
   atributos modificables,Sesion, 2147000000,
   'insert')
Else
   select VnInicio,VnFin into IniFin from TablaD
   where Clave primaria=Clave primaria(TDW) and
    VnFin=2147000000
   if IniFin.VnInicio=Sesion then
      update TablaD  set <Atr>modificables=
      TDW<Atr>modificables, operacionVN=
      TDW<Atr>operacion
   else
      update TablaD with VNFin=Sesion-1 where
      Clave primaria=llave primaria TDW and
      VnFin=2147000000
      insert into TableD(Clave primaria(TDW),
      TDW<Atr> atributos modificables, sesion
      2147000000, TDW<Atr>operacion)
   end if
   update TablaH with UltimoVN=Sesion
   update control tabla with MaintenanceActive=true
end if
```

En las figuras Figura 57, Figura 58, Figura 59, Figura 60, Figura 61, Figura 62 y Figura 64 se detalla como el algoritmo αNVNLTR irá modificando la vista materializada del almacén de datos sobre el sistema universitario presentado en la Figura 54. Se supone que las siguientes operaciones de mantenimiento se realizaron en el almacén de datos:

a) Insertar en Docencia la tupla <20, 5, 10>

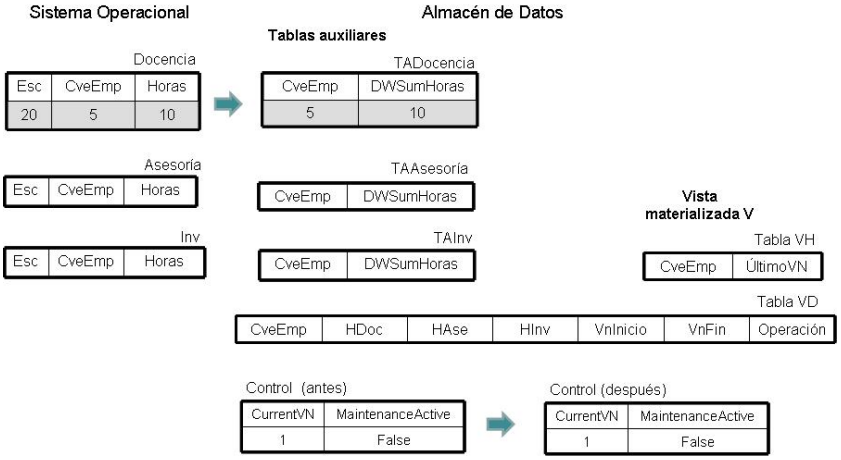


Figura 57. Estado del AD después de la inserción en Docencia.

b) Insertar en Investigación la tupla <20, 5, 5>

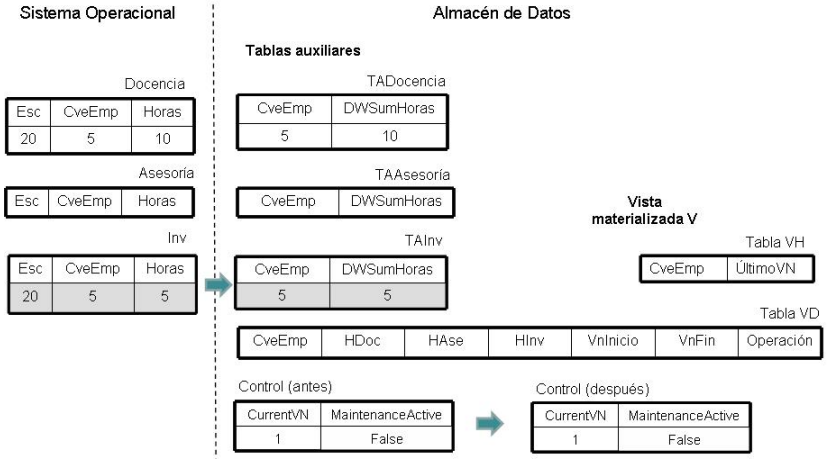


Figura 58. Estado del AD después de la inserción en Inv.

c) Insertar en Asesoría la tupla <20, 5, 3>

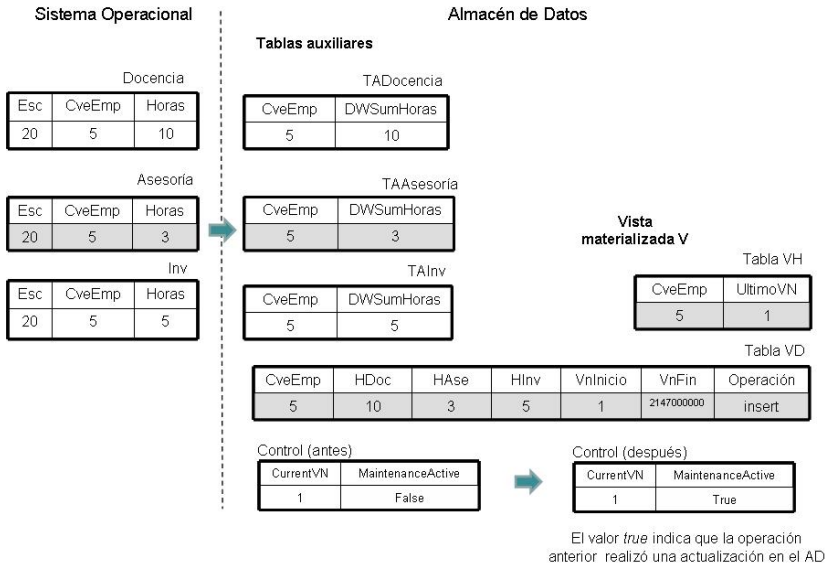


Figura 59. Estado del AD después de la inserción en Asesoría.

d) Insertar en Asesoría la tupla <30, 5, 4>

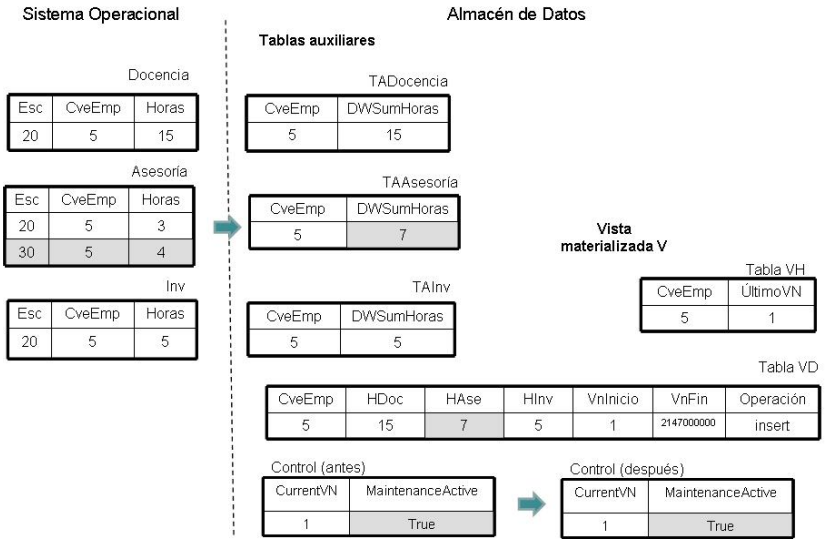


Figura 60. Estado del AD después de la inserción en Asesoría.

e) Borrar en Asesoría la tupla <20, 5, 3>

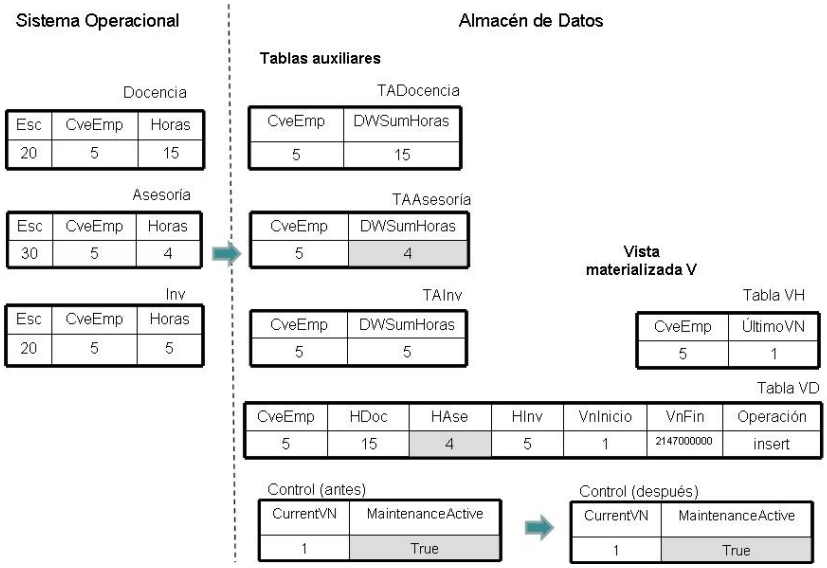


Figura 61. Estado del AD después del borrado en Asesoría.

f) Borrar en Asesoría la tupla <30, 5, 4>

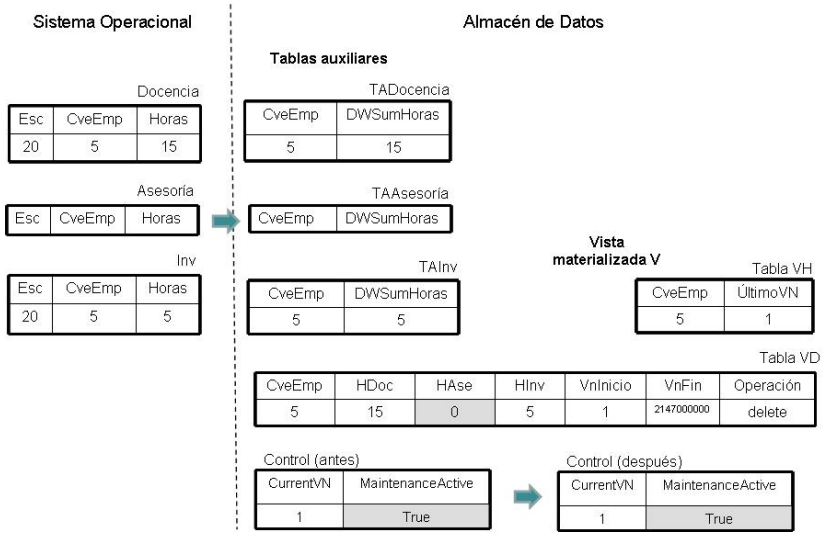


Figura 62. Estado del AD después del borrado en Asesoría.

g) Un analista realiza una consulta al almacén de datos.

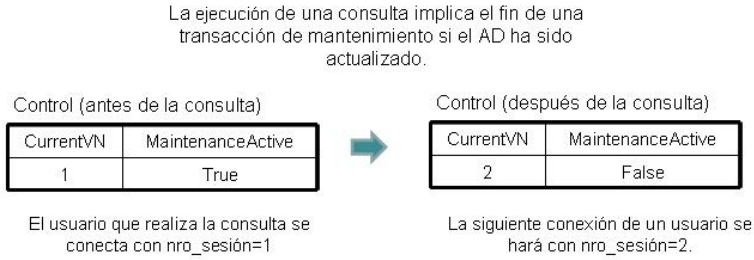


Figura 63. Estado actual de la tabla de control después de la consulta.

h) Insertar en Asesoría la tupla <30, 5, 15>

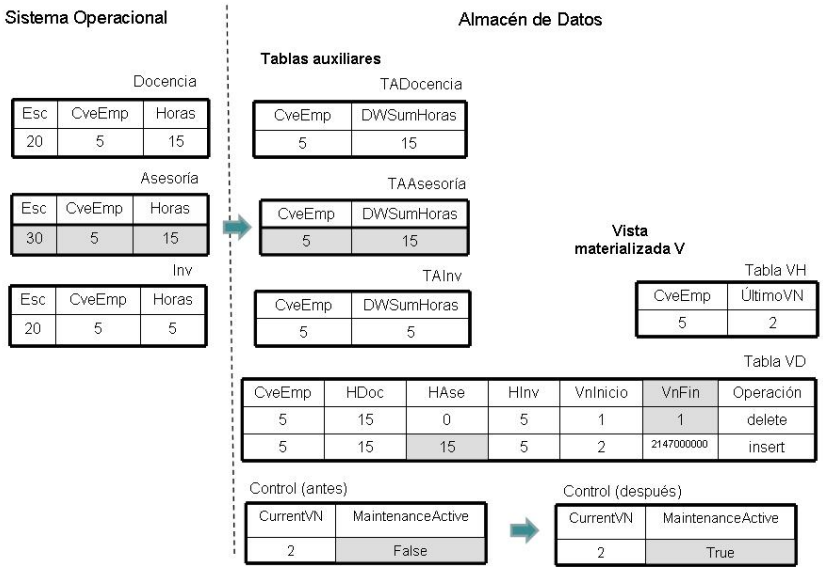


Figura 64. Estado del AD después de la inserción en Asesoría.

i) Después de la actualización, el valor de la tabla de control será el siguiente.

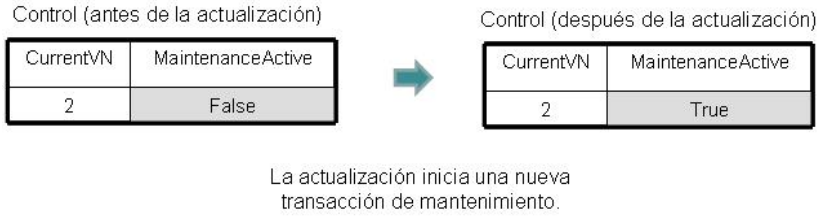


Figura 65. Estado actual de la tabla de control después de la inserción.

Como se puede observar en la Figura 65 la tabla de control no se actualiza cuando se realiza una actualización al almacén de datos. La Figura 63 muestra la tabla de control actualizada cuando se cumplen los dos criterios que indican que una transacción de mantenimiento ha finalizado: un usuario ha consultado el almacén, y antes de la consulta se ha producido una actualización del almacén de datos.

Capítulo 6. Sistema de mantenimiento de almacenes de datos.

Como resultado de la investigación realizada, presentada en los capítulos anteriores, se desarrolló un sistema de mantenimiento de almacenes de datos [76].

6.1 Descripción del sistema de mantenimiento.

Cuando el sistema se ejecuta muestra la interfaz de la Figura 66.

Figura 66. Interfaz de inicio del sistema de mantenimiento

La interfaz principal del sistema de mantenimiento, cuenta con ocho opciones, en este momento cinco opciones están inhabilitadas, debido a que aún no se ha seleccionado el nombre del Data Mart en el que se desea trabajar. La primera opción que debe seleccionar el administrador del almacén de datos es la de Data Mart y con ello mostrará la interfaz de la Figura 67.

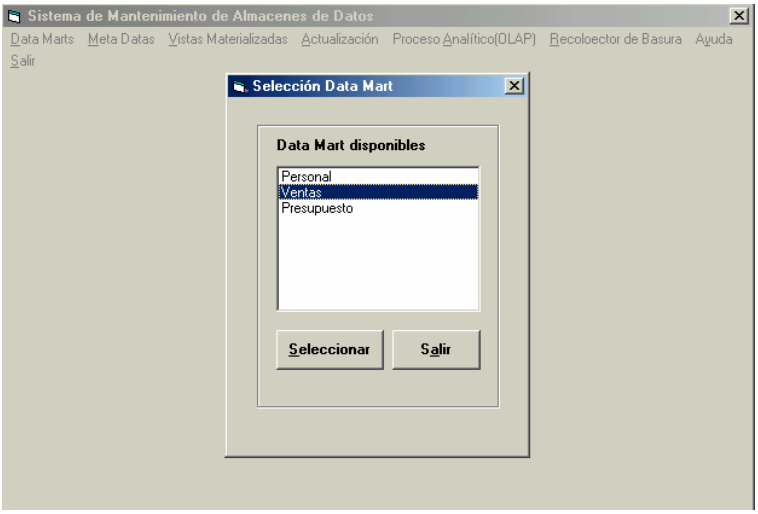


Figura 67. Selección del Data Mart

Observe en la Figura 67 que el sistema detectó que existen tres nombres que representan Data Marts, se selecciona Ventas, generando con esto que las ocho opciones del sistema estén habilitadas como se muestra en la Figura 68.

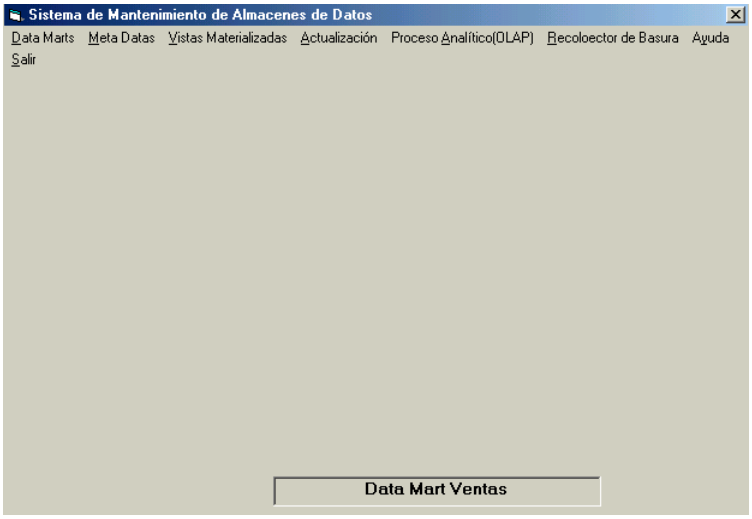


Figura 68. Sistema de mantenimiento

En la parte inferior de la Figura 68 se muestra el nombre del Data Mart que está habilitado. Para que la opción de actualización se pueda ejecutar correctamente, el administrador del sistema debe ejecutar las opciones de Meta Datos (descritas en la sección 3.4.3) y de Vistas Materializadas (Figura 69).

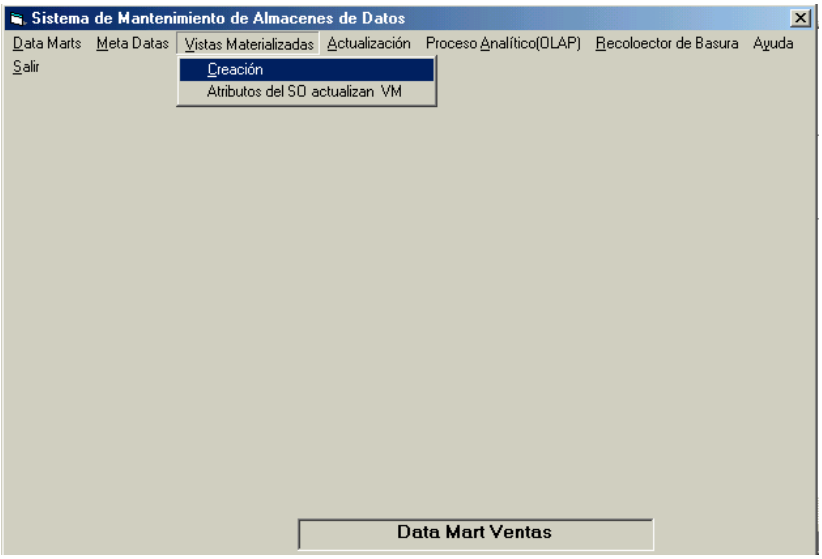


Figura 69. Opciones de Vistas Materializadas

En la Figura 69 se muestras las opciones para:

- **Creación** de vistas materializadas, haciendo uso del esquema propuesto en la sección 5.2.1, se muestra en la Figura 70 y la Figura 71.
- **Atributos del SO actualizan la VM**, para realizar la actualización del almacén de datos con la información relevante de los sistemas operacionales, es necesario definir la asociación de atributos como se muestra en la Figura 72.

Figura 70. Definición del esquema de la VM (primera parte)

En la Figura 70 se muestra la definición de la tabla TablaH que forma la primer parte de la vista materializada. La definición incluye los atributos de clave primaria (prefijo key), atributos no modificables (no existen en este ejemplo) y el atributo de control (UltimoVn), éste es asignado de manera automática al esquema de TablaH. El botón de "Generarla" se utiliza para crear la tabla en la base de datos correspondiente al Data Mart seleccionado. El botón "Eliminar" borra físicamente la tabla del Data Mart. La definición de la segunda parte de la vista materializada se realiza como se muestra en la Figura 71.

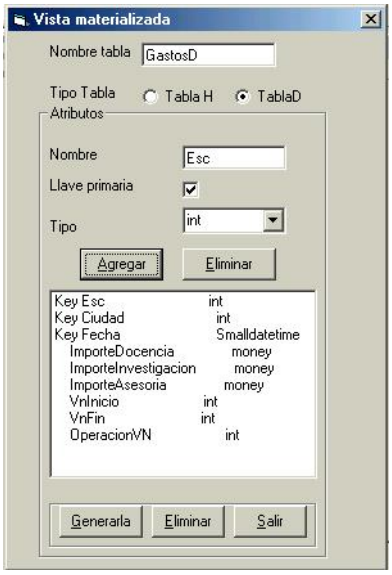


Figura 71. Definición del esquema de la VM (segunda parte)

En esta interfaz se muestra la definición de la tabla TablaD. La definición incluye los atributos de clave primaria (prefijo key), atributos modificables (ImporteDocencia, ImporteInvestigacion e ImporteAsesoria) y los atributos de control (VnInicio, VnFin y OperacionVN), asignados automáticamente a TablaD. El botón de "Generar" se utiliza para generar la tabla en la base de datos correspondiente al Data Mart seleccionado. El botón "Eliminar" borra físicamente la tabla de la base de datos que representa el Data Mart.

Para realizar la actualización de la vista materializada es necesario establecer una asociación entre los atributos de la tabla operacional que contiene las tuplas relevantes y los atributos de la vista materializada (Figura 72).

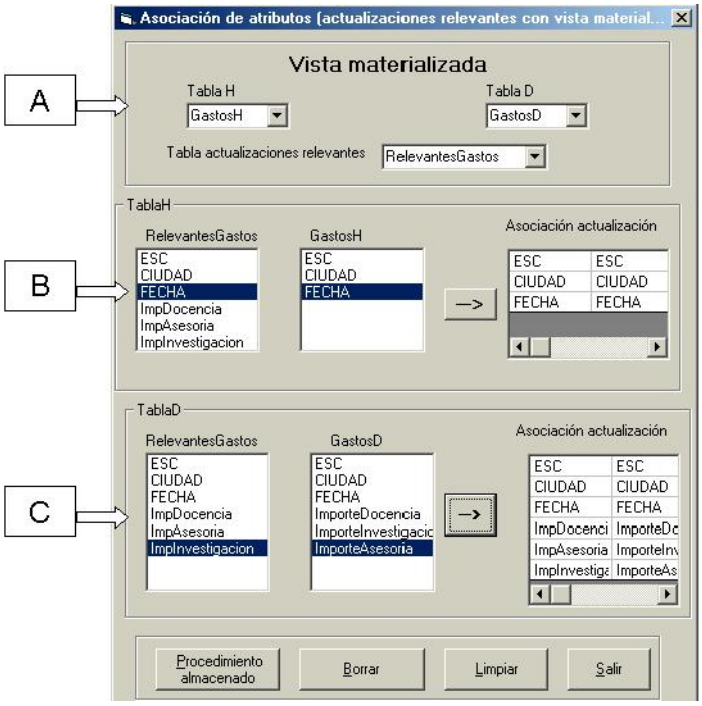


Figura 72. Asociación de atributos

En la Figura 72 se muestra como establecer la asociación de los atributos de la tabla operacional que contiene las tuplas relevantes (TR) y la vista materializada. La Figura 72 está dividida en tres secciones:

Sección A: permite seleccionar el nombre de la vista materializada (TablaH y TablaD) y el nombre de la tabla TR.

Sección B: muestra los atributos de la tabla TR y los atributos de la tabla TablaH.

Sección C: muestra los atributos de la tabla TR y los atributos de la tabla TablaD.

Para establecer qué atributo de la tabla TR actualizará un atributo de la tabla TablaH, en la sección B, se pulsa el botón izquierdo del ratón en el atributo de la tabla TR y se pulsa el botón izquierdo del ratón sobre el atributo de la tabla TablaH y, para establecer la asociación pulse el botón izquierdo del ratón sobre el botón que muestra la flecha, los dos atributos seleccionados (asociados) pasarán a la derecha de la pantalla, indicando la asociación de los atributos. Para realizar la asociación de los atributos de la tabla TR y la tabla TablaD, el proceso es similar al descrito anteriormente, sólo debe realizarlo sobre la sección C.

Una vez establecida la asociación de atributos entre la vista materializada y la tabla TR, se pulsa el botón de "Procedimientos almacenados" y de manera automática se generan un procedimiento almacenado con el nombre formado por el prefijo Sp y Nombre de la vista materializada. En la Figura 73 se muestra el procedimiento almacenado generado automáticamente, para la vista materializada que se usó Figura 72.

```
CREATE procedure SP_Gastoss
  @Resultado  int output
As
/* Definición de Variables para recuperar los atributos de Control*/
Declare @Sesion int, @MantenimientoActivo tinyint, @VnInicio int,@VnFin int, @UltimoVN int
Select @Sesion=0
select @Sesion=CurrentVn,@MantenimientoActivo=MaintenanceActive From ControlGastos
If @Sesion=0
Begin
   select @Resultado=0
End
ELSE
Begin
   Declare ALMACEN CURSOR For select Fecha,ImpDocencia,ImpAsesoria,ImpInvestigacion,Esc,
                                     Ciudad,Operacion from
   RelevantesGastos
   declare @Fecha datetime
   declare @ImpDocencia float
   declare @ImpAsesoria float
   declare @ImpInvestigacion float
   declare @Esc int
   declare @Ciudad int
   declare @Operacion int
   open ALMACEN fetch next from ALMACEN into  @Fecha,@ImpDocencia,@ImpAsesoria,
   @ImpInvestigacion, @Esc, @Ciudad,@Operacion
   while (@@FETCH_STATUS=0)
   BEGIN
      Select @UltimoVN=0
      Select @UltimoVN=UltimoVN from GastosH where Esc=@Esc And Ciudad=@Ciudad
      And Fecha=@Fecha
      If @UltimoVN=0
      Begin
         Insert Into GastosH(ESC,Ciudad,Fecha,UltimoVN) Values (@ESC,@Ciudad,@Fecha,@Sesion )
         Insert Into GastosD (ESC, Ciudad, Fecha, ImporteDocencia, ImporteAsesoria,
         ImporteInvestigacion,Vninicio,Vnfin,OperacionVN) Values  (@ESC,@Ciudad, @Fecha,
         @ImpDocencia, @ImpAsesoria, @ImpInvestigacion,@Sesion,2147000000,@Operacion )
      End
      Else


      Begin
         select @VnInicio=0
         Select @VnInicio=VnInicio,@VnFin=VnFin from GastosD Where Esc=@Esc And
         Ciudad=@Ciudad And Fecha=@Fecha And VnFin=2147000000
         If @VnInicio=0
         Begin
            Select @Resultado=0
         end
         Else
         Begin
            If @VnInicio=@Sesion
            Begin
               Update GastosD set ESC=@ESC, Ciudad=@Ciudad, Fecha=@Fecha,
               ImporteDocencia=@ImpDocencia, ImporteAsesoria=@ImpAsesoria,
               ImporteInvestigacion=@ImpInvestigacion, OperacionVN= @Operacion
               Where Esc=@Esc And Ciudad=@Ciudad And Fecha=@Fecha And VnFin=2147000000
            End
            Else
            begin
               Update GastosD set VnFin=@Sesion-1 Where Esc=@Esc And Ciudad=@Ciudad And
               Fecha=@Fecha And VnFin=2147000000
               Insert Into GastosD (ESC, Ciudad, Fecha, ImporteDocencia, ImporteAsesoria,
               ImporteInvestigacion, Vninicio,Vnfin,OperacionVN)Values  (@ESC, @Ciudad,
               @Fecha, @ImpDocencia, @ImpAsesoria,@ImpInvestigacion, @Sesion,2147000000,
               @Operacion )
            End
         End
      End
      fetch next from ALMACEN into @Fecha,@ImpDocencia,@ImpAsesoria,@ImpInvestigacion,@Esc,
      @Ciudad,@Operacion
   End
   Update GastosH Set UltimoVN=@Sesion Where Esc=@Esc And Ciudad=@Ciudad And Fecha=@Fecha
   Update Control set MaintenanceActive=1 /* se ha realizado una actualizacion al DW*/
 eNd
select @resultado=1
 CLOSE ALMACEN
deallocate ALMACEN
```

Figura 73. Procedimiento almacenado

Con el procedimiento almacenado generado y estando preparadas las actualizaciones relevantes, las actualizaciones pueden ser incorporadas (batch o lote) a la VM, seleccionando el nombre del procedimiento almacenado que se desea ejecutar (Figura 74).

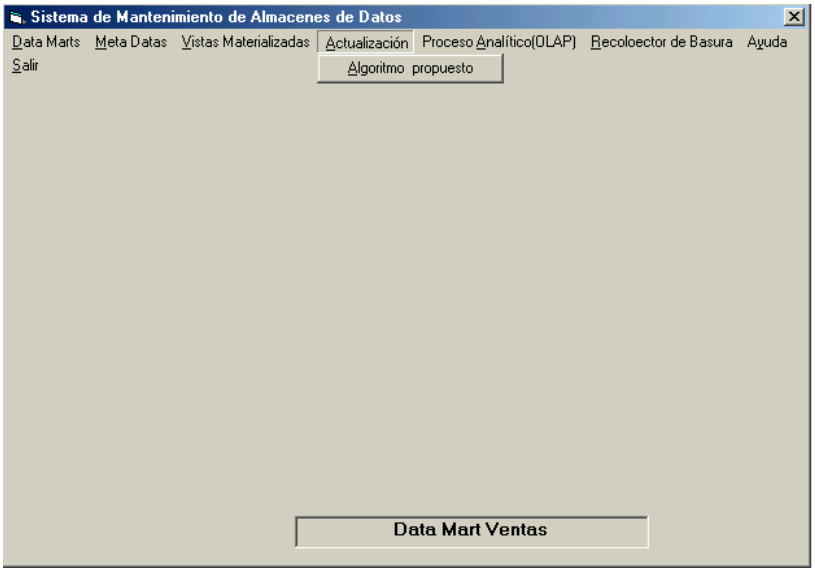


Figura 74. Selección del procedimiento almacenado a ejecutar

El sistema cuenta con una opción que permite realizar proceso analítico, esto se logra a través de la herramienta Bussines Object (Figura 75).

Figura 75. Proceso analítico

En la propuesta de mantenimiento se estableció que se requiere un recolector de basura que permita eliminar las versiones de las tuplas de sesiones que ya no van a ser consultadas (Figura 76).

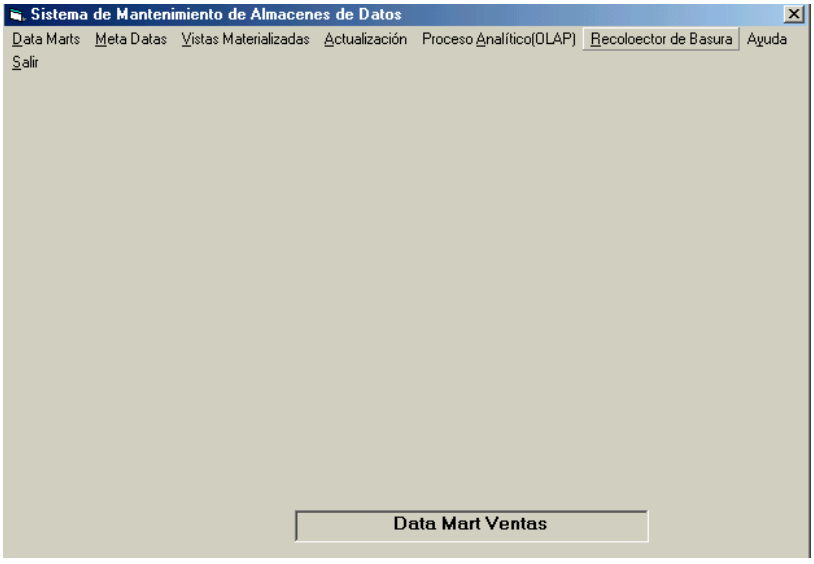


Figura 76. Recolector de basura

6.2 Pruebas realizadas.

Para evaluar la eficiencia del algoritmo **αVNL** (propuesto en este trabajo, sección 5.2.1), con respecto a su antecesor el algoritmo NVNL (con un esquema con valor de n igual a 10), se ha implementado la propuesta de actualización en un prototipo desarrollado en SQL Server Personal Edition (Microsoft). El hardware utilizado es un HP AMD Turion[64] a 1.61 GHz con 2 GB de memoria principal (RAM).

Continuando con el experimento se ha usado un almacén de datos con esquema en estrella (Figura 77), donde se almacena información referente a importes gastados en docencia, investigación y asesoría en las diferentes escuelas de la Universidad.

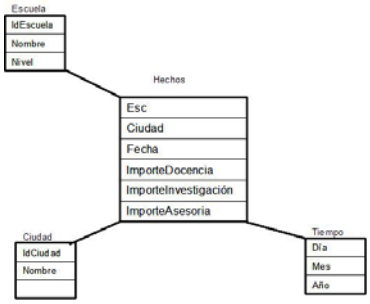


Figura 77. Esquema multidimensional en estrella

La tabla de hechos mostrada en la Figura 77, se definió como una vista materializada (GastosH, GastosD) en la sección 6.1. Recuérdese que el algoritmo **αVNL** realiza la actualización en batch, para lograr esto se definió la tabla que contiene las actualizaciones relevantes al almacén (RelevantesGastos). Los datos de la tabla RelevantesGastos fueron generados al azar.

6.2.1 Carga inicial de los datos.

Ambos algoritmos permiten realizar el proceso de carga inicial de los datos; se realizaron diferente pruebas (con diferente cantidad de tuplas) dando los resultados que se muestran en la Figura 78.

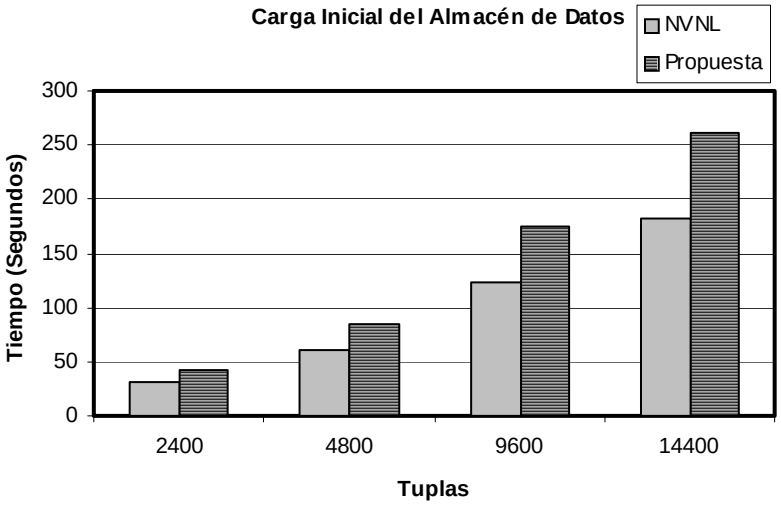


Figura 78. Tiempos obtenidos para los algoritmos NVNL y αVNL **en la carga inicial de datos**

En esta gráfica se ilustra claramente que el algoritmo antecesor (NVNL) tiene un mejor desempeño que el propuesto (**αVNL**), esto es debido a que el algoritmo antecesor sólo hace una inserción ya que la vista materializada para este esquema sólo requiere de una tabla y en la propuesta se requiere de dos inserciones debido a que utiliza dos tablas para representar la vista materializada.

6.2.2 Mantenimiento del almacén de datos

- Se realizaron cuatro transacciones de mantenimiento con diferente cantidad de tuplas a propagar en el almacén, obteniendo los resultados mostrados en la Figura 79.

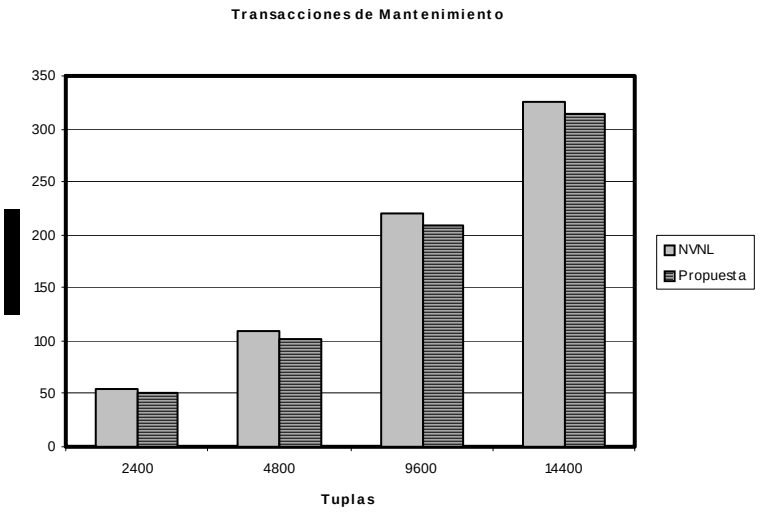


Figura 79. Tiempos obtenidos por los algoritmos NVNL y αVNL en actualización del almacén de datos

En la Figura 79 se puede observar que el tiempo requerido para realizar las 4 transacciones de mantenimiento, es menor en el algoritmo propuesto que en su antecesor.

- Se ejecutaron cinco transacciones de mantenimiento con 14,400 actualizaciones, obteniendo los tiempos que se muestran en la Figura 80.

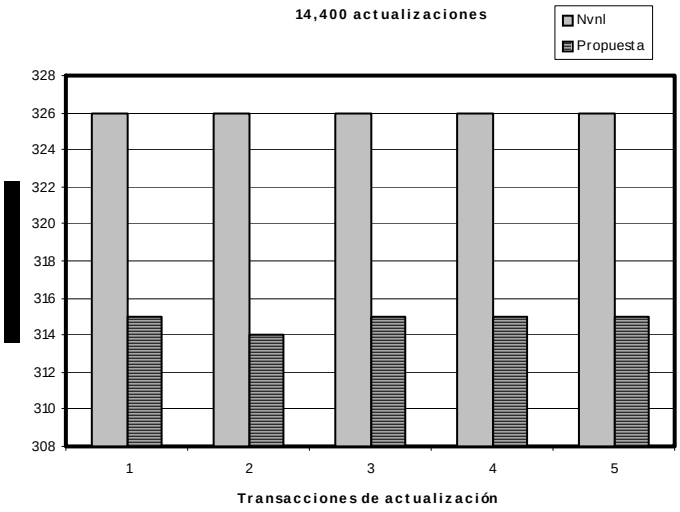


Figura 80. Tiempos obtenidos para los algoritmos NVNL y αVNL en actualización del almacén de datos.

- Once veces se propagaron 2400 actualizaciones al almacén de datos, generando los siguientes tiempos.

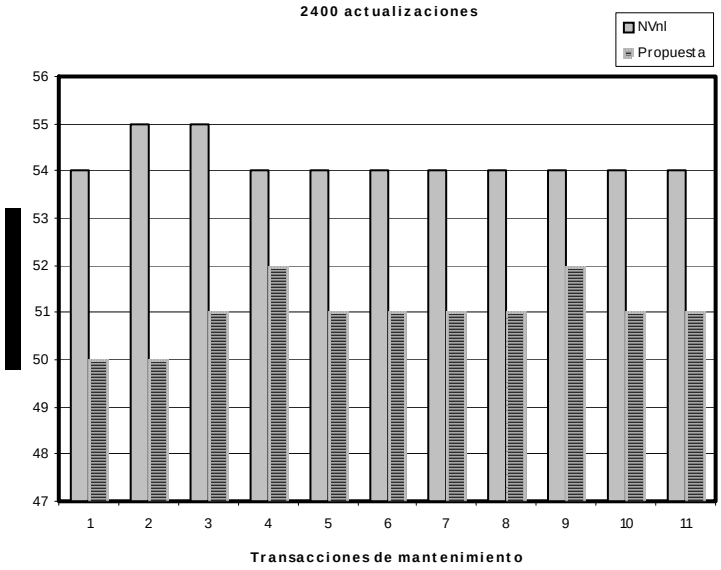


Figura 81. Tiempos obtenidos por los algoritmos NVNL y αVNL en actualización del almacén de datos

Aquí se ilustra claramente que la propuesta realizada tiene un mejor tiempo de ejecución ya que en todas las propagaciones resultó con un tiempo menor a la ejecución del algoritmo antecesor. El algoritmo propuesto después de haber realizado las once transacciones de actualización, garantiza consultas consistentes mientras que el algoritmo NVNL ya no puede garantizarlo.

Los tiempos del algoritmo antecesor se degradan más, cuando el valor de N es incrementado, recuerde que el valor de N define el tamaño del esquema de la vista materializada.

Capítulo 7. Caso de estudio.

En este capítulo se muestra el proceso que se realiza para actualizar el almacén de datos con nuestra primer propuesta de mantenimiento de almacenes de datos en batch, iniciando desde la extracción de los datos de los sistemas operacionales hasta la actualización del almacén de datos.

7.1 Introducción.

La Universidad Autónoma de Sinaloa (UAS), como institución pública y autónoma de educación superior esta distribuida en el estado de Sinaloa como se muestra en la Figura 82, cubriendo las cuatro zonas geográficas que lo forman; Norte, Centro-Norte, Centro y Sur.

Figura 82. Estado de Sinaloa

La oferta educativa de la UAS cubre los niveles medio superior, medio profesional, técnico superior universitario, profesional; posgrado y enseñanzas especiales. Actualmente la UAS cuenta con una población de 102,952 alumnos inscritos, distribuidos como lo indica la Tabla 40.

Nivel	No.Alumnos
Posgrado	1,268
Licenciatura	40,737
Técnico Superior Universitario	468
Medio Profesional	2,296
Bachillerato	41,704
Enseñanzas Especiales	15,779
Total	102,952

Tabla 40. Distribución de la población estudiantil de la UAS

La UAS cuenta con 4386 docentes, de los cuales 2098 tiene puesto de tiempo completo, lo que los obliga a realizar una jornada de trabajo de 30 horas semanales. El resto de los maestros son de asignatura que van de 5 a 30 horas semanales. Los docentes justifican su puesto principalmente con tres tipos de actividades; docencia, investigación y asesoría.

7.2 Requisitos

Una de las principales necesidades expresada por las autoridades de la UAS, es conocer cuanto se invierte del presupuesto asignado a la UAS

en los conceptos de docencia, investigación y asesoría en cada escuela que la forman. Además una incógnita es saber si lo invertido en estos rubros es representativo al giro de trabajo que existe en las ciudades donde se encuentran las diferentes escuelas. Constantemente se escuchan preguntas tales como, ¿Qué importe se está invirtiendo en investigación en las escuelas que están en ciudades cuyo giro de trabajo es la industria?, ¿Escuelas de la zona Sur donde se esta invirtiendo más en asesoría que en investigación?.

7.3 Elementos con los que se cuenta

Actualmente la UAS cuenta con la emisión de nómina y la carga laboral del personal docente.

7.3.1 Nómina

La UAS emite su nómina en periodos quincenales y mantiene un archivo histórico del detalle de las emisiones, haciendo uso de las relaciones básicas que se muestran en la Tabla 41.

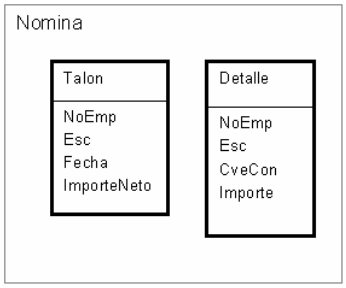


Tabla 41. Esquema de las tablas de Nómina

En la tabla de detalle cada empleado cuenta con un mínimo de cinco tuplas por cada pago realizado (al mes se realizan dos pagos), produciéndose 21,930 tuplas por cada pago.

Diccionario de datos

Campo	Significado
NumEmp	Número de empleado
Esc	Clave de Escuela
Fecha	Fecha de emisión
ImporteNeto	Importe real que percibio el empleado
CveCon	Clave de concepto que se aplica (valor de 1-500 indican percepciones y de 500-999 deducciones
Importe	Importe del concepto

Tabla 42. Diccionario de datos para las tablas de Nómina

7.3.2 Carga laboral

En el periodo que existe entre la finalización del semestre e inicio del próximo semestre se realiza un proceso de definición de carga laboral de cada docente (durante el semestre frecuentemente se modifica la carga de un gran porcentaje de docentes) realizado por una comisión sindical y el director de cada escuela, el cual se ilustra en el siguiente caso de uso.

CARGA LABORAL DEL PERSONAL DOCENTE

CASO DE USO:	Carga laboral del semestre
ACTORES:	Director, Sindicato (iniciador), Departamento Contraloría

DESCRIPCIÓN:	Un representante del sindicato llega con el Director de la escuela o facultad a elaborar o modificar la carga de trabajo de un académico. El departamento registra la carga laboral.	
ESPECIFICACIÓN DEL CASO DE USO		
GENERAL	**COMUNICACIÓN ACTOR/SISTEMA**	**RESPUESTA DEL SISTEMA**
1. Este caso de uso inicia cuando un representante del sindicato llega con el director a realizar o modificar la carga laboral de un profesor.		
2. La carga laboral es enviada al departamento de contraloría académica y se entrega una copia a cada encargado de área (académica, investigación, asesoría), los cuales las codifican.		
	3. Cada encargado inicia una nueva sesión de captura de carga laboral, introduciendo número de empleado y clave de escuela.	
	.	4. El Sistema crea una nueva carga (si el empleado ya existe en esa escuela el sistema lo borra).
	5. El encargado de cada área captura número de horas asignadas.	
		6. El sistema

		almacena los datos de la carga
7. Cada encargado archiva la copia de la carga del docente. El caso de uso termina		

Debido al proceso anterior, constantemente se modifica la información de la carga laboral del personal docente de la UAS, la información se tiene representada como se muestra en la Tabla 43.

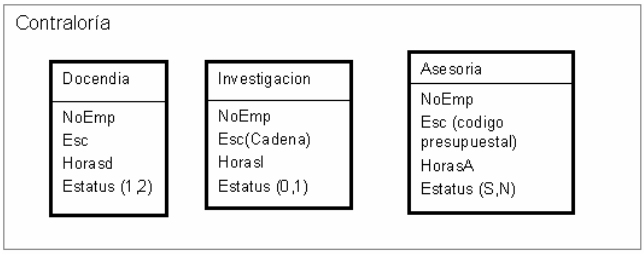


Tabla 43. Esquema de las tablas de la Carga Laboral del docente

Diccionario de Datos

Campo	Significado
NoEmp	Número de Empleado
Esc	Clave de la escuela
Estatus	Clave que indica si un docente esta activo o inactivo en la UAS
Horasd	Número de horas a la semana de docencia
Horasl	Número de horas a la semana de Investigación
HorasA	Número de horas a la semana de Asesoria

Tabla 44. Diccionario de datos de las tablas de Carga Laboral

7.4 Solución

En el almacén de datos se crea una base de datos con el nombre CtlGastos, la que contendrá las vistas materializadas, los metadatos y los procedimientos almacenados necesarios para realizar la actualización.

7.4.1 Esquema multidimensional

Para dar solución a los requerimientos de las autoridades universitarias se propone representar la información con en el esquema multidimensional que muestra la. Figura 83.

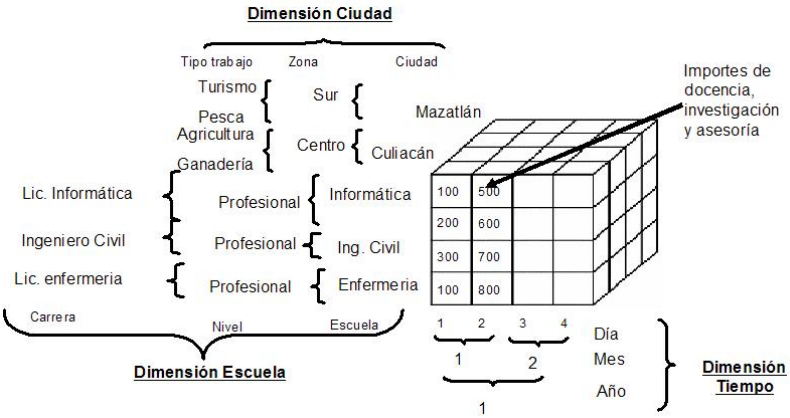


Figura 83. Esquema multidimensional

El esquema multidimensional representa información resumida (importes invertidos en docencia, Investigación y Asesoría) en cada escuela y ciudad, haciendo uso de una granularidad por día.

7.4.2 Vista Materializada

El esquema multidimensional se representa a través de una vista materializada (Figura 84), la cual tendrá como nombre GastosH y GastosD

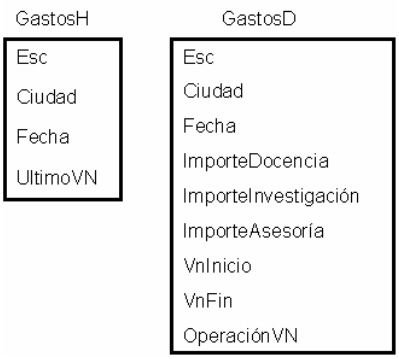


Figura 84. Representación de la vista materializadas del caso de estudio

La tabla Gastos contendrá 108 tuplas quincenales, donde resume las 21930 tuplas generadas en la emisión de la nómina.

7.4.3 Metadatos

Para poder hacer uso de nuestra propuesta de solución de mantenimiento de almacenes de datos es necesario definir los metadatos que se muestran en la Tabla 45, Tabla 46 y Tabla 47.

7.4.3.1 Formatos

Para efecto de este caso de estudio, es necesario poner en el metadato (Tabla 45) el campo "Esc" de las tablas; "Investigación" y "Asesoría", las cuales se encuentran en el sistema operacional "Contraloría". A este campo se le debe cambiar el formato de cadena a numérico.

Sistema Operacional	Tabla	Atributo	Tipo de dato	Tamaño en Bytes	Tipo de dato en el AD	Tamaño en Bytes en el AD
Contraloria	Investigacion	Esc	Char	3	int	2
Contraloria	Asesoria	Esc	Char	7	long	4

Tabla 45. Definición del metadato Formatos

7.4.3.2 Operación

En el metadato que se muestra en la Tabla 46 se describen dos campos del sistema operacional "Nomina", de la tabla "Talon", el campo "ImporteNeto", el cual es una cantidad entera que contiene los dos dígitos que representan los centavos, como ejemplo en el sistema operacional viene esta importe 158689 que para efecto de almacenarlo en al almacén debe representarse como 1586.89.

Sistema Operacional	Tabla	Atributo	Expresión aritmética
Nomina	Talon	ImporteNeto	ImporteNeto/100
Nomina	Detalle	Importe	Importe/100

Tabla 46. Definición del matadato Operaciones Aritméticas

7.4.3.3 Limpieza de datos

En el metadata que se muestra en la Tabla 47 contiene las diferentes escuelas que requieren cambiar el código presupuestal (codificado en la tabla de asesoría) por la clave de escuela.

Sistema Operacional	Tabla	Atributo	Valor Original	Valor para el AD
Contraloria	Asesoria	Esc	0711001	10
Contraloria	Asesoria	Esc	0710010	20
Cantroloria	Asesoria	Esc	0731001	30
Contraloria	Asesoria	Esc	0732000	40
Contraloria	Asesoria	Esc	0740009	50
Contraloria	Asesoria	Esc	0720001	60
•				
•				
•				

Tabla 47 Definición del matadata Limpieza

7.4.4 Proceso de solución

En las Figura 85, Figura 86 y Figura 87 se muestra el proceso de obtención de tuplas relevantes desde los sistemas operacionales hasta la actualización del almacén de datos. La Figura 85 muestra el proceso de extracción e integración de datos de las tablas (Talon y

Detalle del sistema de nóminas) del sistemas operacional, la información extraída e integrada se graba en la tabla Pagos.

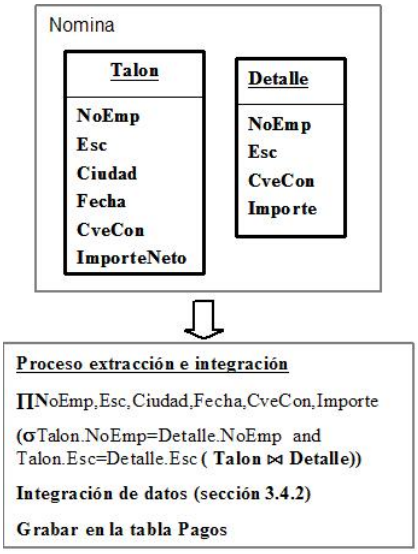


Figura 85. Proceso de extracción e integración de datos de la Nomina

La Figura 86 muestra el proceso de extracción de las tablas Docencia, Investigación y Asesoria.

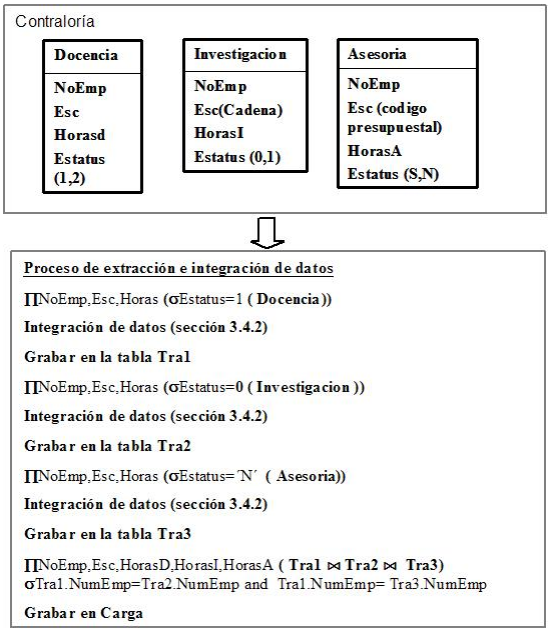


Figura 86. Proceso de extracción e integración de datos de Carga Laboral

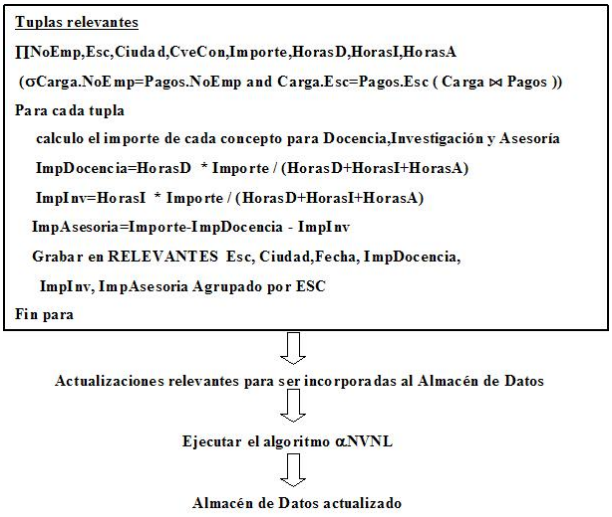


Figura 87. Creación de la tabla de tuplas relevantes y actualización del almacén de datos

El esquema de las tablas requeridas en el proceso de extracción se muestra en la Tabla 48

Pagos
NoEmp
Esc
Ciudad
Fecha
CveCon
Importe

Carga
NumEmp
Esc
HorasD
HorasI
HorasA

Relevantes
Esc
Ciudad
Fecha
ImpDocencia
ImpInv
ImpAsesoria

Tabla 48. Esquema de las tablas necesarias en el proceso de extracción

7.4.5 Puntos de vista con los que se puede analizar

En la Figura 88 se representa la actividad Gastos como un esquema multidimensional en estrella.

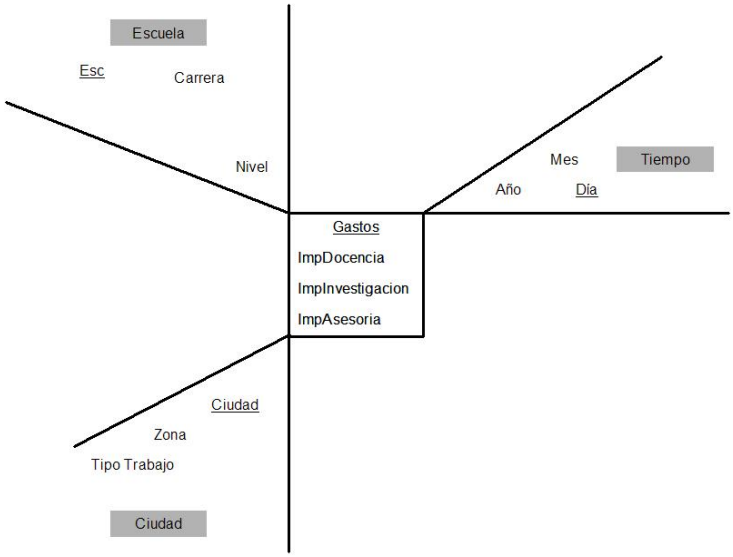


Figura 88. Esquema multidimensional en estrella (actividad Gastos)

7.4.6 Jerarquía de las dimensiones

Los diferentes niveles de agregación que se pueden aplicar a las medidas del esquema multidimensional se ilustran en la Figura 89.

Figura 89. Jerarquías de las dimensiones en la actividad Gastos

7.4.7 Asociación de actualización

La asociación de atributos entre la tabla de actualizaciones relevantes y la vista materializada se muestra en la Figura 90 y Figura 91.

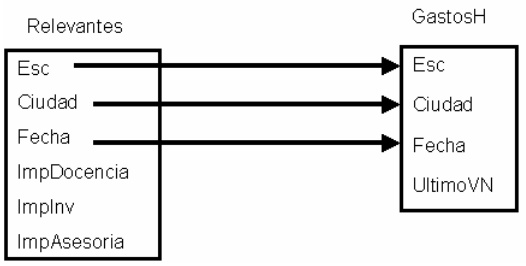


Figura 90. Asociación de atributos TR y GastosH

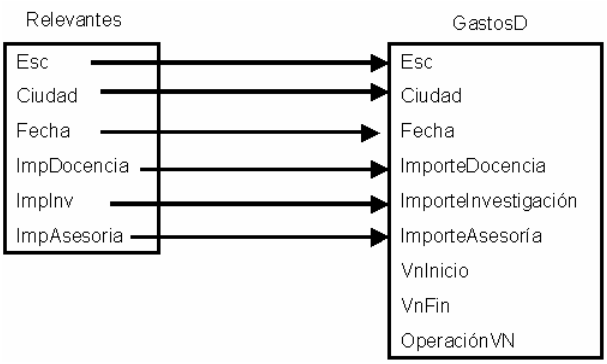


Figura 91. Asociación de atributos TR y Gastos

Capitulo 8. Conclusiones y trabajos futuros.

Con el desarrollo y uso extendido de la tecnología de bases de datos, las organizaciones han ido incrementando las exigencias de funcionalidad de sus sistemas de información. Del uso exclusivo de sistemas de información para la gestión, los usuarios han pasado a demandar sistemas de información de apoyo a la toma de decisiones. Es el origen de la tecnología de almacenes de datos.
Este nuevo tipo de sistemas de información ha planteado nuevos retos en el campo de las bases de datos, y replanteado algunos ya existentes como son la integración de datos y la actualización de vistas materializadas. Es en este ámbito en el que se sitúan las aportaciones de este trabajo de tesis.

8.1 Conclusiones.

En este trabajo de tesis se abordan dos problemas de gran importancia en el ámbito de los almacenes de datos:

- La integración de los datos.
- La actualización del almacén de datos.

Respecto al primer problema se ha desarrollado un prototipo que permite realizar la integración de datos procedentes de diferentes sistemas operacionales. En el método se contempla la aplicación de tres tipos de transformaciones sobre los datos: transformación de formato, limpieza y operaciones aritméticas (Capítulo 3).

En relación con el segundo problema se han hecho las siguientes aportaciones:

1. Se han propuesto dos métodos (Capítulo 5) para la actualización de almacenes de datos. Las características de estos métodos son:
    - se realiza una actualización incremental del almacén.
    - la actualización es en línea: el almacén de datos está siempre disponible para los analistas.
    - se garantiza a los usuarios consultas consistentes: los usuarios acceden a la misma versión del almacén durante su sesión de trabajo.

   El primer método actualiza el almacén de datos a partir de una transacción en batch (o lote), que envía al almacén las actualizaciones ya preparadas para ser aplicadas.

   El segundo método actualiza el almacén de datos a partir de las actualizaciones sobre las relaciones básicas (fuentes operacionales) que son enviadas en tiempo real al almacén. El método hace un auto-mantenimiento débil para un patrón de vistas materializadas de tipo SPJ (del inglés Select-Project-Join) con funciones de agregación (Sum, Count, Avg, Max, Min).

2. Se ha desarrollado una herramienta (Capítulo 6) que permite aplicar las ideas desarrolladas en esta tesis a través de las siguientes opciones:

    - Definición de las transformaciones que requieren los datos para su integración en el almacén de datos.
    - Definición del esquema de la vista materializada.

- Definición de la correspondencia entre atributos (operacionales-almacén de datos) para la actualización de la vista materializada.
- Generación automática de los procedimientos almacenados que realizan la actualización de las vistas materializadas.

3. Pruebas de rendimiento del método que realiza el mantenimiento en batch (o lote) con respecto al método antecesor (algoritmo de Quass), demostrándose que el algoritmo propuesto en este trabajo tiene un rendimiento mejor que el algoritmo antecesor (Capítulo 6).

4. Un caso de estudio donde se muestra el proceso completo que debe realizarse para crear y mantener actualizado un almacén de datos (Capitulo 7).

8.2 Publicaciones relacionadas.

- "Una propuesta de solución para realizar Mantenimiento de Almacenes de Datos en Línea".

Simposium Intertecnológico de Computación e Informática (SICI) en el 10º Congreso Internacional de Investigación en Ciencias Computacionales (CIICC2003), Oaxtepec, Morelos, 22-24 de Octubre del 2003.

Institute of Electrical and Electronics Engineers (IEEE), Academia Nacional de Ciencias Computacionales (ANaCC) y Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET). ISBN: 968-58-23-02-2.

- "Mantenimiento de Almacenes de Datos en Línea y Tiempo Real", 8º Workshop Iberoamericano de Ingeniería de Requisitos (IDEAS05), Valparaíso (Chile), del 2 al 6 de Mayo del 2005.

ISBN 956-7051-07-0.

- "Real time self-maintenable data warehouse"

Proceedings of the 44th Annual ACM Southeast Regional Conference, 2006, Melbourne, Florida, USA, March 10-12, 2006.

- "Real Time Self-Maintenable Update to Aggregate Information for Data Warehouse"

Proceeding of 15th International Conference on Software Engineering and Data Engineering (SEDE-2006), July 6-8, 2006, Los Angeles, (California), USA

ISBN 978-1-880-843-59-5.

8.3 Trabajos futuros.

- Generar de manera automática el código de los triggers (monitores/wrapper) necesarios para realizar la actualización del almacén de datos en tiempo real.
- Desarrollar una herramienta gráfica que nos permita explotar el almacén de datos.
- Desarrollar un método más robusto para la integración de datos, que sea capaz de integrar información no estructurada (imágenes, email, etc).

- Desarrollar un método de actualización de almacenes de datos que nos permita definir vistas materializadas que contengan funciones de agregación tales como desviación estándar, varianza, etc.
- Desarrollar una plataforma para el manejo de almacenes de datos orientados a objetos.
- Desarrollar un método que nos permita realizar el mantenimiento del almacén de datos en un ambiente de datos distribuido.

Bibliografía

[1] “OLAP Council White Paper”, http://www.olapcouncil.org/research/whtpapco.html, Marzo 2003.

[2] Jhon D. Porter and Jhon J. Rome, “The Data Warehouse: 2 year late, Lesson Learned”, Arizona State University, 1994.

[3] Chris Todman, “Designing a Data Warehouse”, John Wiley and Sons, Inc., New York, NY

[4] Kimball, R., ”The Data Warehouse Tollkit: Practical Techniques for building Dimensional Data Warehouses”, John Wiley and Sons, Inc., New York, NY. 1998

[5] Burleson Donald, “High Performance Oracle Data Warehousing”, Ed. Coriolis Group Books, USA 1997.

[6] S. Samtani, V. Kumar, M. Mohania, “Self Maintenance of Multiple Views in Data Warehousing”. Proceeding of ACM International Conference on Information and Knowlwdge Management, CIKM, November 1999.

[7] Rami Rifaieh, Nabila Aïcha Benharkat, “Query-based Data Warehousing Tool“, Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP, November 2002.

[8] I.Manolescu, D.Florescu and D.Kossmann. "Anwering XML Queries over Heterogeneous Data Sources", 27th VLDB, Roma Italia, 2001.

[9] Rami Rifaieh, Nabila Aïcha Benharkat, “Query-based Data Warehousing Tool“, Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP, November 2002.

[10] Joseph Albert, “Data Integration in the RODIN Multidatabase System”, Proceeding 1st IFCIS International Conference on Cooperative Information Systems, Brussels, Belgium, 1996.

[11] Maurizio Lenzerini,” Data Integration: a Theoretical Perspective”, Universita di Roma “La Sapienza”. ACM PODS 2002

[12] Oliver M. Duschka, “Query planning and optimization in integration information”, tesis Phd, 1997

[13] Farshad Hakimpour, Andreas Gepper, “Resolving Semantic Heterogeneity in Schema Integration: an Ontology: Based Approach”, in the preceeding of the International Conference on Formal Ontology in Information Systems, Maine, USA, 2001.

[14] Diego Calvanese, Guiseppe De Giacomo, Mauricio Lenzerini at el, “Source Integration in Data Warehousing”, In proceeding of Workshop on Data Warehouse Design and OLAP Technology, Wein, Australia 1998, IEEE Computer Society.

[15] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge. The Stanford data warehousing project. *IEEE Bull. on Data Engineering*, 18(2):41–48, 1995.

[16] R. Hull and G. Zhou. A framework for supporting data integration using the materialized and virtual approaches. In *Proc. of ACM SIGMOD*, pages 481–492, 1996.

[17] Maurizio Lenzerini,” Data Integration: a Theoretical Perspective”, Universita di Roma “La Sapienza”. ACM PODS 2002, USA

[18] Anca Vaduva, Klaus R. Dittrich, “Metadata Management for Data Warehousing: Between Vision and Reality”, 2001 International Database Engineering & amp; Aplications Symposium (IDEAS´01), Gronoble France.

[19] Jhon Poole, Dan Chan, Douglas Tolbert And David Mellor, "Common Warehouse Metamodel, An Introduction to the Standard for Data Warehouse Integration", 2002, pp 70.

[20] José A. Blakeley, P.-A Larson and F.W. Tompa. "Efficiently Updating Materialized Views", In procceding of the ACM SIGMOD International Conference on Management of Data, pp 61-71, Washington, D.C.,June 1986.

[21] Inderpal Singh Mumick, Dallan Quass y Barinderpal Singh Mumick, "Maintenance of Data Cubes and Summary Tables in Warehouse", SIGMOD, 1997.

[22] Dallan Quass , Jennifer Widom, "On-line warehouse view maintenance", Proceedings of the 1997 ACM SIGMOD international conference on Management of data, p.393-404, May 11-15, 1997, Tucson, Arizona, United States

[23] Wilburt Juan Labio, Jun Yang, at el, "Performance Issues in Incremental Warehouse Maintenance", Techincal Report, Stanford University, 1999.

[24] Henrik Engström Sharma Chakravarthy Brian Lings, "A Holistic Approach to the Evaluation of Data Warehouse Maintenance Policies". Technical reports 2000.

[25] Ashish Gupta, Inderpal Singh Mumick," Maintenance of Materialized Views: Problems, Techniques, and Applications". In IEEE Data Engineering Bulletin, Special Issue on Materialized View & Data Warehousing. pp. 3-18, June 1995.

[26] Levitin, Anany V., and Redman, Thomas C. Data as a resource: Properties, implications, and prescriptions, *Sloan Management Review*; Cambridge; Fall 1998, 40, 1, 89-101.

[27] J. Gray, A. Bosworth, A. Layman, and H.Pirahesh. Data cube. A relational aggregation operator generalizing group by, cross-tab, and sub-totals. In Procceding of Twelfh IEEE International Conference on Data Engineering,pages 152-159, 1996.

[28] Yue Zhuge, Hector García-Molina, Joachim Hammer, Jennifer Widom, "View Maintenance in a Warehousing Environment". In SIGMOD, pp 316-327, San Jose, CA, May 1995.

[29] Richard Hull and Gang Zhou. A framework for supporting data integration using the materialized and virtual approaches. In H. V. Jagadish and Inderpal Singh Mumick, editors, Proceedings of ACM SIGMOD 1996 International Conference on Management of Data, Montreal, Canada, June 1996.

[30] C. Batini, M. Lenzerini, and S. Navathe, A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys. 18(4):323-364. 1986.

[31] C. Collet, M. N. Huhns, and W.-M. Shen. Resource integration using a large knowledge base in Carnot. *EEE Computer*, 24(12):55–62, 1991.

[32] Y. Arens, C. A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. *J. of Intelligent Information Systems*, 6:99–130, 1996.

[33] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *J. of intelligent Information Systems*, 5:121–143, 1995.

[34] I. Manulescu, D. Florescu and D. Kossmann. Answering XML queries on heterogeneous data sources. In Proc. of 27th Int. Conf. on Very Large Data Bases (VLBD 2001), pages 241-250,2001.

[35] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Básalos and J. Widom. The TSIMMIS approach to mediation: Data models and language. J. of Intelligent Information Systems, 8(2):117-132,1997.

[36] C.H. Goh, S. Bressan, S. E. Madnick and M. D. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. ACM Trans. ON Information Systems, 17(3):270-293,1999.

[37] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. Luniewsky, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams and E. L. Wimmers. Towards heterogeneous multimedia information systems: The Garlic approach, In Proc. of the 5th Int. Workshop on Research Issues in Data Engineering – Distribuided Object Management (RIDE-DOM´95), pages 124-131. IEEE Computer Society, 1995.

[38] Christop Quix, Repository Support for Data Warehouse Evolution, Proceedings of the Intl. Workshop DMDW'99, Heidelberg, Germany, June 14-15, 1999.

[39] Maurizio Rafanelli, Libro "Multidimensional Databases: Problem and Solutions", Chapter VIII, Idea Group Publishing,2003, ISBN 1-59140-053-8.

[40] Mauricio Rafanelli, Libro "Multidimensional Databases: Problem and Solutions", páginas 224-225, Idea Group Publishing, 2003, ISBN 1-59140-053-8.

[41] Shirley Becker, Libro "Data Warehousing and Web Engineering", página 20, idea Group publishing, 2003, ISBN 1931777020

[42] W.H.Inmon, Libro “Building tha Data Warehouse, thirt edition”, página 2, Wiley computer publisher, 2002, ISBN 0471081302

[43] An Oracle with paper, “DataBase Architecture: Federated vs. Clustered”, march 2002.

[44] Matthias Jarke, Mauricio Lenzerini, Yannis Vassiliou, Panos Vassiliais, “Fundamentals of Data Warehouses”, Springer, 1999

[45] Yue Chuge, Hector García-Molina, Janeth L. Wiener, “The Strobe Algorithms for Multi-Source Warehouse Consistency”, In proceeding of the Fourth International Conference on Parallel and Distributed Information Systems, IEEE Computer Society, December 1996.

[46] Jennifer Sampson, Clare Atkins, Semantic Integrity in Data Warehousing: A framework for understanding, 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 8 January 07 - 10, 2002, Big Island, Hawaii

[47] Cood, E. F., Cood,S.B.,Salley, C.T., Providing OLAP (On-Line Analytical Processing) to user-analyst: An IT mandate.Technical report, 1993.

[48] Dallan Quass, “Materialized Views in Data Warehouses”, PhD Thesis, Standfor University, 1997.

[49] http://alarcos.inf-cr.uclm.es/doc/bbddavanzadas/dwh.pdf, Octubre 2004.

[50] http://bbdd.escet.urjc.es/documentos/datawarehouse_upv.pdf, Octubre 2004.

[51] Alexandros Lambrinidis; Nick Roussopoulos. “A performance Evaluation of Online WareHouse Update Algorithms”. CSHCN (Center for Satellite and Hybrid Communication Networks),1998

[52] Songting Chen, Bin Liu and Elike A. Rundensteiner, "Multiversion-based wiew maintenance over distributed data source", ACM Tran. Database System", Vol. 29, No. 4, December 2004.

[53] Stefano Rissi, Open problems in Data Warehousing: 8 years later, DOLAP, Nov.7 2003.

[54] Michael Boyd, Sasivimol Kittivoravitkul, Charalambos Lazanitis, Peter M^{c}Briend and Nikos Rizopoulos. AutoMed: A BAV Data Integration System for Heterogeneous Data Sources. In Proceeding of CAISE2004, springer-Verlang LNCS, 2004.

[55] Koenig, Shaye, and Robert Parge, "A transformational Framework for the Au-tomatic Control of Derived Data," Proc of the 7th Internatronal Conference on Very Large Data Bases, 1981, Pages 306-318

[56] Shmueh, Oded, and Alon Ital, "Maintenance of views", SIGMOD 84 Proceeding of Annual Meeting (Boston, MA), Sigmod Record, Vol 14, No. 2, 1984, pages 240-255.

[57] Hammer, Michael, and Sunil K Sarm,"Efficient Monitoring of Database Assertions, Supplement Proc ACM SIGMOD Internatronal Conference on anagement of Data, Austin, TX, May 31-June 2, 1978, Page 38

[58] Buneman, 0 Peter, and Eric K Clemons, "Efficlently Monitoring Relational Data-bases," ACM transactions on Database Systems, Vol 4, No 3, September 1979, Pages 368-382

[59] Josep Silva, Jorge Belenguer, Matilde Celma, "Materialización de Vistas Multi-Origen: Vistas Multinivel", Actas de las VII Jornadas de Ingeniería del Software y Bases de Datos. 2002.

[60] G. Zhou, R. Hull, R. Kiing, J.C. Franchitti, “Data Integration and Warehousing Using H2O”, IEEE Data Enginnering Bulletin 18(2), 1995.

[61] Stefano Rizzi, II-Yeol Song, Report on the ACM Sixt International Workshop on Data Warehousing and OLAP (DOLAPR 2003), ACM SIGIR Forum,Vol.38, No.1 Junio 2004

[62] Themistoklis Palpanas, Richard Siidle, Roberta Cochrane and Hamid Pirahesh, Incremental Maintenance for Non-Distributive Agregate Function, Procceding of 28th VLDB Coonference, Hong Kong, China, 2002.

[63] Erhard Rahm ,Hong Hai Do, Data Cleaning: Problems and Current Approaches. IEEE Bulletin of the Technical Committee on Data Engineering, Vol 23 No. 4, December 2000

[64] Songting Chen, Xin Zhang and Elike A. Rundensteiner, A Compesation-Based Approach for View Maintenance in Distributed Environments”, IEEE Transactions on Knowledge and Data Engeneering, Vol 18, No. 8, August 2006

[65] Patric Ziegler and Klaus R. Pittrich (2004),”Three Decades of Data Integration –All problem solved?”, WCC2004, 3-12.

[66] http://en.wkipedia.org/wiki/Data_integration, Feberero 2005

[67] Bonnie O´Neil, Data Archeology a Day in the Life an Integration Consultant (http://www.b-eye-network.com/newsletters/inmon/3010), Junio 2006.

[68] Jeennifer Widom,”Research Problems in Data Warehouse”, Proceedings of 4th International Conference on Information and Knowledge Management, pages 25-30, 1995.

[69] P. Vassiliadis. “Gulliver in the land of data warehousing: practical experiences and observations of a researcher”. In Proc. of

Design and Management of Data Warehouses (DMDW'2000) 2nd International Workshop in conjunction with the 12th Conference on Advanced Information Systems Engineering CAiSE*00, June 5-6, 2000, Stockholm, Sweden.

[70] Laura Hass, Donald Kossmann, Edward Wimmers and Jun Yang. "Optimizing queries across diverse data source". In proc. Of the VLDB conf., Athens Greece, 1997.

[71] S. Cluet, C. Delobel, J. Simeon and K. Smaga. "Your mediators need data conversion". In proc. Of ACM SIGMOD conf. on Management of Data, Seattle, WA, 1998.

[72] T. Kirk, A. Levy, Y. Sagiv and D. Srivastava. "The Information Manifold". In AAAI Spring Sysposium on Information Gathering, 1995.

[73] A. Tomasic, L. Raschid and P. Valduriez. "Scaling access to distributed heterogeneous data source with Disco". IEEE Transactions on Knowledge and Data Enggenering, 1998.

[74] D. Vesset. "Worldwide Data Warehousing Tools 2004-2008 Forecast and 2003 Vendor shares", Technical Reporte DOC #31616, IDC, July 2004.

[75] M. Jarke, M. Lenzerini, Y. Vassiliou and P. Vassiliadis, "Fundamentals of data warehouse", Springer-Verlang, 2 Edición, 2003.

[76] Wayne Eckerson, "Four Ways to Build a Data Warehose", Business Intelligence Network, May 2007.

[77] Ki Young Li and Myoung Ho Kim, "Optimizing the Incremental Maintenance of Multiple Join Viwes", DOLAP'05, November 4-5 2005, Bremen, Germany.

[78] Dan Vesset and Brian McDonough. Worldwide Business Intelligence Tools 2006 Vendor Shares. Technical Report Doc # 207422, IDC, 6 2007.

[79] Sergio Lujan-Mora, Juan Trujillo, “Un método global basado en UML para el diseño de almacenes de datos”, JISBD, 2003

Lista de Figuras

Lista de Tablas

Lista de abreviatura

BAV	Both As View
DASD	Direct Access Strorage Device
DM	Data Mining
DSS	Decisión Support Systems
DW	Data Warehouse
ETL	Extract, Transform and Load
GAV	Global as View
LAN	Local Area Network
LAV	Local as View
MOLAP	Multidimensional On Line Analytical Processing
OLAP	On Line Analytical Processing
OLTP	On Line Transactional Processing
ROLAP	Relational On Line Analytical Processing
SGBD	Sistemas de gestión de bases de datos
UAS	Universidad Autónoma de Sinaloa
VM	Vista Materializada
VPN	Virtual Private Netware
WEB	Abreviatura para World Wide Web