Mantenimiento de Vistas en un Ambiente de Almacenes de Datos

Francisco Javier Cartujano Escobar

Departamento de Computación
Tecnológico de Monterrey - Campus cd. de México
México D. F., México
E-mail: fcartuja@itesm.mx

Abstract. El problema de integración de datos para mantener vistas materializadas es un problema complejo y de suma importancia para ambientes de almacenes de datos. El presente artículo describe la problemática inherente en el proceso de integración de datos en un almacén de datos; se hace referencia a los trabajos realizados en dicha área y se define un proyecto donde se establece la arquitectura e implementación de una aplicación de almacenes de datos para el mantenimiento de vistas materializadas.

1 Introducción

Un almacén de datos es una gran colección de datos que tienen la característica de ser orientados a entidades, integrados, no volátiles y con un tratamiento especial de tiempo, cuya finalidad es el de apoyar el proceso de toma de decisiones en los negocios [Himmon 96]. Orientado a entidades significa que se modelan entidades de negocio y no a las aplicaciones; integrada significa que se recolecta la información de distintas fuentes de datos y se resuelven los problemas de inconsistencia que pudieran presentarse entre estas; no volátiles significa que en general la información del almacén de datos no sufre modificaciones; con un tratamiento especial en el tiempo significa que (a) el almacén de datos almacena información histórica, (b) el tiempo de vida de los datos en el almacén es de años y (c) la llave de las entidades en el almacén involucra algún elemento del tiempo.
De todos los aspectos del almacén de datos, la integración de datos es quizás el aspecto más importante y el que representa el reto más fuerte al construir el almacén. La razón de esta aseveración es que al integrar la información se tiene que acceder a distintas fuentes operacionales, las cuales se encuentran distribuidas, son autónomas entre sí y en la mayoría de los casos existe heterogeneidad entre ellas.

David Juárez Romero, J. Crispín Zavala-Díaz,(eds.). AGECOMP2004, Memorias del 3er Congreso de Cómputo AGECOMP, UAEM, México, ISBN(e) 968-878-205-X, pp. 15 - 24

Por una parte, el acceso a datos distribuidos esta sujeto a la disponibilidad por parte de los sitios operacionales donde ellos residen, además es necesaria una administración eficiente del tráfico de información en la red. Por su parte, la autonomía de las fuentes operacionales puede originar inconsistencias en el almacén de datos en el proceso de actualización de los datos, es decir, si el proceso de actualización de datos no contempla métodos que contemplen la actualización concurrente del almacén de datos por parte de las fuentes operacionales (muchas fuentes operacionales pudieran estar descargando datos en el mismo periodo de tiempo) es posible que hayan inconsistencias en el almacén. En lo que respecta a la heterogeneidad de la información (quizá el problema más fuerte y no exclusivo de esta tecnología, también lo presentan los llamados sistemas multibase de datos o federados) se refiere a los modelos de datos, lógicos y/o físicos en los que los distintos sitios operacionales pudieran tener discrepancias o conflictos entre ellos, por ejemplo: conflicto de nombres (sinónimos y homónimos), conflicto en los tipos de datos, conflicto en la escala de datos, conflicto en la precisión de datos, conflicto en las restricciones de integridad, conflictos en la compatibilidad a la unión, conflicto de isomorfismos de esquemas, conflicto de datos inconsistentes, y discrepancia esquemática, entre otros. Como vemos la integración de datos representa un problema muy fuerte e interesante en el ambiente de almacenes de datos.

La figura 1. representa la arquitectura básica de un almacén de datos. En cada fuente, un módulo llamado *monitor*, detecta las actualizaciones de interés realizadas en las fuentes operacionales y manda dichas actualizaciones al almacén. En el almacén, el integrador recibe los datos fuentes, ejecuta cualquier integración o transformación de datos necesaria, añade cualquier información extra deseada, tal como una estampa de tiempo para análisis histórico y almacene los datos en el almacén, es decir se crea una vista materializada. Una vez realizado esto, los datos están disponibles para que los usuarios puedan realizar procesamiento analítico para apoyar el proceso de toma de decisiones.

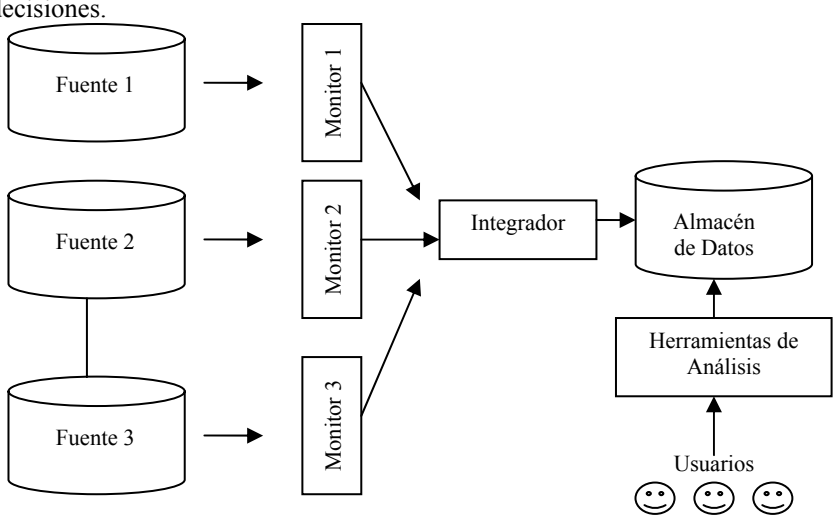

Fig. 1. Arquitectura básica de un almacén de datos.

El resto de este artículo está organizado de la siguiente manera. En la sección 2 se definen algunos términos relacionados al proceso de integración de datos. En la sección 3 se hace mención a algunos trabajos relacionados al mantenimiento de vistas en un ambiente de almacenes de datos. En la sección 4 se describe la propuesta de trabajo a realizar, indicando el objetivo, los requerimientos y su alcance, así como un análisis de su implementación. Por último, en la sección 5 se presentan una serie de comentarios finales.

2 Terminología

¿Qué es una vista? Una vista es una relación derivada definida en términos de relaciones bases. Una relación define una función que mapea un conjunto de tablas base a una tabla derivada. Esta función, en ambiente tradicionales, recalcula la vista cada vez que es referenciada [GM95].
¿Qué es una vista materializada? Una vista puede ser materializada almacenando los tuplos de la vista en alguna parte del sistema. Consecuente mente el acceso a vistas materializadas puede ser mucho más rápido que recalcular la vista [GM95].
¿Qué es el mantenimiento de una vista? Es el proceso de actualizar una vista materializada en respuesta a actualizaciones que se originan en las tablas que soportan la vista [GM95].

¿Qué es el mantenimiento incremental de una vista? En la mayoría de los casos es costoso mantener una vista volviendo a recalcularla (es decir, como si se creara por primera vez). Frecuentemente es calcular solo los cambios para materializarlos. Algoritmos que calculan cambios a una vista en respuesta a cambios en las relaciones bases son llamados algoritmos de mantenimiento incremental de vistas [GM95].

¿Qué es una vista SPJ? Es aquella vista que se forma mediante una expresión que involucra operaciones de join, selección y proyección, es decir, $V= \pi_{\text{atributos}} (\sigma_{\text{condición}} (r_1 \blacktriangleright\blacktriangleleft r_2 \blacktriangleright\blacktriangleleft \ldots \blacktriangleright\blacktriangleleft r_n))$ [ZGMW96].

¿Qué es una vista automantenible? Cuando una vista junto con un conjunto de vistas auxiliares pueden ser mantenidas sin acceder a las relaciones base, decimos que las vistas son automantenibles [QGMW96].

¿Qué es un monitor? Es un programa que detecta las actualizaciones de interés realizadas en las fuentes operacionales y manda dichas actualizaciones al almacén de datos. [ZGW96].

¿Qué es un integrador de datos? Es un programa que reside en el almacén de datos. Recibe los datos fuentes, ejecuta cualquier integración o transformación de datos necesaria y añade dichos datos al almacén[ZGW96].

3 Trabajos Realizados

En esta sección se describe brevemente algunas de las investigaciones relacionadas con el proceso de integración de esquemas y en forma más específica lo relacionado en un mantenimiento de vistas en un almacén de datos.

En lo que concierne al proceso de detectar las actualizaciones realizadas en las fuentes operacionales (monitoreo), se han propuesto varios esquemas. En [DEVLIN97] se mencionan seis técnicas para llevar acabo esta actividad (él les llama seis técnicas de captura de datos): captura estática, captura asistida por la aplicación, captura asistida por triggers, captura apoyada por la bitácora, captura basada en estampas de tiempo y comparación de archivos, también llamada diferencial de instantáneas (snapshot differential). Cada una de las técnicas es apropiada bajo cierto tipo de condiciones, por ejemplo, para fuentes operacionales donde no existe un DBMS (conocidos como sistemas "legacy") y el volumen de información que se maneja es pequeño quizás una captura por comparación de archivos sería adecuada. Si por el contrario, en las fuentes operacionales existe un DBMS, la captura asistida por triggers sería lo más fácil de implementar. En particular en [WGM96] se plantea dos algoritmos para realizar de manera óptima un diferencial de instantáneas, estos algoritmos están basados en join externos (outer joins) y técnicas de compresión de archivos.
Referente a la creación de un modelo conceptual para la integración de esquemas, en [CGLNR98] se presenta un modelo conceptual formal que permite modelar los conflictos entre las diferentes fuentes operacionales. En [SR96] se presentan una serie de estrategias para la administración de un metadato que integra diferentes fuentes operacionales. Algunas de las técnicas planteadas en [OV90] para sistemas multibase de datos pueden también ser aplicadas para la creación de un modelo integrado y unificado de datos en un almacén de datos.

En lo que respecta a un modelo de vistas materializadas, básicamente se presentan dos alcances: vistas automantenibles y vistas que necesitan referenciar a los datos fuentes. Referente a las vistas automantenibles, en [QGMW96] se describe un algoritmo que dada una vista permite derivar un conjunto de vistas auxiliares, tal que la vista original y las vistas auxiliares generadas sean automantenibles. El trabajo muestra que conociendo restricciones de llave y de integridad referencial de las tablas fuentes, es posible generar el conjunto de vistas auxiliares. En [HUYN97] se resuelve el mismo problema de vistas automantenibles, pero para múltiples vistas en el dw. Respecto a las vistas que necesitan diferenciar a datos fuentes existen distintos algoritmos, algunos de ellos con problemas de consistencia al momento de integrar los datos en el almacén de dtaos. En [ZGMW96] se presenta una familia de algoritmos, llamada algoritmos Strobe, los cuales permiten mantener la consistencia de datos en el almacén, aún bajo escenarios que contemplan transacciones de actualización concurrentes

por parte de las fuentes operacionales. En [AASY97] se presenta un algoritmo alterno, llamada SWEEP, que también permite la consistencia de los datos con transacciones de actualización concurrentes.

4 Trabajo a desarrollar

En la sección 1 de este artículo se dio una descripción general de la problemática que se presenta al integrar datos en un dw. En la sección 3 se hizo mención de una serie de trabajos que solucionan algunos de los problemas presentados en la sección 1. En esta sección establecemos el trabajo a desarrollar.

4.1 Objetivo

El proyecto planteado en este artículo consiste en implementar una arquitectura para la integración de datos en una aplicación de almacenes de datos con vistas materializadas. El objetivo final es implementar algunos de los algoritmos establecidos en la sección 3 y analizar la problemática inherente al establecer una aplicación de dw.

4.2 Requerimientos

a) Establecer una aplicación de almacén de datos.
b) Establecer la arquitectura para el almacén considerando la integración de datos.
c) Implementar algoritmos (existentes o nuevos) que permitan realizar la integración de datos.
d) Analizar, resolver y documentar la problemática que surja al momento de implementar la arquitectura del almacén.
e) Realizar un análisis de los algoritmos implementados con la finalidad de ver si son mejorables.

4.3 Alcance del trabajo

a) La aplicación a desarrollar consiste de 3 fuentes de datos operacionales. El esquema global de la base de datos distribuida contempla de tres tablas.
b) El almacén de datos consta cinco tablas materializadas.
c) No se contempla el manejo de heterogeneidad en las fuentes operacionales.
d) Se contempla implementar el monitor y el integrador de la arquitectura del almacén de datos

4.4 Definición de la aplicación

Para el proyecto se define una aplicación de ventas. El modelo global de la base de datos distribuida es el mostrado en la figura 2.

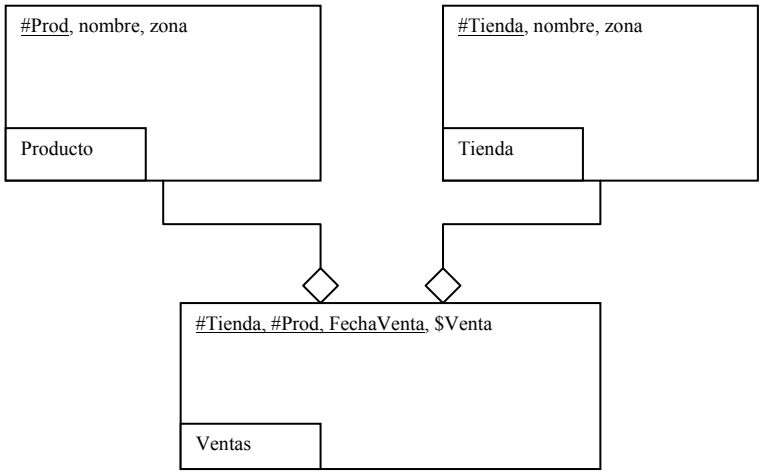

Fig. 2. Esquema global de la base de datos de la aplicación a Ventas

Para la implementación de esta aplicación se considera únicamente dos sitios (dos fuentes operacionales). Un sitio representará a la zona geográfica norte (N) y el otro en la zona geográfica sur (S). Alrededor de cada sitio existen varias tiendas. El control de las tiendas que están en la zona N son controlados en sitio N y las tiendas que están en la zona sur son controladas en el sitio S. En cada sitio se encuentra el mismo modelo de datos, pero no así la misma información, es decir, las 3 tablas fueron fragmentadas horizontalmente de acuerdo al atributo zona (en ventas realizó una fragmentación horizontal derivada), de tal forma que la información que se encuentre en la base de datos del sitio N se refiere a las tiendas de la zona N, y la información que se encuentra en la base de datos del sitio S se refiere a tiendas en la zona S.

Definición del almacén de datos.

Las vistas que serán materializadas en el almacén son las siguientes (especificadas en SQL):

a. **Producto =** Select* From ProductoN
Unión
Select* From ProductoS

b. **Tienda** = Select*From TiendaN
Unión
Select*From TiendaS

c. **Ventas** = Select*From VentasN
Unión
Select*From VentasS

d. **VentasTienda** = Select #tienda, SUM($venta)
From Ventas
Group By #tienda

e. **VentasZona** = Select zona, SUM($venta)
From Ventas, Tienda
Group By zona

Referente a las tres primeras vistas estas representan la unión de los datos de los dos sitios. Las otras dos vistas representan información agregada con el fin de proporcionar rapidez al hacer una consulta sobre las ventas por tienda o por zona. Por otro lado, se enfatiza que las vistas serán materializadas, por lo que estas no serán creadas con el comando create view el cual involucra un query, en realidad estás serán creadas con create table y serán mantenidas al haber cambios en las fuentes operacionales.

4.6 Arquitectura de la integración de datos al dw.

En la figura No. 3, se propone una arquitectura para la integración de datos en nuestra aplicación. Los números encerrados en el círculo, representan la secuenciación de las actividades que se involucran en el proceso de integración para el mantenimiento de las vistas materializadas en el almacén de datos. A continuación se explican estos pasos.

1. Los usuarios de los sistemas operacionales realizan actualizaciones a las bases de datos.
2. Dichas actualizaciones son detectadas por el monitor a través de un conjunto de trigger's.
3. El monitor tiene la función de registrar todas las actualizaciones en un archivo llamado delta. De esta manera, todos los cambios que sufren las bases de datos operaciones en cada uno de los sitios quedan registrados en su respectivo archivo delta. Se aclara que cada fuente operacional tiene su propio monitor (por cuestiones de espacio solo se indicó uno).

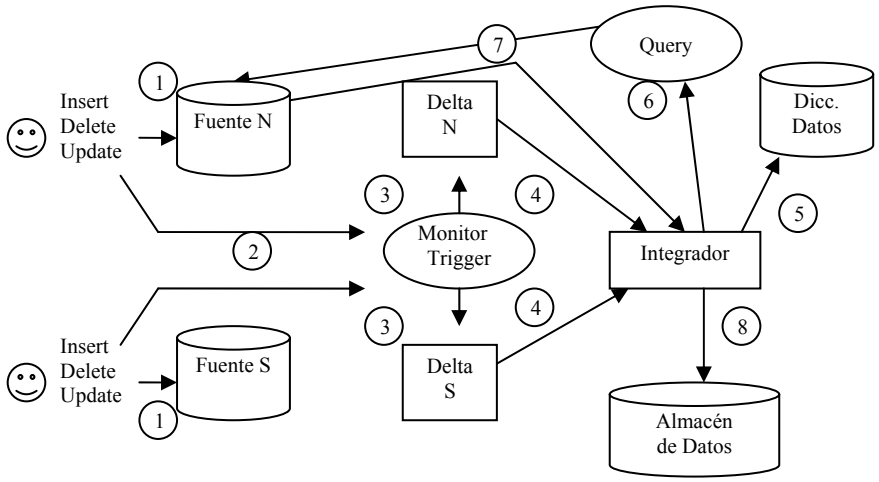

Fig. 3. Arquitectura planteada para la integración de datos.

4. Periódicamente, quizás al final del día o de acuerdo a otras políticas, se aplicarán todas las actualizaciones que se encuentran en cada uno de los archivos delta. Esta actualización se hará de manera sincronizada, es decir, primeramente un archivo delta y luego otro. El conjunto de actualizaciones registradas en el archivo delta las procesará el integrador como una sola transacción de tipo batch.
5. El integrador consultará el diccionario de datos para verificar qué vista está involucrada para una actualización en particular.
6. Es posible que algunas actualizaciones originadas en las fuentes operacionales requieran de información adicional para actualizar la vista materializada; por ejemplo, cuando se tiene una vista que involucra un join, cualquier inserción o borrado a cualquiera de las tablas involucradas en el join pudiera ocasionar que la vista se actualizara, como estas se encuentran únicamente actualizadas en los sitios operacionales es necesario ir a dichos sitios para poder realizar el join. La forma de cómo llevar a cabo este proceso es generando una consulta.
7. La consulta generada por el integrador es ejecutada en los sitios operacionales y de vuelta al integrador para ir formando una tabla con los registros que deberán ser actualizados en la vista.
8. Una vez calculados los registros del paso 7, estos son aplicados a las vistas materializadas del dw. Como se nota, las vistas se actualizaron únicamente con los cambios aún no registrados en ellas, es decir se hizo un mantenimiento incremental.

4.7 El Monitor

Para la implementación del monitor se utilizarán trigger's, los cuales serán disparados cada vez que se realice una actualización a la base de datos. Estos trigger's tendrán la responsabilidad de guardar los cambios de interés que han sufrido las bases de datos operacionales. Es posible que el trigger tenga la tarea adicional de acceder información del registro, particularmente esto es necesario, si los updates son manejados en el archivo delta como un borrado seguido de una inserción, por ejemplo, si se modifica únicamente un atributo de un registro, el update quedaría registrado de la siguiente manera "UPDATE <tabla> SET <atributo> = <valor>", como esto origina un borrado luego de una inserción, será necesario recuperar la información adicional del registro del archivo delta, indicando la fecha y hora en que se registró dicho evento.

4.8 El Integrador

Para la implementación del módulo que procede a la actualización de la vista materializada, se utilizará alguno de los algoritmos planteados en [ZGMW96] o en [AASY97]. De manera particular se ha pensado en el algoritmo TSTROB de [ZGMW96], inclusive bajo ciertas características particulares en el proceso de integración es posible que pueda ser modificado para eficientar el proceso. También es conveniente aclarar que ninguno de los algoritmos mencionado en las dos referencias, manejan la actualización de las vistas con información agregada bajo las condiciones que ellos plantean. Debido a que en este trabajo se tienen dos de tales vistas, se analizará la manera de cómo llevar a acabo esta actividad bajo la arquitectura planteada en el punto 5.2.

5 Comentarios finales

El problema de integración de datos para mantener vistas materializadas es un problema complejo y de suma importancia para ambientes de almacenes de datos.

El proyecto propuesto implementa una aplicación de almacén de datos bajo una arquitectura que contempla el mantenimiento de vistas materializadas. Se propone utilizar algunos algoritmos ya conocidos, pero en la medida que estos puedan ser mejorados, se procederá a la modificación. En el caso de las vistas con agregados se analizará la manera más conveniente de llevar acabo esta actualización.

Los resultados y el análisis de los mismos se documentarán en una continuación del presente artículo.

Referencias

[**AASY97**] D. Agrawal, El Abbadi, A. Singh and T. Yurek. Efficient View Maintenance at Data Warehouses. University of California. 1997.

[**CGLNR98**] D. Calvanese, G. De Giancomo, M. Lenzerini, D. Nardo and Ricardo Rosati. Source Integration in Datawarehousing. Universita di Roma "La Sapienza". 1998

[**DEVLIN97**] b. Devlin. Data Warehouse: from arquitecture to implementation. Addison Wesley. 1997.

[**GM95**] A. Gupta and I. Munik. Maintenance of materialized views incrementally. In SIGMOD, Washington D.C., May 1993.

[**HUYN97**] N. Huyn. Multiple View Self-Maintenance in Data Warehousing. Stanford University.1997.

[**INMON92**] W.H. Inmon. Building the Data Warehouse. QED Information Sciences. Wellesley, 1992.

[**OV98**] M.T. Ozsu and P. Valduriez. Principles of Distributed Database Systems. Prentice Hall. 1991.

[**QGMV96**] D. Quass, A. Gupta, I. Munik and J. Widom. Making Views Self-Maintenance for Data Warehousing. In Proc. 4th Int. Conf. on Very Large Data Bases. Miami Beach, F1. Dec 1996.

[**SR96**] L. Seligman and A.Rosenthal. A Metadata Resources to Promote Data Integration. The MITRE Corporation. 1996.

[**WGM96**] W. J. Labio and H. García-Molina. Efficient Snapshot Differential Algorithms for Data Warehousing. Stanford University. 1996.

[**ZGMW96**] Y. Zhuge, H. García-Molina and J.L. Wiener. Consistency Algorithms for Multiple-Source Warehouse View Maintenance. Stanford University 1996.

[**ZGW96**] Y. Zhuge, H. García-Molina and J.L. Wiener. The Strobe Algorithms for Multiple-Source Warehouse Consistency. Stanford University 1996.