Visualización tridimensional de consultas *OLAP* de un cubo de datos utilizando la paginación

E. Bojórquez-Martínez[1], M.T. Serna-Encinas[2], C.E. Rose-Gómez[2], S.R. Meneses-Mendoza[2], O.M. Rodriguez Elías[2]

[1]Universidad de Sonora, Departamento de Ingeniería Industrial,
Blvd. Luis Encinas y Rosales S/N, Col. Centro, Hermosillo, Sonora, México.
[2]Instituto Tecnológico de Hermosillo, División de Estudios de Posgrado e Investigación, Av. Tecnológico S/N CP 83170, Hermosillo, Sonora, México.
eddiebomtz@gmail.com, {tserna, crose, so_meneses, omrodriguez}@ith.mx
(Paper received on June 30, 2013, accepted on August 15, 2013)

Abstract. Actualmente, las herramientas con arquitectura *ROLAP* (Procesamiento Analítico en Línea Relacional) permiten la visualización de datos multidimensionales en forma de reporte bidimensional, este proceso dificulta el análisis de la información ya que se requiere la anidación de las dimensiones en una sola tabla. El trabajo de investigación se enfoca en el desarrollo de un sistema que permite visualizar los datos en forma tridimensional, para facilitar el análisis de los mismos. En este artículo, se presenta tanto el diseño como la implementación del sistema de visualización tridimensional de un almacén de datos; el diseño explica la arquitectura *HOLAP* (*Hybrid OLAP*), misma que considera la generación de consultas *OLAP* y la creación, manipulación y visualización del cubo de datos. La implementación del sistema muestra cómo se convierte un reporte bidimensional almacenado en una vista materializada, a un cubo tridimensional, además de describir el proceso de paginación del cubo de datos creado.

Keywords: Almacén de datos, servidor *HOLAP*, vistas materializadas, visualización multidimensional, paginación.

1 Introducción

Los almacenes de datos están orientados al análisis de grandes volúmenes de información y permiten realizar consultas comparativas y de proyección, esto facilita la toma de decisiones en la organización, permitiendo así lograr una ventaja competitiva [1-2]. Según Bill Inmon, conocido como el padre de los almacenes de datos, un almacén de datos "es un conjunto de datos integrados, históricos, variantes en el tiempo y unidos alrededor de un tema específico, que es usado por la gerencia para la toma de decisiones" [3].

Las herramientas actuales permiten mostrar los datos multidimensionales en forma bidimensional, la principal desventaja radica cuando se tienen varias dimensiones, puesto que consumen un mayor espacio en pantalla, dificultando la comparación delos datos multidimensionales [4-5].

Special Issue in Information Technology
Research in Computing Science 64, 2013, pp. 87-94

El presente trabajo describe el diseño e implementación del sistema de visualización tridimensional para la arquitectura *HOLAP*. La sección 2 describe el desarrollo del sistema y la arquitectura que se implementó, la implementación de la generación de la consulta OLAP y su visualización tridimensional, así como el componente para la paginación. El análisis de resultados es presentado en la sección 3; y finalmente, las conclusiones generadas del trabajo realizado son abordadas en la sección 4.

2 Desarrollo

El proyecto de investigación tiene como objetivo el desarrollar una herramienta que permite la visualización tridimensional de un cubo de datos, con la cual se podrá analizar la información de una manera más organizada y fácil de comparar, el proyecto completo considera la creación de una interfaz que permite la selección de dimensiones, tabla de hechos y medidas de un almacén de datos, a partir de dicha selección se genera un cubo de datos y se almacena en una vista materializada, una vez realizado dicho proceso, se visualiza el cubo de datos de manera tridimensional y también permite aplicar los operadores *Roll-up*, *Drill-down* y *rotate*.

La metodología que se siguió para el desarrollo del sistema es la del análisis y diseño orientado a objetos y se utilizó el lenguaje de modelado *UML*, esto permitió identificar los componentes del sistema implementado, y se diseñó una arquitectura con servidor de tipo *HOLAP*. El sistema se desarrolló con el lenguaje de programación *Java* 1.6 y el entorno de desarrollo integrado *Netbeans*. Para el almacén se utilizó el manejador de base de datos *Oracle* 10g. Para dibujar el cubo se utilizó la librería *Lightweight Java Game Library (LWJGL)*, que es una *API* escrita en *JAVA* para programación de juegos y aplicaciones 3D [6].

2.1 Arquitectura del sistema

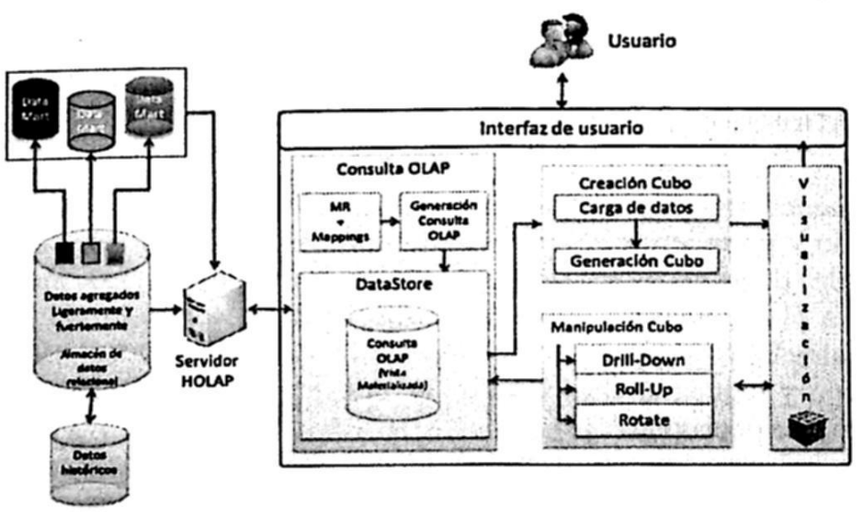

Fig. 1. Arquitectura del sistema propuesto (aportación propia).

En la figura 1 se muestra la arquitectura implementada misma que tiene 2 grandes componentes: Las fuentes de datos y la Interfaz de usuario; las primeras contienen tanto datos internos como externos de la empresa que están almacenados en un servidor *HOLAP*; la segunda está subdividida en 4 elementos principales: Consulta *OLAP*, Creación del cubo, Manipulación del cubo y Visualización del cubo.

El componente para la consulta *OLAP* permite que el usuario seleccione los datos que desea visualizar en el cubo de datos, se genera una consulta *OLAP* y se procede a materializarla en el *DataStore*; la materialización permite almacenar tanto la estructura de las tablas como el conjunto de datos resultado de la consulta, mejorando así el tiempo de respuesta, ya que el acceso frecuente a tablas básicas resulta demasiado costoso [7].

El siguiente proceso es la creación del cubo, este componente permite que el usuario seleccione la vista materializada y el sistema genera el cubo. Una vez que se ha generado el cubo de datos, el componente para la visualización permite la visualización bidimensional o tridimensional. Por último la manipulación del cubo permite que el usuario pueda utilizar los operadores *Drill-down*, *Roll-up* y *rotate*.

2.2 Conversión de consultas *OLAP* para la visualización tridimensional

Para cada cubo de datos, es necesario que se generen tres consultas distintas para convertir el reporte bidimensional, de cada una de las caras visibles del cubo, en forma de matriz y poder ser mostrado de manera tridimensional; sólo se obtendrán los primeros 9 elementos en X, Y, y 4 elementos en Z; además deben coincidirlos datos con los títulos. En todas las consultas se utilizó la pseudo-columna *ROWNUM*, misma que permite obtener los primeros 9 registros para la cara de frente, sólo los 4 registros en el caso de la cara de arriba y los 4 primeros registros para la cara del lado derecho; de esta forma es posible implementar la paginación de las caras del cubo de datos, debido a que puede excederse el límite de registros que pueden ser visualizados en la pantalla.

El ejemplo que se explica a continuación es un cubo de datos con la tabla de hechos `ed_sales` y la medida `quantity_sold`, las dimensiones son: `ed_times`, `ed_customers` y `ed_products`; para el primero se tiene el campo `calendar_year` en el eje de las X, el segundo `cust_state_province` en el eje de las Z y el último `prod_category` en el eje de las Y.

Cara de frente: Primeramente se añade una sub consulta y para ello se obtuvieron los años del campo `calendar_year`, ya que este campo se encuentra en el eje de las X y se mostrará la cantidad vendida por año. El primer año que se comparó fue 1998, si coincide, entonces se muestra la cantidad vendida, en caso contrario se agrega un 0; el proceso se repite para los años 1999, 2000, 2001 y *null* (este último representa el total vendido para cada categoría).

En la cara de frente, el operador *CUBE* muestra tanto los subtotales como los totales [8]; esto fue generado al momento de crear la vista materializada; además, se añaden valores *null* al primer campo, que en el ejemplo es `cust_state_province`, con esto se identifica que es el total y por ello, se añadió la condición `cust_state_province` *IS NULL*, en la cláusula *WHERE*.

Una de las problemáticas que se tenía era que si la medida seleccionada contenía números grandes se empalmaban los datos, la solución que se le dio fue si los datos eran millones, se dividía entre un millón y se añadía una M, en caso de ser miles, se dividía entre mil y se añadía una K. Otra de las problemáticas era que si los datos en el eje de las X son de tipo cadena, al igual que en el caso anterior, se empalmaban la solución que se le dio fue rotar cada nombre 45 grados y así evitar que suceda esto.

A continuación se presenta la consulta generada para la cara de frente del cubo de datos:

```
SELECT * FROM (SELECT ROWNUM R, C0, C1, C2, C3, C4 FROM (SELECT
max(case when CALENDAR_YEAR='1998' then QUANTITY_SOLD else 0 end)
C0,max(case when CALENDAR_YEAR='1999' then QUANTITY_SOLD else 0 end)
C1,max(case when CALENDAR_YEAR='2000' then QUANTITY_SOLD else 0 end)
C2,max(case when CALENDAR_YEAR='2001' then QUANTITY_SOLD else 0 end)
C3,max(case when CALENDAR_YEAR IS NULL then QUANTITY_SOLD end) C4
FROM CUBOESTADOCATANIOMV WHERE CUST_STATE_PROVINCE IS NULL GROUP BY
CUST_STATE_PROVINCE, PROD_CATEGORY ORDER BY CUST_STATE_PROVINCE,
PROD_CATEGORY))WHERE r between 1 and 6
```

2.3 Paginación

La paginación permite mostrar cierta parte del total de registros que se tienen en una consulta, esto es necesario ya que el conjunto de datos no cabe en la pantalla.

Para realizar la paginación es necesario obtener el número de registros que se tienen por dimensión. Al mostrar por primera vez el cubo de datos, se definen las variables `filaIniX`, `filaIniY` y `filaIniZ`, en las cuales se almacenará el número de registro que se comenzará a visualizar; y de la misma manera, se tienen las variables `filaFinX`, `filaFinY`, `filaFinZ`, para almacenar el último valor que se mostrará. Además, se tienen las variables `incrementoX`, `incrementoY`, `incrementoZ`, mismas que tienen la finalidad de saber el número de elementos que se mostrarán, de manera que cuando se requiera visualizar otra parte de los registros, se añade o se resta a las variables, dependiendo si se desea mostrar hacia adelante o devolverse hacia atrás; por último, las variables `maxFilasX`, `maxFilasY`, `maxFilasZ`, sirven para saber cuál es el máximo de filas que se tienen en cada uno de los ejes.

Se creó el método `paginacion` que permite volver a generar las consultas, sólo necesita recibir como parámetros el número de registro en el que se iniciará y finalizará en X, Y, y Z. Este método invoca al proceso para generación de consultas.

Cuando el usuario presiona el botón para devolverse a los primeros registros, se invoca al método paginación, se le proporciona el valor de inicio desde 1, hasta el valor que tenga la variable `incrementoX`, se tiene configurado que en el eje de las X y de las Y se permitan 9 registros, y en el eje de las Z 4 registros como máximo; en caso de que existan menos se ajusta automáticamente al número de registros que se tengan en cualquiera de los ejes.

A continuación se presenta el código contenido en el botón que permite desplazarse a los primeros registros.

```
paginacion(1, incrementoX, 1, incrementoY, 1, incrementoZ);
```

Si el usuario presiona el botón para desplazarse hasta el último registro, de la misma forma se llama al método paginación y se le proporciona como valor de inicio a cada eje el máximo de filas almacenado en las variables `maxFilasX`, `maxFilasY`, y `maxFilasZ` y se le resta el incremento almacenado en las variables `incrementoX`, `incrementoY`, `incrementoZ`; para el valor final se toma el que tiene en las variables `maxFilasX`, `maxFilasY` y `maxFilasZ`.

Cuando el usuario presiona el botón para devolverse a los registros anteriores, a las variables que contienen número de registro con el que comienza (`filaIniX`, `filaIniY` y `filaIniZ`), se les resta el incremento definido en las variables `incrementoX`, `incrementoY` e `incrementoZ`; a las variables que contienen el número de registro con el que se termina a las variables `filaIniX`, `filaIniY` y `filaIniZ` se les resta 1.

3 Análisis de resultados

Para el análisis de resultados se crearon varios cubos de datos y se eligieron las siguientes dimensiones del almacén de datos de ejemplo de *Oracle*: `ed_customers`, `ed_promotions`, `ed_products` y `ed_times`.

3.1 Visualización 3D

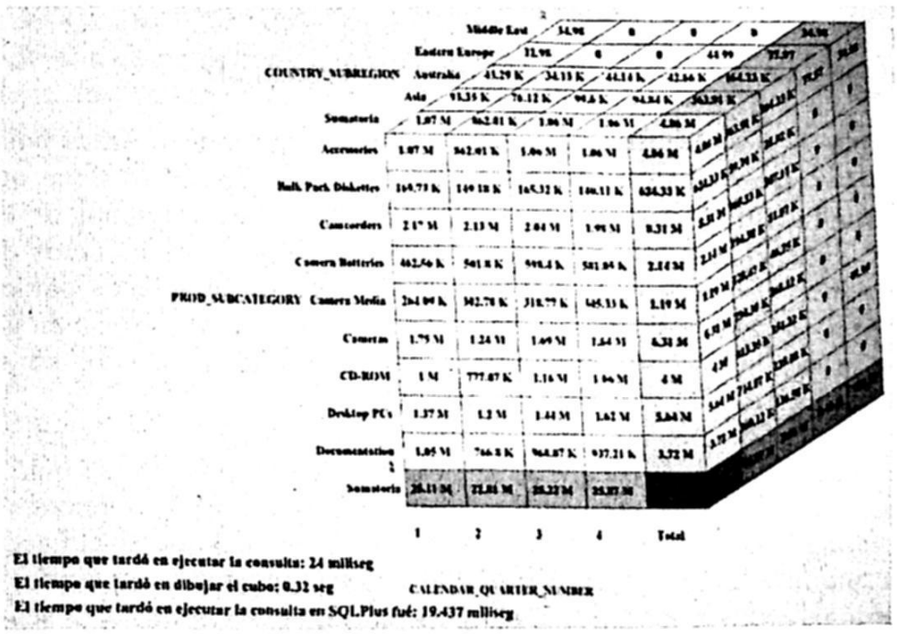

Fig. 2. Ejemplo cubo de datos por cuatrimestre, subcategoría y subregión.

En la figura 2 se muestra un ejemplo de cubo de datos en el que se eligieron los campos `calendar_quarter_number`, en el eje de las X, `prod_subcategory`, el eje de las Y, y `country_subregion`, en el eje de las Z. En el eje de la X se

muestran todos los cuatrimestres que se encuentran en el campo `calendar_quarter_number` (1 al 4), en el eje de las Y se muestran sólo las primeras 9 subcategorías de producto que se tienen almacenados en el campo `prod_subcategory`; para poder visualizar los demás datos, el usuario debe utilizar el componente para la paginación. Por último, en el eje de las Z, se muestran las primeras 4 subregiones del campo `country_subregion`, por lo que de igual manera, también se requiere de la paginación para visualizar el resto de los datos.

En el cubo de datos de prueba, los datos que están de color blanco representan la cantidad vendida en las diferentes subregiones; por ejemplo, en la subregión de Asia se vendieron 93.35 miles de accesorios en el primer cuatrimestre. Los que están de color amarillo representan los subtotales por subregión, en la subcategoría accesorios se vendieron en todas las ciudades 1.07 millones en el primer cuatrimestre. Los de color gris representan los totales por subregión y cuatrimestre, en la subcategoría accesorios se vendieron 4.06 millones en todas las subregiones y cuatrimestres. Los de color naranja, representa los totales por subregión y subcategoría, en el primer cuatrimestre, se vendieron 25.11 millones en todas las subregiones y subcategorías. El único de color azul representa la cantidad vendida total, por lo tanto se vendieron 98.21 millones.

El sistema reutilizó una vista materializada y se ejecutaron los datos desde el *DataStore* y no desde el almacén de datos. El tiempo que se tardó en ejecutar la consulta, por parte del sistema, fue de 24 milisegundos, y en dibujar el cubo 0.32 segundos. Se hizo una comparación con *SQLPlus* y el sistema propuesto, el primero tardó 19.437 milisegundos en mostrar los datos, por lo tanto, la diferencia de tiempos es de 4.563 milisegundos.

3.2 Manipulación del cubo de datos con Visualización 3D

El sistema también permite manipular el cubo de datos; es decir, es posible aplicar los operadores *Drill-down, Roll-up* y *rotate.* Para el primero, se tiene que bajar un nivel de detalle en la jerarquía, lo que permitirá visualizar las ventas de un producto por mes en lugar de por cuatrimestre; con *Roll-up* se agrupan los datos del cubo; es decir, siguiendo con el ejemplo anterior, si se encuentran los datos por cuatrimestre entonces se mostrarían los datos por año; y el operador *rotate* consiste en girar el cubo alrededor de uno de los tres ejes, visualizándose así otra cara del cubo para su análisis [9 - 10].

La figura 3, es un ejemplo de aplicación del operador *Drill-down*, en este caso se aplicó al cubo de datos mostrado en la figura 2, en este ejemplo se bajó el nivel de agrupación en todas las jerarquías quedando de la siguiente manera: número de semana, producto y estado. El sistema ejecutó los datos desde el almacén de datos, debido a que no existía una vista materializada con los datos requeridos. El tiempo que se tardó en ejecutar la consulta, por parte del sistema, fue de 20.72 segundos, y en dibujar el cubo 0.31 segundos. Al igual que en el ejemplo de la figura 2, se hizo una comparación con *SQLPlus* y el sistema propuesto, el primero tardó 21.36024 segundos en mostrar los datos, por lo tanto, la diferencia de tiempos es de 0.64024 segundos.

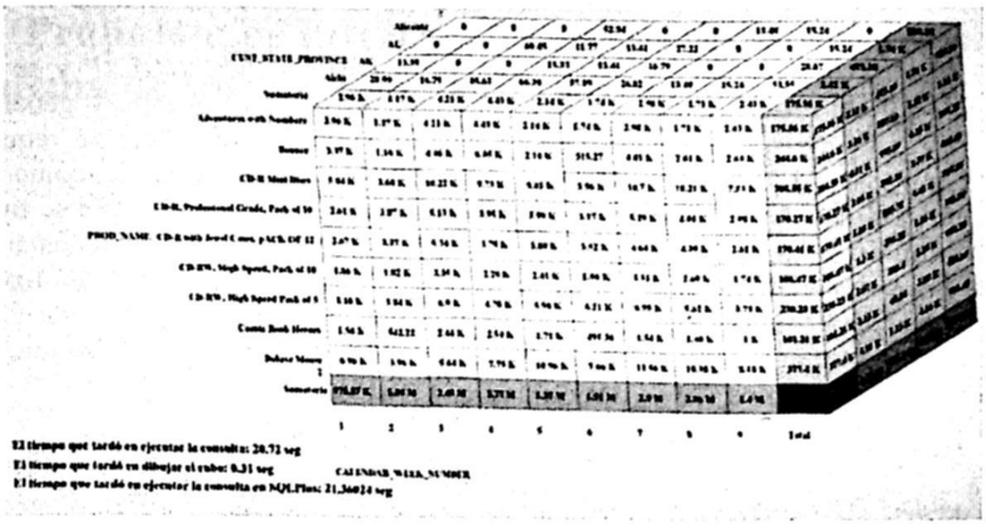

Fig. 3. Ejemplo cubo de datos por número de semana, producto y estado.

Para aplicar los operadores, el sistema reutiliza las vistas previamente materializadas, cargándolos datos del *DataStore* directamente al cubo; en caso contrario, los datos se obtienen del almacén debido a la limitante en espacio de almacenamiento requerido.

3.3 Aplicando la paginación

Continuando con el ejemplo de la figura 3, en este caso se puede ver que en el eje de las X se tiene dos ">" de color rojo, y en el de las Y se tiene dos "v"; esto indica que existen más datos que se pueden visualizar, de la misma manera en el eje de las Z se tienen dos "^" indicando lo mismo.

En caso de que el usuario requiera aplicar la paginación en el eje de las X, sólo se actualizan los datos de la cara de frente; en el ejemplo de la figura 3 se mostrarían los siguientes números de semana que van después del número 9. En la paginación sobre el eje de las Y, se actualizan todas las caras del cubo, debido a que en la cara de arriba ya no se mostraría el primer producto, sino que se mostraría el producto que va después de "deluxe mouse". La cara del lado derecho también requiere actualización, debido a que se deben mostrar los totales correspondientes a los productos siguientes. En la paginación sobre el eje de las Z, no se actualiza la cara de frente; sin embargo, tanto en la cara del lado de arriba como la del lado derecho, se requiere la actualización al mismo tiempo, ya que deben de coincidir con los estados mostrados en el eje de las Z.

4 Conclusiones

La visualización tridimensional permite visualizar los datos de una manera ordenada, sin anidaciones del conjunto de dimensiones y por lo tanto, se aprovecha el espacio en

pantalla para mostrar más datos, y de esta forma facilitar el análisis de los datos contenidos en el cubo de datos.

Se decidió implementar el sistema sobre una arquitectura *HOLAP* y se eligió el almacenamiento relacional, con el fin de tener los datos de detalle y generar vistas materializadas de sólo los atributos que elija el usuario; esto último no requiere pre-calcular todas las demás combinaciones de las jerarquías, teniendo como objetivo mejorar los tiempos de respuesta, sin utilizar en exceso el espacio que se tiene en el *DataStore*. El tiempo de ejecución de las consultas realizadas por el sistema en cubos de datos, conteniendo datos almacenados en una vista materializada, fue ligeramente mayor al tiempo de ejecución de la misma consulta sobre *SQLPlus* de Oracle; la diferencia reportada en las distintas pruebas realizadas en nuestro sistema, es en el orden de milisegundos.

5 Agradecimientos

Agradecemos el apoyo de CONACYT con la beca para los estudios de la maestría al primer autor.

Este trabajo fue parcialmente financiado por PROMEP, bajo el proyecto PROMEP/103.5/12/4633.

Referencias

1. ZhaoHui Tang, Jamie MacLennan. Data Mining with SQL Server 2005, Chapter 11: Mining *OLAP* Cubes. Wiley Publishing, Inc., (2005).
2. G. Satyanarayana Reddy, Rallabandi Srinivasu, M. Poorna Chander Rao, and Sri-kanth Reddy Rikkula. Data warehousing, data mining, *OLAP* and *OLTP* technologies are essential elements to support decision-making process in industries. International Journal on Computer Science Engineering, 2(9):2865 – 2873, 2010.
3. William Harvey Inmon. Building the Data Warehouse Third Edition, Chapter 2: The Data Warehouse Environment. Wiley Publishing, Inc., 2002.
4. Narasimhaiah Gorla. Features to consider in a data warehousing system. Communications of the ACM, 46(11):111 – 115, 2003.
5. Erik Thomsen. The Functional Requirements of *OLAP* Systems, Chapter 1. Wiley Computer Publishing, 2002.
6. The Lightweight Java Game Library. The lightweight java game library (lwjgl). http://lwjgl.org, 2012.
7. Ashish Gupta and Inderpal Singh Mumick, Materialized Views, Techniques, Implementations, and applications, Part 1: Rejuvenation of Materialized Views, The MIT Press Cambridge, Massachusets, London, England, 1999.
8. Paul Lane, Oracle Database Data Warehousing Guide 10g Release 2, SQL for aggregation in Data Warehouses, 20-1 – 20-24 2005.
9. Patrick Marcel and Place Jean Jaurs. Modeling and Querying Multidimensional Databases: An Overview, Networking and Information Systems Journal, 1999.
10. Jim Gray, Surajit Chaudhuri, Adam Bosworth, et. al. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals, Kluwer Academic Publishers, 1997.