Universidad de las Américas

Facultad de Ingeniería

DATA WAREHOUSING: APLICACIÓN DE LA METODOLOGÍA DE MICROSOFT PARA LA IMPLEMENTACIÓN DE UN DATA WAREHOUSE EN UNA EMPRESA DE SERVICIOS TRANSACCIONALES FINANCIEROS Y NO FINANCIEROS

Trabajo de titulación presentado en conformidad a los requisitos para obtener el título de Ingeniero de Sistemas en Computación e Informática

Ing. Byron Rodríguez

Susana M. Paredes F.

2000

Agradecimiento

Agradezco a Dios por haberme permitido culminar con éxito el proyecto emprendido.

Agradezco también a mi director de tesis, el Ing. Byron Rodríguez, por la ayuda y el apoyo que en todo momento me brindo y a todos mis amigos que de una u otra manera colaboraron con el desarrollo de esta tesis.

Dedicatoria

Dedico este trabajo a mis padres y a mi novio por su apoyo incondicional en esta etapa de mi vida.
Muy especialmente a mi Madre por su amor, sus sabios consejos y su comprensión.
A mi querido tutor, que supo guiarme correctamente para la elaboración de este trabajo.

RESUMEN EJECUTIVO

Este trabajo esta orientado a introducir todos los elementos que intervienen en una solución de data warehousing, para comprender lo que también se conoce como tecnología del data warehouse.

El propósito de un data warehouse es el de asistir a la alta gerencia de una organización en el entendimiento del pasado para planear su futuro.

Debido a que esta tarea tiene particular importancia en nuestros tiempos, el departamento de tecnología de la información debe prestar singular cuidado en el proceso de su implementación, pues trabaja con información crítica del negocio, se enfrenta a nuevas tecnologías de hardware y software, volúmenes siempre crecientes de datos; en definitiva, a la implementación de un complejo sistema en el que confluyen conceptos del negocio y conceptos técnicos.

En este documento se muestra de manera práctica los pasos a seguir en la Metodología recomendada por Microsoft para el desarrollo de data warehouse utilizando las herramientas de esta casa de software.

Una implementación exitosa requiere de un conjunto de pasos estructurados que guíen el proceso. El objetivo principal de este trabajo es de proporcionar una demostración práctica de una metodología que cumpla con este objetivo.

El objetivo principal de este documento tiene como objetivos específicos:

- Presentar una clara concepción del concepto de data warehousing
- Presentar consideraciones para llegar a un rendimiento óptimo de un data warehouse
- Analizar diversos aspectos que deben tomarse en cuenta para obtener una base de datos confiable y clara.

La tesis esta dirigida a personas con diferentes perfiles, desde usuarios del negocio hasta personas técnicas de sistemas.

1. Conceptos Generales a cerca de data warehouse.
Aquí se presenta una visión general de los conceptos principales de data warehouse, las características principales de un data warehouse y de un data mart. Además, se presenta los pasos a seguir durante todo el desarrollo del data warehouse.

2. Características de las herramientas de Microsoft SQL Server 7.0
En vista de que se ha escogido una herramienta tan versátil y fácil de manejar como Microsoft SQL Server 7.0, se presenta información de sus principales características y facilidades.

3. Determinar los requisitos del negocio, del usuario y técnicos.
En este capítulo se determina los requisitos del negocio y de usuario, así como los costos del proyecto.

4. Diseñar y generar la base de datos.
Se presenta una guía para identificar y seleccionar las tablas de hechos y de dimensiones, así como las técnicas de diseño que se pueden escoger. Seguiremos con la creación de un esquema de base de datos y la implementación de la misma.

5. Extraer y cargar datos.
Se presenta en forma teórica y práctica los pasos a seguir para cumplir con la extracción y carga de datos. Además, algunos puntos a tomar en cuenta para diseñar y procesar agregados

6. Consultar y mantener el Data warehouse y las base de datos OLAP.
Aquí se encontrará como funcionan las bases de datos multidimensionales y la tecnología OLAP.
Algunas pautas para mantener el data warehouse.
Se presentará paso a paso el procedimiento para utilizar como herramienta de consulta del data warehouse a Microsoft Excel 2000.

7. Conclusiones y Recomendaciones.
Las conclusiones y recomendaciones que se pueden obtener a partir de este trabajo se encuentran en este capitulo.

El desarrollo de este trabajo ha sido basado en varias fuentes bibliográficas: libros, artículos y publicaciones en Internet mencionados durante todo el documento y un compendio de los mismos al final de trabajo en la sección de bibliografía; así también de información recolectada de entrevistas a implementadores y usuarios.

INDICE

INTRODUCCION.

El ambiente competitivo en las empresas de los 90s así como el avance tecnológico en materia de sistemas de información, han provocado un "nuevo" enfoque en el tratamiento y proceso de la información ejecutiva, la cual es un elemento vital hoy en día como soporte en el proceso de toma de decisiones.

El concepto Data Warehousing, o el proceso de contar con la información más importante de la empresa (incluyendo la histórica), en un sólo lugar, ha logrado convertirse en una valiosa herramienta y clave desde el punto de vista tecnológico.

La definición de Data Warehouse tiene múltiples vertientes, según W.H. Inmon, uno de los precursores del concepto data warehouse,

> *"Data Warehouse es un sistema orientado al usuario final, integrado, con variaciones de tiempo y sobretodo una colección de datos como soporte al proceso de toma de decisiones".*

De acuerdo con algunas otras organizaciones, data warehouse es una arquitectura. Para otros, es simplemente un Data warehouse (separado y que no interfiere con los sistemas operativos actuales de una empresa) para satisfacer las diversas consultas y requerimientos de información. Para algunos otros, data warehouse es un proceso que agrupa datos desde múltiples fuentes heterogéneas, incluyendo datos históricos para soportar la continua necesidad de consultas, reportes analíticos y soporte de decisiones.

Data warehouse difiere de las bases de datos operacionales que soportan aplicaciones con el Proceso Transaccional en Línea de muy diversas formas. Data Warehouse incluye lo siguiente:

- Orientado al usuario final.
- Administra y maneja un gran volumen de información.
- Información sumarizada y agregada.
- Integra y asocia información desde múltiples fuentes y orígenes.

Data Warehouse como un Sistema de Misión Crítica

Debido a que las empresas actualmente demandan mayor información oportuna, confiable y completa, desde un acceso a la información más importante, esto hace tornar este tipo de sistemas como de Misión Crítica.

Los requerimientos de un sistema de Misión Crítica principales a considerar son:

- Disponibilidad.
- Confiabilidad y consistencia.
- Robustez.
- Estándar.
- Basado en los requerimientos del negocio.
- Compatibilidad con tecnología actual y una infraestructura sólida.
- Utilizado sobre bases diarias, es decir, para el proceso del día con día.

- Amigable.
- Performance (Rapidez).
- Auditable.
- Seguro.

Propósito del Data Warehouse

El propósito del Data Warehouse es asistir al ejecutivo en el entendimiento del pasado y contar con los elementos para la planeación del futuro de corto, mediano y largo plazo.

Los ejecutivos y administradores buscan respuestas a preguntas como:

- ¿ Qué están comprando nuestros clientes? ¿Qué NO están comprando?
- ¿ Qué está haciendo nuestra competencia?
- ¿ Cómo están nuestros costos por cada línea de producto, comparados con los últimos tres años?
- ¿ Qué factores causan incrementos en los costos?
- ¿ Afectan nuestros márgenes?

Una pregunta nos lleva a la siguiente, los ejecutivos quieren tener respuestas a preguntas cruciales y tomar mejores decisiones.

Un Data Warehouse ayuda a resolver estas preguntas de una forma eficiente. El proceso de construcción de un Data Warehouse, en la forma más general, pasa por las siguientes fases:

1. Análisis
2. Definición
3. Diseño
4. Desarrollo
5. Instrumentación

En muchas ocasiones, un Data Warehouse se utiliza como base de los sistemas de ayuda a la toma de decisiones. Se diseña para superar algunos de los problemas que una organización encuentra cuando intenta realizar un análisis estratégico mediante la misma base de datos que utiliza para realizar el proceso de transacciones en línea.

1. CONCEPTOS GENERALES A CERCA DE DATA WAREHOUSE

SISTEMAS TRANSACCIONALES

Un sistema de proceso de transacciones en línea se caracteriza por tener un gran número de usuarios simultáneos que agregan y modifican datos de manera activa. La base de datos representa el estado de una función particular de negocio en un momento específico, como en un sistema de reservas de una línea aérea. Sin embargo, el gran volumen de datos mantenido por muchos sistemas OLTP puede desbordar a una organización. Puesto que el crecimiento de una base de datos se hace mayor cuanto más complejos son los datos, el tiempo de respuesta puede deteriorarse rápidamente debido a la lucha por los recursos disponibles. Un sistema OLTP típico tiene muchos usuarios que agregan datos nuevos a la base de datos mientras unos pocos generan informes a partir de la base de datos. A medida que aumenta el volumen de datos, se tardará más en generar los informes.

Normalmente, los sistemas OLPT están diseñados específicamente para administrar el proceso de transacciones y minimizar los requisitos de almacenamiento en disco mediante una serie de tablas normalizadas y relacionadas. Sin embargo, cuando los usuarios necesitan analizar los datos, una serie de problemas impiden a menudo su uso:

- Los usuarios no pueden entender las complejas relaciones entre las tablas y, en consecuencia, no pueden generar consultas.
- Las bases de datos de la aplicación pueden estar divididas entre varios servidores y dificultan a los usuarios encontrar las tablas en una primera inspección.
- Las restricciones de seguridad pueden impedir a los usuarios el acceso a los datos detallados que necesitan.
- Los administradores de la base de datos impiden las consultas pesadas en sistemas OLPT, para evitar que los usuarios que realizan el análisis puedan disminuir el rendimiento de las bases de datos de producción de gran importancia.

DATA WAREHOUSE COMO SOLUCIÓN PARA ANÁLISIS DE DATOS

Se puede mejorar el tiempo de respuesta para las consultas e informes copiando la base de un sistema OLTP a un servidor de informes sobre una base programada regularmente, pero aun así, se pueden tener datos tan detallados como los de los sistemas OLTP que al momento de un reporte se resumen cada vez que se genera un informe generando un mayor tiempo de ejecución y menor satisfacción del usuario.

Además, muchas organizaciones almacenan los datos en varios sistemas de bases de datos heterogéneos. Realizar informes es más difícil, no sólo porque los datos están almacenados en sitios distintos sino porque, además, pueden encontrarse en diferentes formatos.

La creación de data warehouse y el proceso analítico en línea (OLAP) ofrecen soluciones a estos problemas. La creación de data warehouse es un enfoque en el que varias bases de datos OLTP se migran a un Data warehouse homogéneo y separado.

Los data warehouse proporcionan estos beneficios a los usuarios que realizan el análisis:

- Los datos se organizan de manera que faciliten las consultas de análisis en lugar del proceso de transacciones.
- Pueden resolverse las diferencias entre las estructuras de datos existentes en varias bases de datos heterogéneas.
- Al mover los datos de una base OLTP a un Data warehouse se pueden aplicar reglas de validación y consolidación de los datos.
- Los aspectos de seguridad y rendimiento pueden ser resueltos sin cambiar los sistemas de producción.

Algunas veces, las organizaciones desean mantener almacenes de datos más pequeños y más especializados en un tema, llamados centrales de datos o Data marts. A diferencia de un Data warehouse, que normalmente encapsula todos los datos de análisis de una empresa, un Data mart es, un subconjunto de datos de la empresa cuyo objetivo es constituir un conjunto más pequeño de funciones de usuario o de negocio.

Las diferencias de un Data Warehouse con un sistema tradicional las podríamos resumir en el siguiente esquema:

SISTEMA TRANSACCIONAL	DATA WAREHOUSE
Predomina la actualización	Predomina la consulta
La actividad más importante es de tipo operativo (día a día)	La actividad más importante es el análisis y la decisión estratégica
Predomina el proceso puntual	Predomina el proceso masivo
Mayor importancia a la estabilidad	Mayor importancia al dinamismo
Datos en general desagregados	Datos en distintos niveles de detalle y agregación
Importancia del dato actual	Importancia del dato histórico
Importante del tiempo de respuesta de la transacción instantánea	Importancia de la respuesta masiva
Estructura relacional	Visión multidimensional
Usuarios de perfiles medios o bajos	Usuarios de perfiles altos
Explotación de la información relacionada con la operativa de cada aplicación	Explotación de toda la información interna y externa relacionada con el negocio

Una de las claves del éxito en la construcción de un Data Warehouse es el desarrollo de forma gradual, seleccionando a un departamento usuario como piloto y expandiendo progresivamente el almacén de datos a los demás usuarios. Por ello es importante elegir este usuario inicial o piloto, siendo importante que sea un departamento con pocos usuarios, en el que la necesidad de este tipo de sistemas es muy alta y se puedan obtener y medir resultados a corto plazo.[1]

LA TECNOLOGÍA OLAP APOYA AL DATA WAREHOUSE

Si bien en un Data warehouse o en un Data mart se almacenan los datos para su análisis, OLAP es la tecnología que permite a las aplicaciones cliente el acceso eficiente a estos datos. OLAP proporciona los siguientes beneficios a los usuarios que realizan análisis de los datos:

- Agregación previa de los datos frecuentemente consultados, que permite que el tiempo de respuesta de las consultas ad hoc sea muy pequeño.
- Un modelo intuitivo multidimensional de datos que facilita su selección, recorrido y exploración.
- Una eficaz herramienta para crear nuevas vistas de datos basada en un rica matriz de funciones de calculo ad hoc.
- Tecnología para administrar la seguridad, la administración de consultas cliente-servidor y el almacenamiento de los datos en la memoria caché, y facilidades para optimizar el rendimiento del sistema según las necesidades del usuario.

Los términos Data Warehouse y almacén de datos son un mismo concepto que se usaran de manera intercambiable, lo mismo que Data Mart y central de datos.

CARACTERÍSTICAS DE LOS DATA WAREHOUSE

Un Data Warehouse puede ayudar a la toma de decisiones y a las aplicaciones de proceso analítico de datos (OLAP) porque proporciona datos que son:

1. Datos consolidados y coherentes

Un Data warehouse consolida datos operacionales a partir de una gran variedad de orígenes de datos con convenciones coherentes de nomenclatura, medidas, atributos físicos y semántica.

Por ejemplo, las aplicaciones pueden utilizar datos de formatos diferentes: los datos pueden almacenarse en formato juliano o gregoriano; los datos verdaderos o falsos pueden representarse como uno o cero, encendido o apagado, verdadero o falso, o positivo o negativo. Aplicaciones diferentes también pueden utilizar términos distintos para describir el mismo tipo de datos.

Los datos deben almacenarse en el Data warehouse en un formato simple y adecuado, aprobado por los analistas del negocio, a pesar de las variaciones de los orígenes operacionales externos, proporcionando al analista un mejor entendimiento de su negocio.

[1] Que es un data warehouse? http://www.map.es/csi/silice/DW22.html

2. Datos orientados a temas

Los orígenes de datos operacionales de la organización tienden a mantener una gran cantidad de datos acerca de varias funciones relacionadas con el negocio, como registros de usuario, información de producto, etc. La mayor parte de esta información, además, está mezclada con datos irrelevantes para el negocio, y se organiza de forma que dificulta la consulta de los datos.

El Data warehouse, a partir de los orígenes de datos operacionales, sólo organiza la información empresarial clave, de forma que esté disponible para el análisis de negocio.

3. Datos históricos

Los datos de sistemas OLTP representan correctamente el valor actual en cualquier momento, en cambio, los datos de un Data warehouse son precisos para algún momento anterior en el tiempo porque representan información histórica.

Normalmente el data warehouse, representa datos en un largo periodo, quizá superior a los diez años.

A menudo, los sistemas OLTP contienen sólo los datos actuales, porque mantener grandes volúmenes de datos para representar diez años de información en un sistema OLTP puede afectar al rendimiento.

4. Datos de sólo lectura

Una vez que los datos se han pasado correctamente al Data warehouse, no cambian a menos que estuvieran incorrectos desde su sitio de origen. Puesto que los datos del data warehouse representan un momento del tiempo, nunca deben actualizarse. El borrado, la inserción y la actualización solo son aplicados en el proceso de carga de datos, por ende las únicas operaciones que se realizan en un Data warehouse, cuando ha sido configurado, son la carga y la consulta de datos.

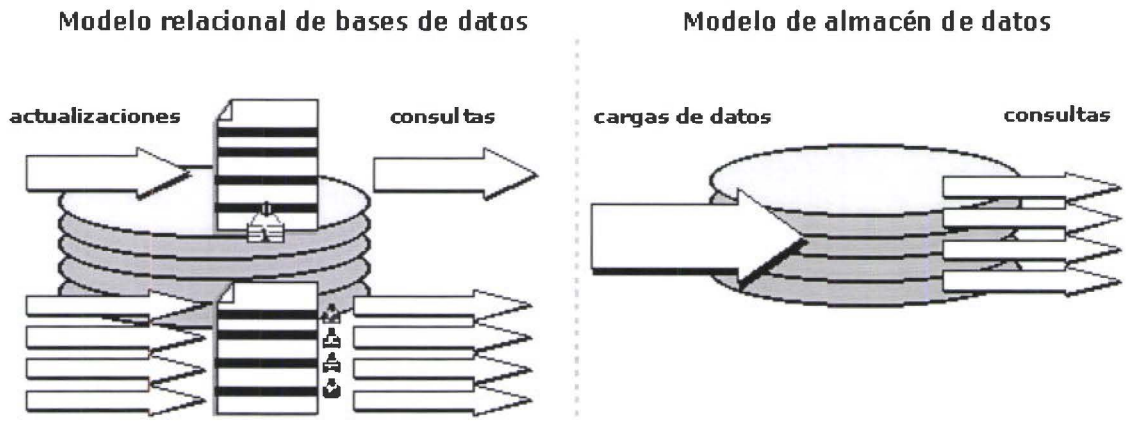


Gráfico 1.

Granularidad de los datos

Una diferencia significativa entre un sistema OLTP u operacional y un Data Warehouse es la granularidad de los datos que almacenan. Un sistema operacional almacena datos con el menor grado de granularidad: con el mayor grado de detalle, en cambio, un data warehouse almacena datos en diferentes grados de granularidad o resumen, dependiendo de los requisitos de los datos del negocio.[2]

Si la empresa necesita datos para apoyar el proceso de diseño estratégico, sólo se necesitan datos muy resumidos. Cuanto menor es el grado de granularidad de los datos requerido por la empresa, mayor es el número de recursos (almacenamiento de los datos) requeridos para generar el Data warehouse. Los distintos grados de resumen por orden de aumento de la granularidad son:

- Datos operativos actuales
- Datos operativos históricos
- Datos agregados
- Metadatos

Los datos operacionales e históricos se toman, sin modificar, directamente de los sistemas operacionales. Los datos históricos son datos de nivel operacional sobre los que no se realizan regularmente consultas y, a menudo, son archivados en un dispositivo de almacenamiento secundario.

Los datos agregados o de resumen, son una versión filtrada de los datos operacionales actuales. El diseño del Data warehouse afecta a la forma de agregar los datos. Se debe tener muy en cuenta el periodo utilizado para agregar los datos (por ejemplo, semanal, mensual, etc.) y las partes de los datos operacionales que deben resumirse.

Los metadatos no contienen datos operacionales, pero se utilizan para documentar la manera de generar el Data warehouse, se pueden describir estructuras del Data warehouse, orígenes de datos, reglas utilizadas para resumir los datos en cada nivel y cualquier otra transformación de los datos realizada desde los sistemas operacionales.

BENEFICIOS DEL DATA WAREHOUSE

Resumiendo los beneficios que un Data Warehouse puede aportar son:

- Proporciona una herramienta para la toma de decisiones en cualquier área funcional, basándose en información integrada y global del negocio.
- Facilita la aplicación de técnicas estadísticas de análisis y modelización para encontrar relaciones ocultas entre los datos del almacén; obteniendo un valor añadido para el negocio de dicha información.
- Proporciona la capacidad de aprender de los datos del pasado y de predecir situaciones futuras en diversos escenarios.

[2] Para obtener mayor información revisar Microsoft SQL Server Books Online

- Simplifica dentro de la empresa la implantación de sistemas de gestión integral de la relación con el cliente.

- Supone una optimización tecnológica y económica en entornos de Centro de Información, estadística o de generación de informes con retornos de la inversión espectaculares.

DATA MARTS

"A un Data Marts se lo puede definir como un subconjunto de un Data warehouse, almacenado dentro de su propia base de datos. Un Data mart contiene datos orientados a los departamentos o a un área específica de negocio. Los datos pueden existir como detalles o como resúmenes. Las centrales de datos pueden llenarse con datos tomados directamente de los orígenes operacionales, de manera similar a lo que ocurre en un Data warehouse, o con los datos tomados del propio Data warehouse. Puesto que el volumen de datos de la central de datos es menor que el del Data warehouse, el proceso de consulta es, a menudo, más rápido"[3]

Las características de las centrales de datos incluyen:

- Implementación más rápida y simple.
- Menor costo de implementación.
- Satisfacción de necesidades para una unidad o función específica del negocio.
- Protección de información sensible almacenada en cualquier sitio del Data warehouse.
- Menor tiempo de respuesta debido al menor volumen de datos.
- Distribución de las centrales de datos entre las organizaciones.
- Construcción de abajo a arriba.

Las divisiones departamentales o regionales determinan, con frecuencia, si se usan almacenes o centrales de datos. Por ejemplo, si los jefes de diferentes regiones de ventas necesitan datos de una única región, puede ser conveniente generar centrales de datos que contengan datos regionales específicos. Si los jefes regionales necesitan tener acceso a todos los datos de la organización, será necesario un Data warehouse.

Generar Data warehouse a partir de Data Marts

Los almacenes de datos pueden generarse utilizando tanto un enfoque de arriba a abajo como de abajo a arriba. El enfoque de arriba a abajo describe el proceso de generación de un Data warehouse para una organización completa, para contener datos de varios orígenes heterogéneos y operacionales.

[3] Tomado de Microsoft Technet edición abril de 1999

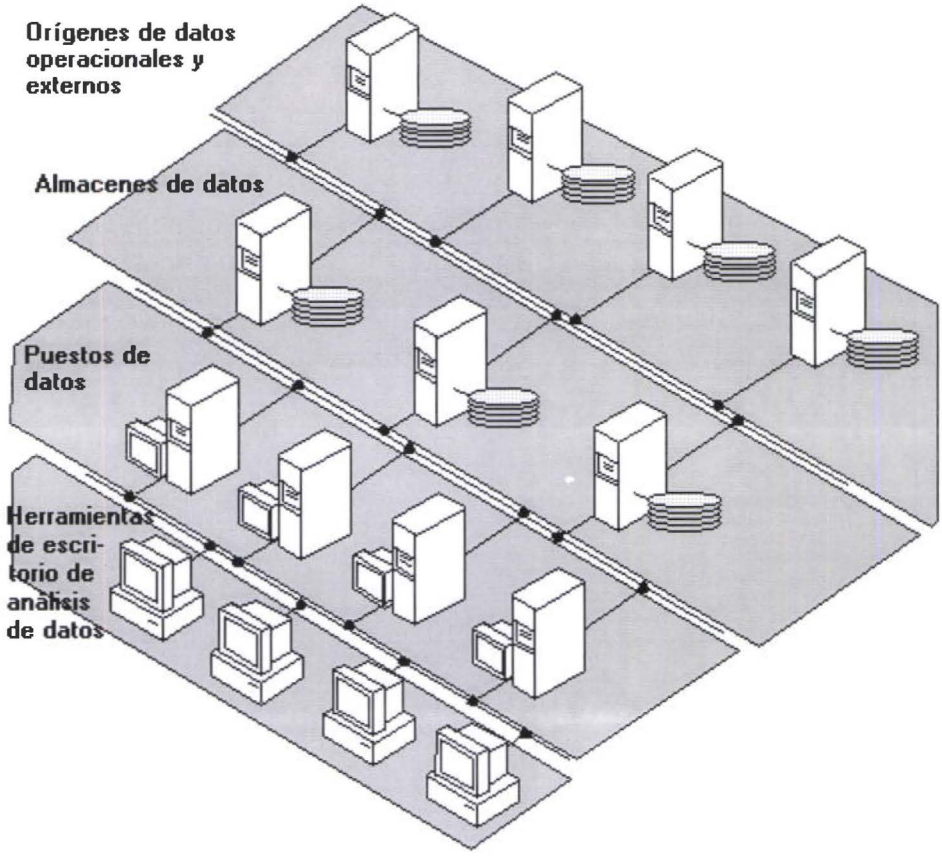


Gráfico 2.

El enfoque de abajo a arriba describe el proceso de generación de centrales de datos para departamentos, o áreas de negocio específicas, y, a continuación, las combina para proporcionar los datos para toda la organización. La generación de un Data warehouse mediante el enfoque de abajo a arriba, mediante la implementación de centrales de datos, es, a menudo, más simple porque es menos ambiciosa.

Un enfoque práctico para usar centrales y almacenes de datos implica almacenar todos los datos detallados dentro del Data warehouse y las versiones resumidas en las centrales de datos. Cada central de datos contiene datos resumidos por cada división funcional dentro del negocio con lo cual se reduce el volumen de datos en la central de datos.

Consideraciones acerca de los Data Marts

Las centrales de datos pueden ser alternativas útiles a los almacenes de datos, pero entre los temas que se deben considerar antes de su implementación están los siguientes:

- Hardware y software adicional.
- Tiempo necesario para llenar regularmente cada central de datos.
- Coherencia con otras centrales de datos y el Data warehouse.
- Acceso a la red (sí cada central de datos se ubica en una región geográfica distinta).

DISEÑAR Y CONSTRUIR UN DATA WAREHOUSE Y UN SISTEMA OLAP

La metodología utilizada para la construcción del data warehouse, en forma general, incluye los siguientes pasos:

- Determinar los requisitos del negocio, del usuario y técnicos.
- Diseñar y generar la base de datos.
- Extraer y cargar los datos en el data warehouse.
- Diseñar y procesar las adiciones utilizando herramientas OLAP.
- Consultar y mantener el data warehouse y las bases de datos OLAP.

En los siguientes capítulos se desarrollarán uno por uno estos cinco pasos que forman la metodología recomendada por Microsoft para el desarrollo de un data warehouse usando como herramientas sus productos.

2. CARACTERISTICAS DE LAS HERRAMIENTAS DE SQL SERVER 7.0

El propósito de este capítulo, es presentar las principales características de Microsoft SQL Server 7.0, por ser la herramienta utilizada, para el desarrollo de un data warehouse. El detalle presentado está basado en información obtenida de la publicación de Microsoft Technet de marzo de 1999.

COMPATIBILIDAD CON EL ALMACENAMIENTO DE DATOS DE LOS SERVICIOS DE TRANSFORMACIÓN DE DATOS.

Al utilizar los Servicios de transformación de datos[4], se puede importar y exportar datos entre varios orígenes heterogéneos utilizando una arquitectura basada en OLE DB, y transferir las bases de datos y los objetos de base de datos (por ejemplo, índices y procedimientos almacenados) en tres equipos en los que se ejecuta la versión 7.0 de Microsoft SQL Server. También puede utilizar las funcionalidades de transformación de datos de DTS para crear un Data warehouse a partir de un sistema de proceso de transacciones en línea (OLTP). Puede generar almacenes y centrales de datos en SQL Server si importa y transfiere los datos desde varios orígenes heterogéneos interactivamente o automáticamente en función de una programación regular.

Entre los componentes de los DTS se incluyen el Asistente para importación con DTS, el Asistente para exportación con DTS y el Diseñador DTS, que están disponibles a través del Administrador corporativo de SQL Server. DTS incluye también interfaces de programación COM que se pueden utilizar para crear aplicaciones personalizadas de importación, exportación y transformación.

Una transformación es el conjunto de operaciones aplicadas a los datos de origen antes de que se almacenen en el destino durante el proceso de creación de un Data warehouse. Por ejemplo, la funcionalidad de transformación de los DTS permite calcular nuevos valores de una o más columnas de origen o descomponer una única columna en varios valores que se almacenarán en columnas de destino independientes. Por tanto, las transformaciones simplifican la implementación de la validación, normalización y mejora de datos complejos durante la importación y exportación.

Los Servicios de transformación de datos permiten importar, exportar o transformar datos en un proceso que puede guardarse como un paquete. Cada paquete define un flujo de trabajo que incluye una o más tareas que se ejecutan en una secuencia coordinada de pasos. Las tareas pueden copiar datos de un origen a un destino, transformar datos mediante una secuencia de comandos de Microsoft ActiveX, ejecutar una instrucción SQL en el servidor o, incluso, ejecutar un programa externo. Las tareas también pueden transferir objetos de base de datos entre equipos en los que se ejecuta la versión 7.0 de SQL Server.

[4] DTS, Data Transformation Services

El paquete DTS puede crearse manualmente mediante la utilización de un lenguaje que admita la automatización OLE, como Microsoft Visual Basic, o interactivamente mediante los asistentes de los Servicios de transformación de datos o el Diseñador DTS. Una vez que un paquete DTS que se ha creado y guardado, está completamente autocontenido, y puede recuperarse y ejecutarse mediante el Administrador corporativo de SQL Server o el programa **dtsrun**.

Los paquetes DTS se pueden almacenar en el Depósito de Microsoft , lo que proporciona la capacidad de registrar el linaje de los datos. Esto permite determinar el origen de los datos y las transformaciones que se aplican a esos datos. Puede realizarse un seguimiento del linaje de los datos en las filas y los paquetes de una tabla, y proporcionar un recorrido de auditoría completo de la transformación de los datos y de la información de ejecución de paquetes DTS en el Data warehouse.

El Diseñador DTS es un entorno de diseño gráfico para crear y ejecutar conjuntos complejos de transformación de datos y flujos de trabajo, como preparación para el traslado de los datos a un Data warehouse. Los usuarios con experiencia pueden utilizar el Diseñador DTS para integrar, consolidar y transformar datos heterogéneos de varios orígenes. Los paquetes creados pueden guardarse en la base de datos **msdb** de SQL Server, en el Depósito o en un archivo de almacenamiento con estructura COM.

Los objetos visuales utilizados por el Diseñador DTS se basan en el modelo de objetos de DTS, una API que incluye objetos, propiedades, métodos y colecciones diseñadas para los programas que copian y transforman datos desde un origen de datos OLE DB a un destino OLE DB. A este modelo de objetos se obtiene acceso a través de secuencias de comandos de ActiveX desde dentro del Diseñador DTS y mediante programas externos escritos en lenguajes como Visual Basic y Microsoft Visual C++. También se puede obtener acceso a los programas personalizados a través del Diseñador e incluir sus tareas e iconos como parte del paquete. Puesto que el Diseñador DTS obtiene acceso a un determinado modelo de programación, realiza la mayor parte del trabajo de programación.

COMPATIBILIDAD CON EL ALMACENAMIENTO DE DATOS DE LOS SERVICIOS OLAP

Los Servicios OLAP de Microsoft SQL Server proporcionan servicios de proceso analítico en línea (OLAP) a las aplicaciones, como ya se lo indico en el capítulo anterior, pero además tiene las siguientes características.

Facilidad de uso

Los servicios OLAP de Microsoft SQL Server ofrecen asistentes, editores y toda la información necesaria para que la tecnología OLAP sea fácil de usar.

Material de tutorial e información general

Puede utilizar el tutorial en línea para familiarizarse con el Administrador de OLAP en, aproximadamente, una hora. El tutorial, que está diseñado para usuarios de OLAP

noveles y avanzados, le muestra paso a paso la forma de crear un cubo básico y la forma de ejecutar operaciones más avanzadas como crear particiones y cubos virtuales. El tutorial es una excelente herramienta para aprender OLAP y conocer el funcionamiento y las características del Administrador de OLAP.

También podrá encontrar información acerca de OLAP y los Servicios OLAP, versión 7.0, en el panel de la derecha del Administrador de OLAP.

Metadatos y vista de datos

Podrá ver metadatos y propiedades de objetos, y examinar los datos del cubo en el panel de la derecha del Administrador de OLAP a medida que se desplaza por la vista de árbol del panel de la izquierda.

Asistente para cubos

Podrá generar todas las estructuras necesarias para crear un cubo OLAP mediante un asistente muy fácil de utilizar. El asistente le guía por todo el proceso de diseño e implementación del cubo, desde la asignación de orígenes de datos y la creación de dimensiones hasta la definición de medidas.

Editor de cubos

Podrá modificar estructuras de cubos existentes y crear otras nuevas mediante sencillas operaciones de arrastrar y colocar. El editor de cubos es un complemento del Asistente para cubos que le permitirá revisar los cubos que haya creado con el asistente o crear otros nuevos con gran rapidez. El editor integra varias vistas del cubo:

- La vista de árbol muestra una vista jerárquica de las dimensiones, niveles de dimensión, medidas y miembros calculados del cubo.
- El panel Esquema muestra las tablas de hechos del Data warehouse, las tablas de dimensiones y las relaciones existentes entre ellas que definen el esquema del cubo. Puede modificar el esquema del cubo en este panel.
- El panel Examinar muestra los datos del cubo. Puede cambiar a esta vista para comprobar su trabajo al mismo tiempo que realiza cambios en la estructura del cubo. Los datos de ejemplo se generan si todavía no se ha procesado la nueva estructura del cubo; los datos reales aparecen si la estructura del cubo no ha cambiado desde que el cubo se procesó por última vez.
- El panel Propiedades permite el acceso a las propiedades de objetos tales como medidas, miembros calculados y elementos de dimensiones privadas.

Asistente para dimensiones

Con el Asistente para dimensiones, podrá crear una dimensión compartida que cualquier cubo pueda utilizar o una dimensión privada para su empleo sencillo y rápido en un único cubo. Podrá asignar columnas de la tabla de dimensiones de la base de datos a niveles de dimensión o utilizar el generador integrado de dimensiones de tiempo para crear varias dimensiones de tiempo basadas en una columna de fecha

y hora de la base de datos. Puede utilizar el Asistente para dimensiones con los esquemas del Data warehouse: plano, de estrella y radial ramificado.

Editor de dimensiones

Podrá modificar estructuras de dimensiones compartidas existentes y crear otras nuevas mediante sencillas operaciones de arrastrar y colocar. El editor de dimensiones es un complemento del Asistente para dimensiones que le permite revisar las dimensiones que haya creado con el asistente o crear otras nuevas con gran rapidez. En el editor, también podrá obtener una vista previa de los datos de dimensiones.

Asistente para actualización incremental

Podrá utilizar este asistente que le guiará por el proceso de incorporar nuevos datos al cubo. Una actualización incremental agrega nuevos datos a un cubo sin la necesidad de reconstruir los agregados y volver a cargar todos los datos.

Asistente para particiones

Este asistente le ayuda a crear una nueva partición que contendrá una parte de los datos del cubo. Las particiones le permiten distribuir y optimizar los datos de un cubo en segmentos discretos en un único servidor o entre varios servidores.
Nota Sólo podrá disponer de particiones definidas por el usuario si instala la edición Enterprise de los Servicios OLAP de Microsoft SQL Server.

Asistente para almacenamiento y agregado de datos

Puede utilizar este asistente para especificar el modo de almacenamiento que desee utilizar con los datos del cubo y para ayudarle a diseñar los agregados apropiados para el uso deseado del cubo. Las opciones disponibles en este asistente le permitirán encontrar el equilibrio óptimo entre el tiempo de respuesta y los requisitos de almacenamiento teniendo en cuenta las necesidades de la aplicación y de los usuarios.

Asistente para análisis de uso

El Asistente para análisis de uso le ayudará a comprender cómo se está utilizando un cubo al mostrarle en forma de tablas y gráficos la información registrada acerca de las consultas, como fecha, usuario, tiempo de respuesta de la consulta y frecuencia.

Asistente para optimización basada en el uso

Puede utilizar el Asistente para optimización basada en el uso para ayudarle a afinar el rendimiento del cubo según el empleo del mismo por parte de los usuarios. Puede utilizar el asistente para crear agregados que mejoren el rendimiento en cuanto a cualquier combinación de usuarios, el número de veces que se ha ejecutado una consulta, el tiempo de respuesta a las consultas, el tipo de almacén donde residen los datos o un intervalo de fechas.

Asistente para cubo virtual

Puede utilizar este asistente para combinar cubos y seleccionar dimensiones y medidas de los mismos para crear un cubo virtual. Un cubo virtual permite enrutar una misma consulta a varios cubos, incluidos cubos que se ejecuten en servidores diferentes. Un cubo virtual se presenta ante los usuarios como si fuese un cubo real, pero no necesita un espacio de almacenamiento adicional; es similar a una vista de una base de datos relacional que combina tablas.

Asistente para dimensiones virtuales

Con el Asistente para dimensiones virtuales podrá crear fácil y rápidamente dimensiones virtuales compartidas. Puede seleccionar propiedades de miembros definidas para dimensiones compartidas para crear una dimensión virtual y, después, utilizar la dimensión virtual en un cubo que también utilice la dimensión a la que pertenecían las propiedades de miembro seleccionadas.

Examinar vistas de datos

Puede ver datos del cubo sin tener que salir del Administrador de OLAP. No necesita cambiar a otra aplicación para comprobar los cubos que haya diseñado.

Integración con el Ubicador de orígenes de datos para OLE DB

Los Servicios OLAP utilizan el Ubicador de orígenes de datos de Microsoft para seleccionar orígenes de datos OLE DB u ODBC.

MODELO DE DATOS FLEXIBLE

Los servicios OLAP de Microsoft SQL Server son compatibles con varios modelos de datos y de almacenamiento para ayudarle a crear y mantener un sistema OLAP que responda a las necesidades de su organización.

Opciones del almacenamiento múltiple de datos

Los Servicios OLAP ofrecen tres modos distintos de almacenamiento para el sistema:

- MOLAP (OLAP multidimensional)
  Los datos subyacentes de un cubo se almacenan con los datos de agregados en una estructura multidimensional de alto rendimiento. El sistema de almacenamiento MOLAP proporciona un rendimiento y compresión de datos excelentes.
- ROLAP (OLAP relacional)
  Los datos subyacentes de un cubo se almacenan en una base de datos relacional con los datos de agregados. El sistema de almacenamiento ROLAP le permitirá sacar el máximo partido de la inversión que ha realizado en tecnología relacional y en herramientas de administración de datos empresariales.
- HOLAP (OLAP híbrido)
  Los datos subyacentes de un cubo se almacenan en una base de datos relacional y los datos de agregados se almacenan en una estructura multidimensional de alto

rendimiento. El sistema de almacenamiento HOLAP ofrece las ventajas de MOLAP para los agregados sin necesidad de duplicar los datos de detalle subyacentes.
Los cubos virtuales y particiones son otras formas de OLAP híbrido que le permitirán personalizar las alternativas de almacenamiento de los cubos para adaptarlas a sus necesidades.

Almacenamiento de un cubo con particiones

Un cubo se puede dividir en secciones físicas independientes. Cada partición se puede almacenar en un modo diferente, en una ubicación física distinta y con un nivel de agregados apropiado para los datos contenidos en la partición. El resultado es que podrá ajustar el rendimiento y las características de administración de datos del sistema.

Nota Sólo podrá disponer de particiones definidas por el usuario si instala la edición Enterprise de los Servicios OLAP de Microsoft SQL Server.

Mezclar particiones

Podrá volver a combinar varias particiones de un cubo en una única partición física. Por ejemplo, puede mezclar particiones para consolidar porciones de datos del cubo; por ejemplo, puede mezclar la información correspondiente a un trimestre que acaba de terminar con una única partición relativa al año.

Nota Sólo podrá disponer de particiones definidas por el usuario si instala la edición Enterprise de los Servicios OLAP de Microsoft SQL Server.

Cubos modificables

Un cubo puede habilitarse para el acceso de escritura simultáneo de varios usuarios. Los cambios iniciados por un usuario en los datos del cubo se registran en una tabla de partición especial, físicamente independiente, asociada con el cubo y se aplican automáticamente a medida que se presentan los datos del cubo. Ante el usuario parece como si los datos del cubo hubieran cambiado. Los cambios se pueden descartar o hacer de sólo lectura a criterio del administrador de la base de datos.[5]

Cubos virtuales

Podrá combinar cubos en cubos virtuales, de forma muy similar a como se pueden combinar tablas y vistas en una base de datos relacional. Un cubo virtual proporciona acceso a los datos almacenados en los cubos combinados sin requerir la construcción de un nuevo cubo, al mismo tiempo que permiten mantener el mejor diseño para cada uno de los cubos individuales.

[5] DBA, *Database Administrator*

Miembros calculados

Podrá crear medidas calculadas y miembros de dimensiones calculadas sin más que combinar expresiones multidimensionales MDX, fórmulas matemáticas y funciones definidas por el usuario (UDF, *User-Defined Function*). Con esta función, podrá definir nuevas medidas y miembros de dimensiones basados en una sintaxis de gran riqueza y fácil de utilizar. Podrá registrar bibliotecas adicionales de UDF para utilizarlas en definiciones de miembros calculados.

Propiedades de miembros

Se pueden definir propiedades para miembros de dimensiones y utilizar datos para estas propiedades en un cubo. Por ejemplo, si los miembros de la dimensión Producto son SKU, es probable que existan varias propiedades asociadas con SKU tales como tamaño, color, tejido, etc. Podrá especificar tales propiedades como propiedades de miembros y utilizarlas en consultas analíticas.

Dimensiones virtuales

Podrá crear dimensiones virtuales a partir de propiedades de miembros asociadas con dimensiones compartidas. Podrá utilizar una dimensión virtual para evaluar las propiedades de los miembros de una dimensión contra los propios miembros. Por ejemplo, podrá evaluar las medidas para SKU en función del tamaño, color, tejido, etc. Las dimensiones virtuales y las propiedades de miembros se evalúan como corresponda para ejecutar las consultas y no necesitan un espacio físico de almacenamiento en el cubo.

ESCALABILIDAD

Los servicios OLAP ofrecen una arquitectura OLAP escalable que se puede adaptar a muchos escenarios de almacenamiento de datos.

Opciones personalizadas de agregado

Con el Asistente para almacenamiento de datos, podrá optimizar el equilibrio entre el rendimiento del sistema y el espacio en disco asignado para almacenar agregados. Los Servicios OLAP de Microsoft SQL Server utilizan un sofisticado algoritmo para determinar el conjunto óptimo de agregados del que se pueden derivar otros agregados. En consecuencia, podrá centrarse en el diseño de aplicaciones y dejar que el sistema se ocupe de la compleja administración del diseño de agregados.

Optimización basada en el uso

El rendimiento de un cubo puede afinarse para que las consultas que se ejecuten con mayor frecuencia obtengan una respuesta rápida; para ello deberá utilizar el Asistente para optimización basada en el uso de forma que pueda diseñar agregados apropiados a dichas consultas al mismo tiempo que mantiene unos requisitos razonables de almacenamiento. Así, podrá generar con rapidez un sistema con un mínimo de agregados y, posteriormente, optimizar el rendimiento de acuerdo con el uso real del sistema.

Compresión de datos y optimización de almacenamiento

En los modos de almacenamiento MOLAP y HOLAP, los Servicios OLAP almacenan en estructuras multidimensionales toda o parte de la información del cubo. En estas estructuras, espacio de almacenamiento no se utiliza para celdas vacías, a la vez que se aplica un sofisticado algoritmo de compresión de datos a la información que está almacenada. Cuando se combinan con las opciones flexibles para el diseño y optimización de los agregados precalculados, estas técnicas ayudan a minimizar el impacto del "síndrome de explosión de datos" inherente a la tecnología OLAP.

Cálculo distribuido

El servicio PivotTable incorpora funciones de servidor que, con frecuencia, permiten realizar los cálculos en el cliente en lugar de hacerlo en el servidor. Esto distribuye la carga de cálculo entre el servidor y el cliente, lo que incrementa la capacidad del servidor, reduce el tráfico en la red y mejora el rendimiento de los clientes.

Particiones

Un cubo puede repartirse entre varios servidores al dividirlo en particiones. Los Servicios OLAP pueden recuperar datos en paralelo para responder a las consultas. La creación de particiones permite administrar la estrategia de almacenamiento, aumentar el alcance con varios servidores e incrementar el rendimiento.

Nota Sólo podrá disponer de particiones definidas por el usuario si instala la edición Enterprise de los Servicios OLAP de Microsoft SQL Server.

Actualizaciones incrementales

Podrá actualizar un cubo procesando únicamente los datos que se hayan agregado, en lugar de procesar todo el cubo; podrá ejecutar actualizaciones incrementales para actualizar cubos OLAP que se encuentren en uso.

LAN, WAN, Internet y escenarios móviles

La administración inteligente de la caché integra el servidor OLAP con el cliente del servicio PivotTable, que disminuye el tráfico a través de las conexiones de LAN y de WAN. El servicio PivotTable contiene un motor de cálculo multidimensional eficaz para disminuir el tráfico por la red y permitir el análisis de los datos locales multidimensionales cuando el cliente no se encuentre conectado al servidor.

Los controles ActiveX de Microsoft, las secuencias de comandos de Páginas Active Server (ASP) y las API de ActiveX Data Objects (ADO), constituyen varias soluciones para consultar datos de OLAP a través del Web.

Compatibilidad con Windows NT en la plataforma Alpha de DEC

Además de la plataforma Intel, todas las tecnologías de cliente y servidor de los Servicios OLAP son compatibles en Microsoft Windows NT para la plataforma Alpha de DEC.

Compatibilidad con aplicaciones clientes para Windows 95 y Windows 98

El servicio PivotTable se puede ejecutar en Microsoft Windows 95 y Windows 98 ya que es compatible con aplicaciones clientes disponibles para estas plataformas así como para Windows NT.

INTEGRACIÓN

Los Servicios OLAP funcionan de forma conjunta con otros componentes y programas para asegurar una solidez a nivel corporativo.

Consola de administración integrada

Los Servicios OLAP de Microsoft SQL Server incluyen una herramienta gráfica de administración denominada Administrador de OLAP. Se trata de un complemento de Microsoft Management Console (MMC). Como tal, este producto proporciona un marco de trabajo y una interfaz de usuario comunes para definir, tener acceso y administrar los servidores y bases de datos OLAP. En MMC podrá utilizar varios complementos; por ejemplo, puede instalar el Administrador corporativo de SQL Server y otros complementos junto con el Administrador de OLAP.

Seguridad integrada

El acceso a los cubos se basa en el sistema de seguridad de Microsoft Windows NT, que proporciona integración con SQL Server a través de las definiciones de usuarios y grupos de Windows NT.

Orígenes de datos OLE DB

Podrá utilizar una variedad de los orígenes de datos de OLE DB y ODBC para almacenes de datos, tales como Oracle versiones 7.3 y 8.0. Podrá utilizar varios orígenes de datos al mismo tiempo.

Caché del lado del servidor

Las consultas del usuario, metadatos y los datos se almacenan en la caché del servidor OLAP. Las definiciones y metadatos de consultas almacenados en la caché permiten responder a nuevas consultas al calcular las respuestas a partir de los datos almacenados en la caché en lugar de tener que recuperar los datos del disco.

Caché del lado del cliente

Las aplicaciones de cliente se conectan al Servidor OLAP mediante el componente de servicio PivotTable basado en el cliente. Debido a que el servicio PivotTable recibe metadatos con los datos devueltos del servidor como respuesta a una consulta, con frecuencia puede utilizar los datos almacenados en la caché del cliente para calcular la respuesta de posteriores consultas sin tener que enviar una nueva consulta al servidor. Por ejemplo, si la caché del cliente contiene valores para los cuatro trimestres de un año específico y el usuario solicita el total de los valores del año, el servicio PivotTable calcula la respuesta a partir de los datos almacenados en la caché.

El servicio PivotTable comparte gran parte de sus funciones con el servidor, lo que permite traer directamente al equipo cliente el motor de cálculo multidimensional, las características de caché y la administración de consultas del servidor. Este modelo de administración de datos cliente-servidor optimiza el rendimiento y minimiza el tráfico de la red.

API AMPLIAMENTE COMPATIBLES Y ARQUITECTURA ABIERTA

Los Servicios OLAP de Microsoft SQL Server ofrecen herramientas que le permiten extender su funcionalidad mediante programación.

OLE DB

Los servicios OLAP se diseñaron para cumplir los requisitos de la especificación OLE DB 2.0 y posteriores, que incluye determinados elementos específicos de OLAP. Esta especificación se ha desarrollado con la información aportada por más de 40 proveedores de clientes y servidores de OLAP, y es un estándar muy difundido para el acceso multidimensional a datos.

Funciones definidas por el usuario

Podrá ampliar la lista de funciones integradas sin más que crear bibliotecas de funciones que utilicen lenguajes de programación del Modelo de objetos componentes (COM, *Component Object Model*) tales como Microsoft Visual Basic o Microsoft Visual C++. Podrá registrar estas bibliotecas y utilizar las funciones en definiciones de miembros calculadas. Esta arquitectura permite agregar herramientas personalizadas de análisis a las aplicaciones de OLAP.

Objetos de ayuda para la toma de decisiones

El modelo de objetos de servidor, denominado Objetos de ayuda para la toma de decisiones (DSO, *Decision Support Objects*), puede utilizarse para crear aplicaciones que definan y administren cubos y otros objetos. Este modelo de objetos se puede utilizar para extender la funcionalidad del Administrador de OLAP o para automatizar el mantenimiento continuado del sistema.

Soporte para complementos

Podrá utilizar la interfaz del Administrador de complementos de OLAP para crear aplicaciones que extiendan la funcionalidad de la interfaz de usuario del Administrador de OLAP. Mediante la interfaz del Administrador de complementos de OLAP y DSO podrá crear extensiones personalizadas, cuadros de diálogo, asistentes y otras aplicaciones que se puedan integrar con el Administrador de OLAP.

ARQUITECTURA DEL CUBO OLAP

El objeto OLAP básico es el cubo, que consiste en una representación multidimensional de datos de detalle y resumen. Un cubo consta de un origen de datos, dimensiones, medidas y particiones. Los cubos se diseñan a partir de los requisitos de análisis de los usuarios. Un Data warehouse puede contener muchos cubos distintos, por ejemplo, el cubo Ventas, el cubo Inventario, etc.

El origen de datos del cubo identifica y se conecta con la base de datos donde se encuentra el Data warehouse, que es el origen de los datos del cubo.

Las dimensiones asignan la información de la tabla de dimensiones del Data warehouse a una jerarquía de niveles, por ejemplo, una dimensión Geográfica con los niveles Continente, País, Estado o provincia, y Ciudad.

Las dimensiones se pueden crear de forma independiente y se pueden compartir entre los cubos para facilitar la construcción éstos y asegurar la consistencia del resumen de datos de análisis. Por ejemplo, si se utiliza una dimensión compartida para una jerarquía de productos en los cubos apropiados, la organización de la información resumida de los productos será consistente entre los distintos cubos que utilicen esa dimensión.

Una dimensión virtual es un tipo especial de dimensión que asigna las propiedades de los miembros de otra dimensión a una dimensión que, a partir de ese momento, se puede utilizar en cubos.

Por ejemplo, una dimensión virtual de la propiedad tamaño de un producto permite que un cubo resuma datos como la cantidad de ventas por producto y por tamaño, por ejemplo, la cantidad de camisetas que se han vendido por estilo y tamaño. Las dimensiones virtuales y las propiedades de los miembros se evalúan como corresponda para ejecutar las consultas y no necesitan un espacio físico de almacenamiento en el cubo.

Las medidas identifican los valores numéricos extraídos de la tabla de hechos que están resumidas para realizar el análisis de precio, costo o cantidad vendida.

Las particiones son los contenedores multidimensionales de almacenamiento que guardan los datos del cubo. Cada cubo contiene, al menos, una partición y los datos de un cubo se pueden combinar a partir de varias particiones. Cada partición puede obtener sus datos de un origen de datos distinto y se pueden almacenar en una ubicación diferente.

Los datos de una partición se pueden actualizar independientemente de los datos contenidos en otras particiones del mismo cubo. Por ejemplo, los datos de un cubo se pueden dividir mediante criterios de tiempo, con una partición para los datos del año actual, otra partición para los datos del año anterior y una tercera partición para los datos más antiguos.

Las particiones de un cubo se pueden almacenar separadas en distintos modos de almacenamiento con diferentes grados de resumen. Las particiones son invisibles para el usuario, para quien el cubo aparece como un único objeto, pero siguen proporcionando al administrador una amplia variedad de opciones para administrar los datos de OLAP subyacentes.

Nota Las particiones definidas por el usuario sólo estarán disponibles si tiene la edición Enterprise de los Servicios OLAP de Microsoft SQL Server.

Las funciones permiten administrar el acceso de los usuarios a los datos del cubo al asignar las cuentas de grupos y de usuarios de Microsoft Windows NT a privilegios de acceso al cubo.

Un cubo virtual es una vista lógica de porciones de uno o más cubos. Un cubo virtual puede utilizarse para combinar cubos relativamente distintos que comparten una dimensión común, por ejemplo un cubo Ventas y un cubo Almacén, con propósitos de análisis, al mismo tiempo que se conserva la funcionalidad propia de cada cubo. Se pueden seleccionar dimensiones y medidas de los cubos combinados para presentarlas en el cubo virtual.

ARQUITECTURA DEL SERVIDOR

Los servicios OLAP de Microsoft SQL Server ofrecen capacidades de servidor para crear y administrar datos de OLAP multidimensionales y para suministrar datos a clientes mediante el servicio PivotTable. Entre las operaciones del servidor se incluyen la creación de cubos de datos multidimensionales a partir de bases de datos y almacenes de datos relacionales, y el almacenamiento de cubos en estructuras multidimensionales de cubos, en bases de datos relacionales o en una combinación de ambas. Los metadatos de las estructuras multidimensionales de cubos se almacenan en un depósito de una base de datos relacional.

El complemento Administrador de OLAP proporciona una interfaz de usuario que se ejecuta como un complemento en Microsoft Management Console (MMC). Las interfaces de programación se proporcionan para permitir que las aplicaciones personalizadas interactúen con el modelo de objetos que controla el servidor y con el Administrador de OLAP.

Arquitectura del sistema de los servicios OLAP (servidor)

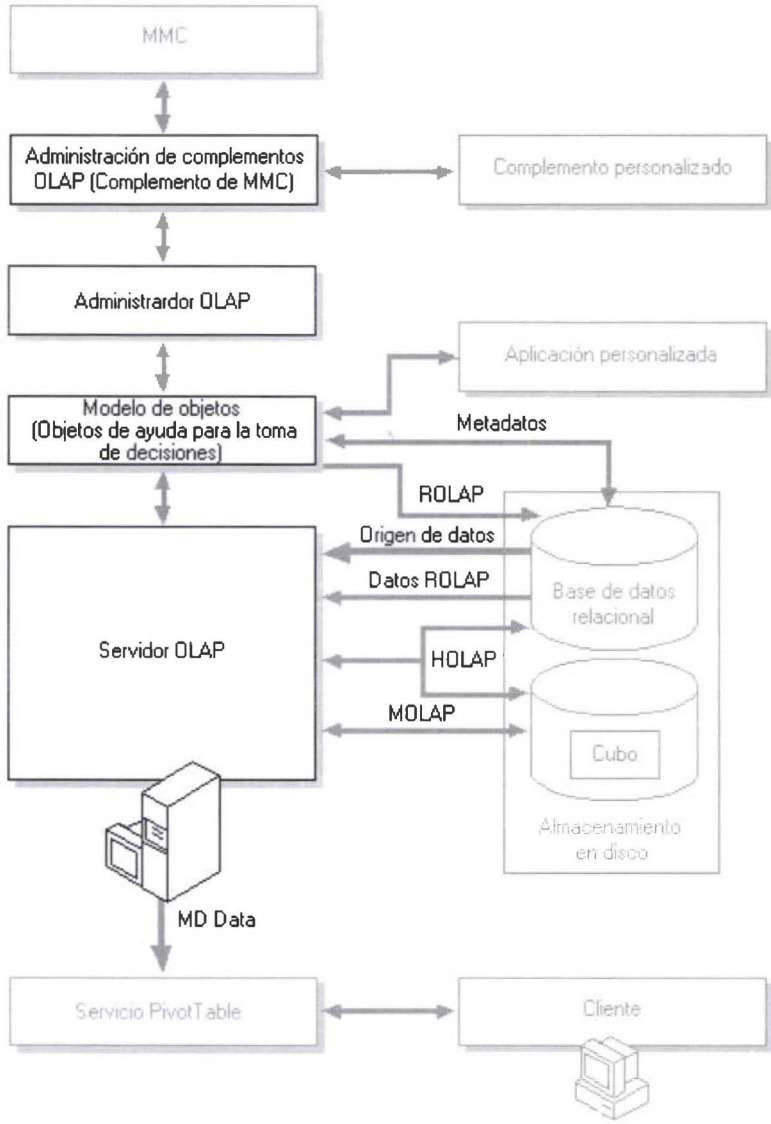


Gráfico 41.

ARQUITECTURA DEL CLIENTE

El servicio PivotTable se comunica con el servidor OLAP y proporciona interfaces que las aplicaciones de cliente utilizan para tener acceso a los datos OLAP almacenados en el servidor. Las aplicaciones de cliente se conectan con el servicio PivotTable mediante interfaces OLE DB para C++ o mediante el modelo de objetos de Microsoft ActiveX Data Objects (ADO) para lenguajes de automatización del Modelo de objetos componentes (COM, *Component Object Model*), como Microsoft Visual Basic.

Arquitectura del servicio PivotTable

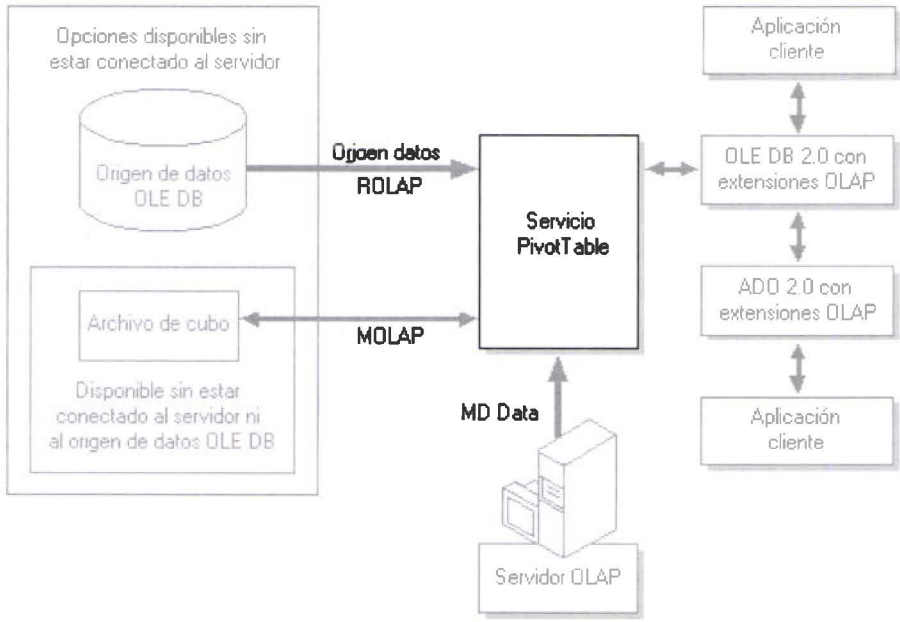


Gráfico 42.

Los servicios PivotTable también pueden crear archivos de cubos locales que contienen datos de un cubo del servidor o de las bases de datos relacionales OLE DB. Los cubos locales se pueden almacenar como archivos de cubos multidimensionales en el equipo cliente. Los cubos locales se pueden utilizar sin conexión gracias al servicio PivotTable para análisis portátiles.

3. DETERMINAR LOS REQUISITOS DEL NEGOCIO, DEL USUARIO Y TECNICOS

Para un proyecto de Data warehouse se deben analizar las herramientas de las que se disponen o se van a disponer para su diseño e implementación, pero antes de esto y antes de poder generar un Data warehouse, se debe escribir un proyecto y un plan de implementación detallados.

El plan de proyecto e implementación podría incluir:

REQUISITOS DEL NEGOCIO Y DE USUARIO

Un proyecto siempre tiene inmerso un caso práctico de negocio que se intenta resolver con la generación de dicho proyecto, este representa costos, pero también existe un retorno de la inversión. Por lo tanto he aquí nuestro caso.

Se tiene una organización establecida en dos grandes ciudades del país como son Quito y Guayaquil con grandes expectativas de crecimiento, que brinda servicios transaccionales financieros y no financieros a instituciones bancarias y de servicios. Por tanto, su principal actividad está representada por las transacciones realizadas por ventanilla.

Una empresa de servicios transaccionales, por su naturaleza misma, maneja grandes cantidades de información transaccional almacenada en la base de datos operacional.

Esta información al no estar disponible, ni agrupada por los criterios necesarios para su análisis y la toma de decisiones sobre la base de lo que refleja y las proyecciones que se pueden realizar a partir de lo conseguido hasta ese momento, dificulta la rápida toma de acciones de las áreas estratégicas de la empresa.

Lo mencionado anteriormente nos lleva a determinar un proyecto de desarrollo de Data warehouse que nos permita satisfacer los requerimientos de las gerencias general, operativa y de procesos.

La recolección de los requisitos del usuario requiere, en gran medida, llevar a cabo entrevistas a los usuarios del Data warehouse. Los requisitos del usuario nos ayudan a determinar que:

- El nivel de granularidad de los datos del data warehouse se mantendrá en el orden de los datos operativos históricos y los datos agregados.
- Los sistemas operacionales dentro de la empresa que contienen los datos, específicamente en este caso de estudio, será uno que es el sistema transaccional central.
- Las reglas de negocio que deben cumplir los datos, serán precisadas a lo largo del diseño del data warehouse.
- Las consultas necesarias para proporcionar los datos a los usuarios, están dadas en su mayoría por el detalle de los requerimientos de usuario que se muestra a continuación.

El proyecto a desarrollar deberá entregar la siguiente información:

- Número de transacciones realizadas por ciudad
- Número de transacciones realizadas por agencia
- Número de transacciones realizadas por cajero
- Número de transacciones por su naturaleza (financieras y no financieras)
- Número de transacciones por institución
- Numero de transacciones por tipo
- Numero de transacciones por hora
- Número de transacciones por tipo de caja
- Número de transacciones reversadas
- Número de transacciones por motivo de reverso
- Sobrantes y faltantes por cajero
- Exceso monto por agencia
- Exceso monto por cajero
- Manejo de efectivo por institución y agencia

COSTO DEL PROYECTO

Para el desarrollo del data warehouse que cumpla con los requerimientos antes mencionados se tienen los siguientes costos:

Tipo de recursos	Descripción	Costo	Tiempo de duración
Recurso Humano	2 personas	$1800	3 meses
Recursos Materiales e insumos	Tapes, diskettes, cds, papelería	$300	3 meses
Recursos Tecnológicos Software	Servidor: Motor de BDD – Microsoft SQL Server 7.0 Sistema operativo- Windows NT 4.0 Cliente: Microsoft Office 2000 Microsoft Windows 95, 98 o Workstation 4.0	 $805 $680 $129 $161+$32	
Recursos Tecnológicos Hardware	Servidor: 1 Servidor en donde resida el data warehouse. Cliente: 1 estación cliente para pruebas	$3000 $1200	
	TOTAL	**$8107**	

Cuadro de costo del proyecto

Costos actuales

Actualmente todos los reportes y cuadros estadísticos los realiza una persona, dedicada a tiempo completo, con el soporte del administrador de la base de datos transaccional, mismo que le proporciona la información que requiere, claro esta con las limitaciones del caso, como son: demora en la generación de la información y en formatos de difícil entendimiento.

Por lo cual se tiene que el gasto mensual, por el uso de estos recursos es:

Recurso	Costo /mensual
Administrador de sistema transaccional	$200
Persona encargada de labor manual	$400
TOTAL	**$600**

Cuadro de gastos mensuales

Por tanto, se tendría una recuperación de la inversión del proyecto, en el lapso **de tiempo de un año,** al ahorrar el costo de los recursos humano que se esta usando este momento.

REQUISITOS TÉCNICOS

En cuanto a los requisitos técnicos se han determinado los siguientes:

- La arquitectura hardware e infraestructura
  La empresa mantiene una administración centralizada en la ciudad de Quito y un sistema operacional centralizado en la misma ciudad, por tanto se contará un solo Data warehouse en la misma ubicación que el sistema operacional.

  Se debe contar con un servidor dedicado para la residencia del Data warehouse con las siguientes características:

  - Servidor Compaq Proliant 1600
  - 512 MB de memoria RAM
  - 3 discos de 9.1 Gb
  - Unidad de tape backup

- Mecanismos de copias de seguridad y recuperación.
  Se mantendrá un sistema de respaldos diarios antes y después de las actualizaciones diarias de la central de datos. Para ello se contará con unidades de tape que mantendrán la información respaldada durante un periodo de 15 días.

- Directrices de seguridad.
  Las políticas de seguridad a cumplir estarán dentro del marco implementado para todo el departamento de sistemas, específicamente sujetas a los estándares

establecidos en el modelo de producción vigente para el manejo de información dentro del departamento.

- Métodos de carga y transformación de datos desde los sistemas operacionales al Data warehouse
  Para la carga inicial se definirá un proceso manual, con la finalidad de ir verificando paso a paso el proceso como tal, las posteriores cargas serán actualizaciones al Data warehouse, este proceso se seguirá durante el primer mes. Luego de lo cual, si no se presentan errores de carga o lógica del proceso, se dará por estabilizado el proceso de carga diaria.

Analizando las herramientas cliente con las que se cuenta en este momento, módulos para consultas típicas de un sistema transaccional, no proporcionan ninguna capacidad que ayude en el proceso de datos. Por otro lado se cuenta con Microsoft Office 97 como herramienta para presentación de reportes, que no presenta las capacidades necesarias para un análisis de la información.

En el siguiente capítulo se indicará a detalle como generar y diseñar la base de datos para el data warehouse, poniendo énfasis en el análisis de las herramientas de Microsoft SQL Server 7.0.

Así tenemos que Microsoft SQL Server proporciona varias herramientas para generar Data warehouse, Data marts, y sistemas OLAP. Al utilizar el Diseñador DTS, puede definir los pasos, el flujo de trabajo y las transformaciones necesarias para generar un Data warehouse a partir de varios orígenes de datos. Una vez generado el Data warehouse, puede utilizar los Servicios OLAP de Microsoft SQL Server, que proporcionan un eficaz servidor de OLAP que se puede usar para analizar los datos almacenados en muchos formatos diferentes, incluidas las bases datos de Oracle y de SQL Server.

4. DISEÑAR Y GENERAR LA BASE DE DATOS

El diseño y la generación de la base de datos es una parte crítica del proceso de generación de un Data warehouse. A menudo, este paso lo llevan a cabo diseñadores de bases de datos experimentados porque puede significar la recolección de datos de varios (y, en ocasiones, heterogéneos) orígenes y combinarlos en un modelo lógico único.

A diferencia de los sistemas OLTP que almacenan los datos extremadamente normalizados, los datos del Data warehouse se almacenan casi sin normalizar para mejorar el rendimiento de las consultas.

Los componentes de diseño del esquema son las dimensiones, las claves y las tablas de hechos y de dimensiones.

TABLAS DE HECHOS

Contienen datos que describen un suceso específico del negocio, como una transacción bancaria o el cobro de un servicio. Por otra parte, las tablas de hechos también contienen agregados de datos, como las transacciones realizadas por mes y por ciudad. Los datos de una tabla de hechos no se actualizan; simplemente, se agregan datos nuevos.

Debido a que las tablas de hechos contienen la mayor parte de los datos almacenados en el Data warehouse, es importante que antes de cargar los datos, la estructura de la tabla sea correcta. Puede ser necesaria una costosa reestructuración de la tabla si los datos requeridos por las consultas se pierden o son incorrectos.

Las características de las tablas de hechos son:

- Muchas filas.
- Principalmente datos numéricos; raramente caracteres.
- Múltiples claves externas (en las tablas dimensionales).
- Datos estáticos.

TABLAS DE DIMENSIONES

Contienen datos que se utilizan para hacer referencia a los datos almacenados en la tabla de hechos, como descripciones de instituciones, transacciones, y agencias. La separación de esta información detallada de los sucesos específicos, como el valor de una transacción en un momento determinado, hace posible optimizar las consultas al reducir el conjunto de datos que se debe recorrer en la tabla de hechos.

Las características de las tablas dimensionales son:

- Menos filas que las tablas de hechos.
- Contienen, principalmente, datos de tipo carácter.
- Varias columnas que se utilizan para administrar las jerarquías de dimensiones.
- Una clave principal (la clave dimensional).
- Datos actualizables.

DIMENSIONES

Son categorías de información que organizan los datos del almacén, como el tiempo, geografía, organización, etc. Normalmente, las dimensiones son jerárquicas en el sentido de que un miembro puede ser un elemento secundario de otro miembro. Para admitir esto, la tabla de dimensiones debería incluir la relación de cada miembro con las capas superiores de la jerarquía.

CLAVES DIMENSIONALES

Son identificadores exclusivos que se utilizan para consultar los datos almacenados en la tabla de hechos central. La clave dimensional, al igual que una clave principal, enlaza una fila en la tabla de hechos con una tabla de dimensiones. Esta estructura permite generar fácilmente consultas complejas. Una base de datos óptima de un Data warehouse contiene tablas de hechos largas y estrechas, y tablas dimensionales pequeñas y anchas.

TÉCNICAS DE DISEÑO

Antes de escoger una técnica de diseño para el Data warehouse, se debe analizar las características que presenta cada una de estas técnicas:

Esquema de estrella

La técnica de diseño más popular utilizada para implementar un Data warehouse es el esquema de estrella. La estructura de este tipo de esquema aprovecha las consultas típicas de ayuda a la toma de decisiones al utilizar una tabla de hechos central acerca del área en estudio y muchas tablas de dimensiones que contienen descripciones no normalizadas de los hechos. Una vez creada la tabla de hechos, las herramientas OLAP pueden usarse para agregar previamente información a la que se tiene acceso frecuentemente.

El diseño del esquema de estrella ayuda a aumentar el rendimiento de las consultas al reducir el volumen de los datos que tienen que leerse del disco. Las consultas analizan los datos en las tablas de dimensiones más pequeñas para obtener las claves de dimensión que indizan la tabla de hechos central, reduciendo el numero de filas que se deben recorrer.

Esquema de copo de nieve

El esquema de copo de nieve es una variación del esquema de estrella en donde las tablas dimensionales se almacenan en una forma más normalizada. Esto puede ayudar a mejorar el rendimiento de las consultas al reducir el número de lecturas en disco.

Luego de haber analizado las dos técnicas para el diseño del esquema de la base de datos para el Data warehouse, para un sistema transaccional de servicios financieros y no financieros, es el esquema en Estrella el que escogeremos, por los siguientes puntos:

- Por ser la más popular
- Porque ayuda al aumento del rendimiento de las consultas
- Porque las herramientas OLAP trabajan bien con este esquema.

CREACIÓN DE UN ESQUEMA DE BASE DE DATOS

El esquema de la base de datos debería admitir los requisitos del negocio en lugar de los requisitos orientados a consultas normales de un diseño de base de datos OLTP.

Para este proyecto, en el **Apéndice 1** se presenta la parte central del esquema de la base de datos del sistema de transaccional de la organización.

Los pasos que se indican a continuación nos llevarán a convertir un esquema OLTP en un esquema de estrella para el data warehouse:

1. Determinación de las tablas de hechos y dimensionales

Es importante determinar correctamente qué tablas y datos existentes en los sistemas operacionales deberían comprender las tablas de hechos y las dimensionales. Por ello lo que principalmente se debe tener en cuenta es:

- Las transacciones de negocio fundamentales en las que se centrará el Data warehouse (tablas de hechos).
- Los datos asociados a las transacciones de negocio que determinan cómo serán analizados los datos del negocio (tablas dimensionales y jerarquías).

Siguiendo estas recomendaciones se puede determinar que las tablas de hechos, para un sistema transaccional de servicios financieros y no financieros, son las siguientes:

Datos	Tabla
Compensación diaria por institución	dat_compensacion_diaria
Registro de totales por institución, por moneda y por cajero	dat_total_veh_legal_moneda_his
Movimiento de faltantes y sobrantes	dat_movimiento_fs
Transacciones con su respuesta	log_cabecera_general_his
Cabecera de transacciones financieras	log_cabecera_his
Detalle de transacciones financieras	log_detalle_his
Detalle de transacciones no financieras	log_detalle_trannofi_his
Excesos diarios en montos de agencias	log_exceso_monto_agencia_his
Excesos diarios en montos de cajeros	log_exceso_monto_cajero_his
Registro de sobrantes y faltantes de cajeros	log_sobrante_faltante
Tramas financieras procesadas	log_trama_financiera_his
Tramas no financieras procesadas	log_trama_nofinanciera_his
Cabecera de transacciones no financieras	log_tran_nofi_his
Detalle de transacciones de consultas	log_transaccion_consulta_his

Cuadro de tablas de hechos

Identificación de las tablas dimensionales

El paso siguiente implica la identificación de las entidades que describen cómo se analizarán los datos de los hechos. Sin embargo, los datos dimensionales elegidos deberían representar el núcleo del análisis del negocio. Como ejemplo, el análisis de negocio que se realiza en el Data warehouse de sistemas de servicios transaccionales financieros variaciones de:

- Depósitos en sucres por ciudad.
- Depósitos en sucres por periodo (por ejemplo, un trimestre).
- Todos los depósitos por ciudad.
- Todos los depósitos por periodo.

Por consiguiente, las tablas dimensionales incluirán datos de ciudades, monedas y datos de periodos. Desde el esquema OLTP para servicios transaccionales financieros, todos los datos de hechos y dimensionales pueden encontrarse en las siguientes tablas.

Datos	Tabla
Agencias	ref_agencia
Ciudades	ref_ciudad
Días de la semana	ref_dia_semana
Errores	ref_error
Horario de agencias	ref_horario_agencia
Monedas	ref_moneda
Motivos de transacciones	ref_motivo
Motivos de reversos de transacciones	ref_motivo_reverso
Productos	ref_producto
Subtransacciones de una transacción	ref_subtransaccion
Tipo de caja	ref_tipo_caja
Tipo de movimiento de faltantes y sobrantes	ref_tipo_movimiento_fs
Tipo de transacciones	ref_tipo_transaccion
Transacción para las instituciones	ref_tran_as400
Transacción plataforma	ref_tran_plat
Motivo de transacciones	ref_tranas_motivo
Transacciones de consulta	ref_trancons
Instituciones financieras y no financieras	ref_veh_legal

Cuadro de tablas de dimensiones

2. Diseño de tablas de hechos

El objetivo principal cuando se diseñan tablas de hechos es minimizar el tamaño sin comprometer los requisitos de los datos. Las tablas de hechos son las tablas más grandes de la base de datos porque contienen los datos de los detalles que representan las transacciones de negocio. Sin embargo, se debe considerar el costo de almacenar y mantener estas extensas tablas. Por ejemplo, cuanto más grande es una tabla, mayor almacenamiento necesita con conexión, y, en potencia, sin conexión; lleva más tiempo hacer una copia de seguridad y restaurarla en caso de error del sistema, y también consultarla cuando se generan los agregados OLAP.

Las formas más sencillas de reducir el tamaño de las tablas de hechos incluyen:

- Reducir el número de columnas, cualquier columna que no sea necesaria para analizar las operaciones del negocio debe ser quitada.
- Reducir el tamaño de cada columna donde sea posible, todos los caracteres y datos binarios son de longitud variable, donde sea posible se debe utilizar tipos de datos que necesiten menos bytes para almacenamiento.
- Archivar datos históricos en tablas de hechos separadas, si los datos de las tablas de hechos se usan poco.

3. Diseño de las tablas dimensionales

El objetivo principal del diseño de tablas dimensionales es eliminar la normalización de los datos que hacen referencia a las tablas de hechos a través de tablas simples. Cuando sea necesario, los datos dimensionales más utilizados deberían hacer referencia directamente a las tablas, más que hacerlo de manera indirecta a través de otras tablas. Este enfoque minimiza el número de combinaciones de tablas e incrementa el rendimiento.

Información de fecha y hora

La información acerca de fechas es un requisito común en un Data warehouse. Para minimizar el ancho de la tabla de hechos, en muchas ocasiones se crea en dicha tabla una clave externa que hace referencia a una tabla de dimensiones que contiene una representación de la fecha y/o la hora. La representación de la fecha depende de los requisitos de análisis del negocio.

Nota: Cuando se diseñan tablas de dimensiones para ser utilizadas con los Servicios OLAP de Microsoft SQL Server, sólo se necesita una fecha. El Asistente para tiempo de los Servicios OLAP permite resumir las fechas en una combinación de semanas, meses, trimestres y años. Con esto no sería necesario.

Siguiendo los tres pasos indicados anteriormente el esquema de la base de datos obtenido se encuentra adjunto en el **Apéndice 2**.

IMPLEMENTAR EL DISEÑO DE LA BASE DE DATOS

Tras el diseño de las tablas de hechos y dimensionales, el paso final es implementar físicamente las bases de datos en Microsoft SQL Server. Para ello tenemos los siguientes pasos:

1. Crear la base de datos

Antes de crear la base de datos debemos indicar los siguientes requisitos:

- Tener un servidor con sistema operativo Windows NT Server 4.0
- Instalar los prerrequisitos para la instalación de Microsoft SQL Server 7.0 que son:
  - Services Pack 4.0
  - Microsoft Internet Explorer 4.01 con Service Pack 1.0
- Instalar Componentes de Microsoft SQL Server 7.0
- Espacio libre en disco 4 GB

Para crear la base de datos ingresamos por **Inicio**, **Programas**, **Microsoft SQL Server 7.0**, **Enterprise Manager.**

A continuación aparecerá una ventana similar a la que se muestra en el **Gráfico 3**.

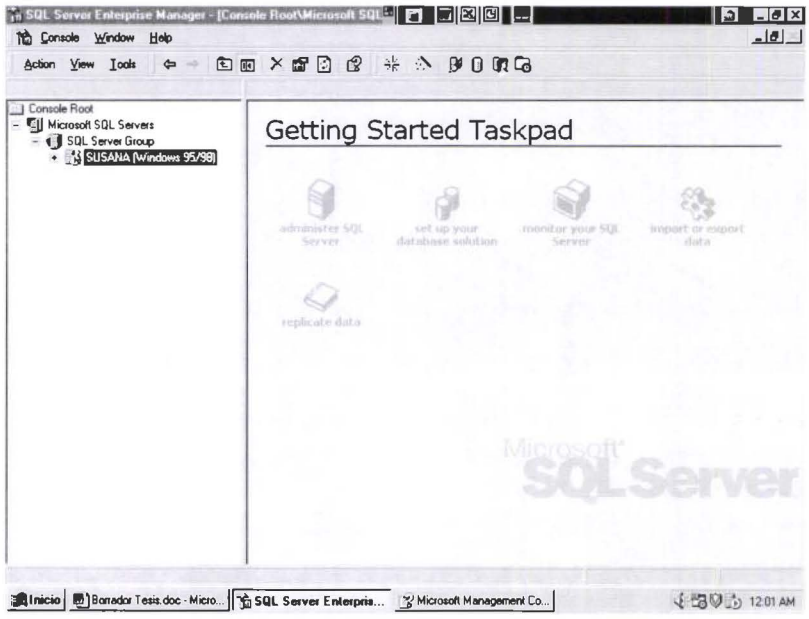


Gráfico 3.

Expandir el **SQL Server Group**, entonces expandir el **Servidor**, en el grupo **Databases** dar un clic con el botón derecho del mouse y escoger **New Database**.
Luego aparecerá una ventana que solicitará la siguiente información para la pestaña **General**:

Nombre de la base de datos:	dawaho
Nombre del archivo en el que se guardaran los datos:	dawaho_Data
Ubicación del archivo:	c:\bdd\dawaho_Data.mdf
Tamaño inicial de la base de datos:	500 MB

Luego escoger la pestaña **Transaction Log** con la siguiente información:

Nombre del archivo para el log de transacciones:	dawaho_Log
Ubicación del archivo:	c:\log\dawaho_Log.ldf
Tamaño inicial del log de transacciones:	100 MB

Los gráficos 4 y 5 ilustran la manera como aparecerá la información ingresada:

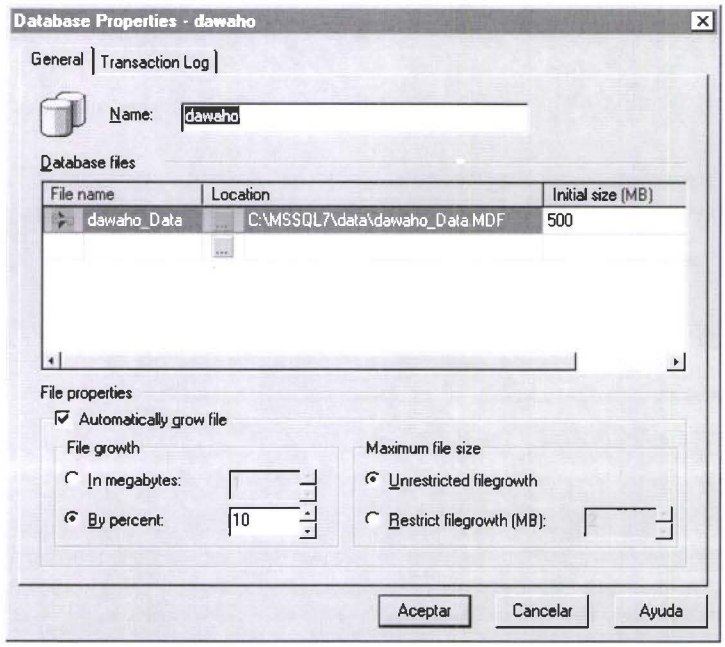


Gráfico 4.

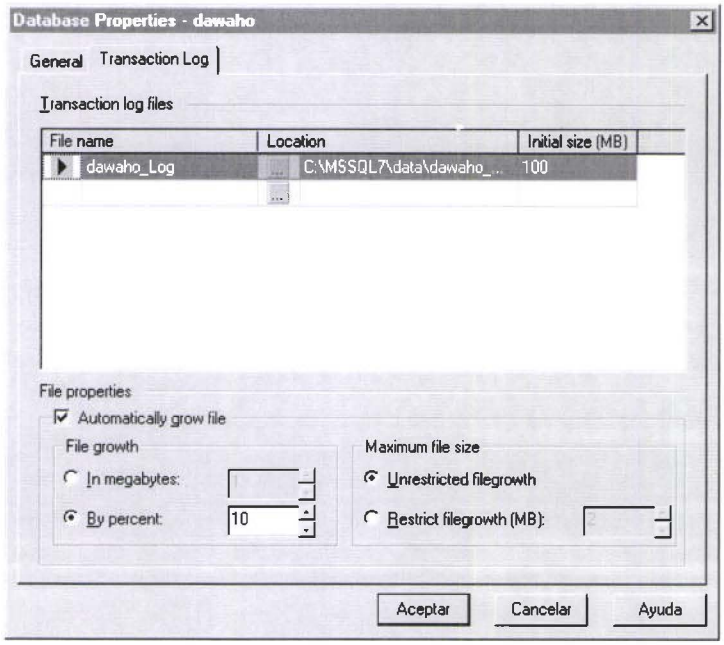


Gráfico 5.

2. Crear las tablas

Luego de creada la base de datos, se procederá a crear las tablas, para ello tenemos varias opciones para su creación como son:

- Mediante el Enterprise Manager de SQL Server
- Mediante el Query Analizer, usando sentencias SQL
- Desde la herramienta case usada para el modelamiento de la base de datos.

En esta oportunidad usaremos un archivo que contienen sentencias SQL, generado a partir del modelo generado por la herramienta case Logic Works ERwin_ERX 3.5, el mismo que se encuentra en el **Apéndice 3.**

Utilizamos la herramienta **Query Analizer** de Microsoft SQL Server 7.0**,** conectarse al servidor correspondiente con el usuario **sa** para tener los privilegios necesarios para la creación de objetos, luego abrimos el archivo **creartablas.sql,** este archivo contiene sentencia de T-SQL[6] y lo ejecutamos. Estos pasos se encuentran ilustrados en los siguientes gráficos 6 y 7.

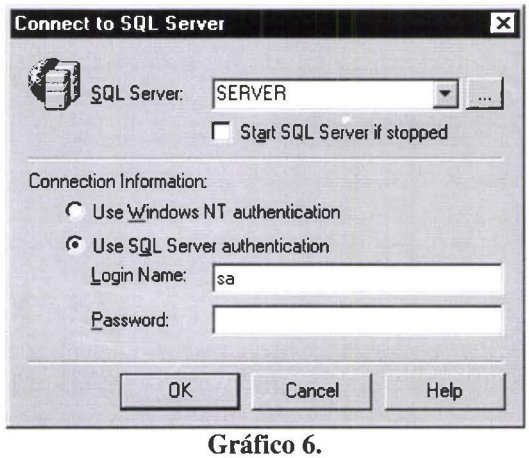


Gráfico 6.

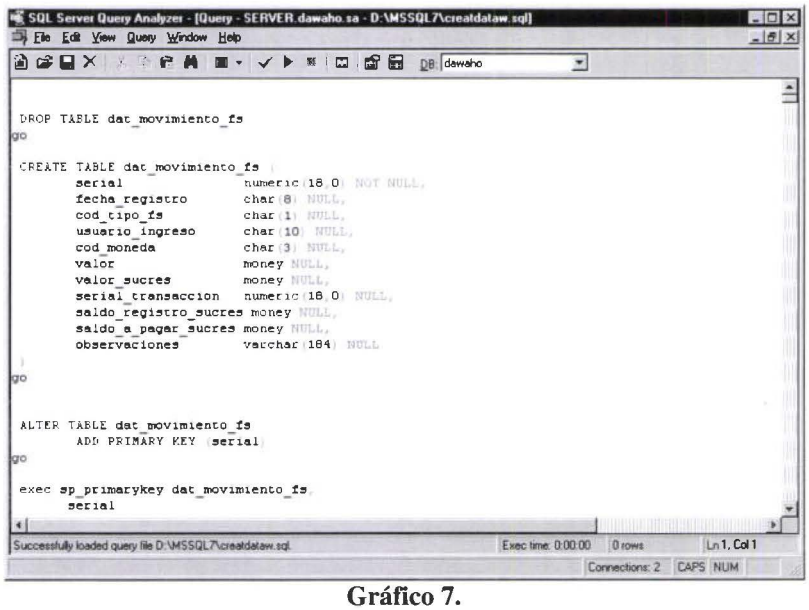


Gráfico 7.

[6] Lenguaje transaccional de consultas

Opcionalmente, se puede considerar la creación de tablas a través de las particiones disponibles de la base de datos, según el uso.

3. Crear algunas vistas definidas por el usuario

Si es necesario, se puede crear vistas definidas por el usuario. Las vistas de SQL Server podrían utilizarse para mezclar lógicamente las tablas con particiones horizontales, como interfaces para consultas predefinidas o como mecanismo de seguridad.[7]

Para este proyecto, para el sistema de transacciones financieras y no financieras, no se requiere crear ninguna vista.

4. Crear índices

Una vez que ya tenemos la base de datos con sus tablas, debemos generar índices para maximizar el rendimiento. Considerándolos en los siguientes casos:

- Columnas de claves.
- Columnas implicadas en combinaciones.
- Varias columnas, para aprovechar el alcance del índice.
- Todas las claves de tablas de dimensiones utilizadas por la tabla de hechos.

Como se puede observar en el **Apéndice 3**, se encuentran también la creación de los índices necesarios.

[7] Para más información consulte Libro guía del programador de Microsoft SQL Server

5. EXTRAER Y CARGAR DATOS

La extracción y la carga de datos desde sistemas operacionales a un Data warehouse varía en complejidad. El proceso puede ser muy simple si hay una correlación directa entre los datos de origen que deberían aparecer en el Data warehouse: por ejemplo, si todos los datos del origen de un único sistema operacional tienen el formato correcto y no tienen que ser modificados. El proceso puede ser también muy complejo: por ejemplo, si los datos del origen residen en varios sistemas operacionales y heterogéneos, y necesitan un cambio de formato y modificación importantes antes de ser cargados.[8]

Existen algunos criterios, en forma general, que se deberían tomar en cuenta para la selección de herramientas de extracción, refinamiento y migración de datos:[9]

- El soporte de archivos planos, archivos indexados y SMBD tradicionales es crítico, debido a que la mayor parte de los datos corporativos siguen manteniéndose en almacenamientos de este tipo.
- La capacidad de combinar datos de múltiples fuentes. Utilizar tecnologías de replicación de datos puede ayudar. La facilidad de captura de datos modificados, provee la funcionalidad de leer logs para obtener solo el dato cambiado para ser transferido. Esto reduce la cantidad de datos que deben ser cargados en el data warehouse para una actualización periódica.
- La interfase de especificación para indicar los datos que van a ser extraídos y los criterios de transformación.
- Es necesaria la habilidad para leer información de los diccionarios y repositorios de datos. Esto puede reducir el esfuerzo requerido para la especificación.
- El código generado por las herramientas debe ser mantenido completamente, desde el ambiente de desarrollo.
- La extracción selectiva de datos y registros permite al usuario extraer únicamente el dato deseado.
- Es necesario el reconocimiento de los datos a nivel de campo para la transformación de estos en información.
- La habilidad de realizar interpretaciones de tipos de datos y conjuntos de caracteres es necesario cuando se mueven datos entre sistemas incompatibles
- La capacidad de crear registros y campos de resumen, agregación y derivación
- El SMBD del data warehouse debe ser capaz de realizar la carga directamente desde la herramienta, utilizando la interfase de programación de aplicaciones nativa disponible en el SMBDR. Alternativamente, el sistema debería soportar la capacidad de crear archivos planos y luego utilizar la función de carga, proporcionada por el SMBDR.
- La estabilidad de los proveedores y el soporte para los productos son items que deben ser evaluados cuidadosamente.

Microsoft SQL Server 7.0, posee la herramienta para realizar el trabajo de extracción y carga de los datos observando los criterios anteriormente mencionados, esta herramienta es DTS (Servicio de transformación de datos).

[8] Tomado de Microsoft Books Online
[9] Metodología para una implementación exitosa de data warehousing, Piedad Valencia

Siguiendo con la metodología propuesta por Microsoft el proceso de extracción y carga implica:

VALIDAR LOS DATOS

Antes de extraer los datos de los sistemas operacionales, puede ser necesario asegurarse de que son completamente válidos. Si los datos no son válidos, la integridad del análisis de negocio que se basa en dichos datos puede verse comprometida. Por ejemplo, el valor que represente una transferencia de dinero entre bancos debe usar la moneda correcta.

Los datos deberían ser validados en el origen por los analistas del negocio puesto que son los que entienden qué es lo que representan. Cualquier cambio debería realizarse en los sistemas operacionales, en lugar de hacerlo en el Data warehouse, porque el origen de datos es incorrecto con independencia de su ubicación.

"La validación de los datos puede ser un proceso que implique mucho tiempo. El proceso de validación podría automatizarse si se escribieran procedimientos almacenados que comprobaran los datos para la integridad del dominio. Sin embargo, puede ser necesario validar los datos manualmente. Si descubre algún dato no válido, determine dónde se originó el error y corrija los procesos que contribuyan al mismo"[10]

Por ejemplo, los datos del sistema transaccional podrían validarse para asegurar que:

- La información de las transacciones (**Producto, Moneda**) representan a una combinación válida de producto y moneda.
- La información de agencia (**Ciudad, Agencia**) representa a una relación válida.

Esta información, si fuera el caso, debería validarse mediante los asistentes para importación y exportación con los Servicios de transformación de datos, cuando se copian los datos desde el origen al destino, puede determinar si la información acerca de los productos para las transacciones y las agencias son válidos. Cualquier dato no válido se puede guardar en el registro de excepciones para ser analizado posteriormente por los analistas del negocio y determinar por qué es incorrecto.

En nuestro proyecto, específicamente, al realizar el análisis de los posibles casos en los que se requeriría validar relaciones o combinaciones de datos se concluyo que no es necesario formular este proceso de validaciones, dado que este proceso se encuentra implementado en el sistema transaccional en el momento mismo de procesar una transacción.

MIGRAR LOS DATOS

La migración de datos desde sistemas operacionales normalmente implica copiar los datos a una base de datos intermedia antes de ser copiados finalmente al Data warehouse. Es necesario copiar los datos a una base de datos intermedia si tienen que ser normalizados.

---

[10] Tomado de Microsoft Technet- Marzo1999

"La copia de los datos debería hacerse durante un periodo de baja actividad del sistema operacional. De otra forma, el rendimiento del sistema podría degradarse y afectar a los usuarios. Además, si el Data warehouse se compone de datos de varios sistemas operacionales interrelacionados, es importante asegurar que la migración tenga lugar cuando los sistemas estén sincronizados. Si los sistemas operacionales no están sincronizados, las consultas a los datos del almacén pueden producir resultados inesperados"[11]

Para nuestro proyecto no será necesario una base de datos intermedia dado que la base de datos del sistema operacional es uno solo, no se requiere normalizar los datos, pero para otros proyectos si podría ser necesario por lo cual a continuación se indica que es la normalización de datos.

NORMALIZAR LOS DATOS

La normalización de los datos implica hacer que éstos sean coherentes. Es posible que los mismos datos estén representados de diferentes maneras en varios sistemas operacionales. Por ejemplo, un nombre de producto podría aparecer abreviado en un sistema operacional, pero no en otro. Si no se hicieran coherentes estos dos valores, cualquier consulta que utilizara los datos, probablemente, podría evaluarlos como productos diferentes. Si los datos detallados del Data warehouse se utilizan para producir información coherente, el nombre del producto debería ser coherente para todos los valores.

Puede conseguir la normalización de los datos:

- Mediante la utilización de los asistentes para importación y exportación con DTS para modificar los datos cuando se copian desde los sistemas operacionales a una base de datos intermedia, o directamente al Data warehouse.

- Al escribir una secuencia de comandos de Microsoft ActiveX, ejecutada por un programa que utilice la API DTS, para conectar con el origen de datos y normalizarlos. Puede realizarse en los datos cualquier manipulación que pueda llevarse a cabo mediante secuencias de comandos de ActiveX, o un lenguaje de programación como Microsoft Visual C++.

- Mediante la utilización de la búsqueda con DTS, que proporciona la capacidad de realizar consultas con una o más cadenas de consultas parametrizadas con nombre que permiten que una transformación personalizada recupere datos desde ubicaciones distintas de la fila de origen o de destino inmediato que se está transformando.

TRANSFORMAR DATOS

Durante el paso de migración de datos, es a menudo necesario transformar los datos operacionales a un formato distinto apropiado para el diseño del Data warehouse.

[11] Tomado de Microsoft SQL Server Books Online

Como ejemplos de transformación podemos citar:

- Cambiar todos los caracteres alfabéticos a mayúsculas.
- Calcular valores nuevos a partir de datos existentes, incluyendo agregación de datos y resumen.
- Separar el único valor de un dato en varios valores como, por ejemplo, el código de un producto en formato nnnn-descripción en valores de descripción y códigos separados, o una fecha en formato MMDDAA en valores separados de mes, día y año.
- Mezclar valores separados de datos en un único valor como, por ejemplo, concatenar el apellido y el nombre.
- Pasar los datos de una representación a otra, como convertir los valores de 1, 2, 3 y 4 a I, II, III y IV.

La transformación de los datos también implica dar formato y modificar los datos extraídos de los sistemas operacionales en valores derivados o mezclados que son más útiles en el Data warehouse. Por ejemplo, copiar el valor de **Fecha** desde el sistema transaccional al esquema en forma de estrella del Data warehouse, implica separar la fecha en los componentes **Mes**, **Trimestre** y **Año**. Estos componentes de la fecha son necesarios para el tipo de análisis de negocio que se realiza en el Data warehouse.

El proceso de transformación tiene lugar, normalmente, durante el proceso de migración: cuando los datos se copian directamente desde el origen operacional o desde una base de datos intermedia, porque los datos han sido normalizados.

La transformación y la migración de los datos pueden realizarse en un único paso mediante los asistentes para importación y exportación con DTS. La transformación y la migración de los datos desde el esquema del sistema operacional OLTP a un esquema de estrella del Data warehouse implica utilizar los asistentes para importación y exportación con DTS, por ejemplo, para:

- Crear una consulta para extraer todos los datos con el detalle necesario (hechos).
- En la tabla **log_cabecera_his**, separar **Fecha** en los componentes **Mes, Trimestre** y **Año** y agregarlos a una tabla **Periodo**, mediante una secuencia de comandos de Microsoft ActiveX.
- Realizar una copia de la tabla **ref_cod_tran_plat**.
- Crear una consulta para generar los datos de **Resumen**.

Cada paso, por ejemplo, puede construirse como un paquete separado, que se almacena en la base de datos **msdb** de Microsoft SQL Server y se programa para ejecutarse cada viernes a media noche.

Además de realizar la transformación de datos basada en inserciones, DTS proporciona consultas controladas por datos, en las que los datos se leen del origen y se transforman, y una consulta parametrizada se ejecuta en el destino, utilizando los valores transformados en la fila de destino.

Nota Cuando utilice DTS para crear tablas de hechos con el objeto de que sean utilizadas con los Servicios OLAP de Microsoft SQL Server, no cree ningún

agregado mientras migre los datos. Los Servicios OLAP están diseñados específicamente para crear los agregados óptimos una vez que el Data warehouse ha sido llenado con DTS. Tampoco es necesario segmentar una fecha en las columnas semana, mes, trimestre o año en la tabla de dimensiones **Time**. El Asistente para tiempo de los Servicios OLAP proporciona una herramienta automatizada para llevar a cabo este tipo de transformaciones de tiempo.

Diremos que no es necesario, para este proyecto en desarrollo, diseñar y copiar los datos del sistema operacional a una base de datos intermedia, no se requiere realizar ninguna normalización de datos porque la fuente de datos es una sola (un solo OLTP), al realizar una revisión de los datos no se encontró ninguna inconsistencia de los mismos.

Además, no se realizará ningún proceso de transformación de datos por las siguientes razones:

- Se utilizará en lo posterior los Servicios OLAP de Microsoft SQL Server para generar los cubos multidimensionales y los agregados necesarios.
- No amerita cambiar el formato de los datos para migrarlos a la base del data warehouse, principalmente porque con los que tenemos se puede satisfacer los requerimientos del negocio y los usuarios.

A continuación se presenta, una guía, paso a paso el proceso realizado, mediante el uso de los asistentes para importación y exportación con DTS, para obtener los paquetes necesarios para poblar la base de datos del data warehouse.

GUIA PARA LA EXTRACCIÓN Y CARGA DE DATOS

Para iniciar el proceso de exportación de datos desde la fuente de los mismos; ir por **Microsoft SQL Server 7.0, Enterprise Manager** (en el servidor origen). Se abrirá la ventana del administrador empresarial y escoger en el menú **Tools, Data Transformation Services, Export Data..**, como se ilustra en el Gráfico 8.

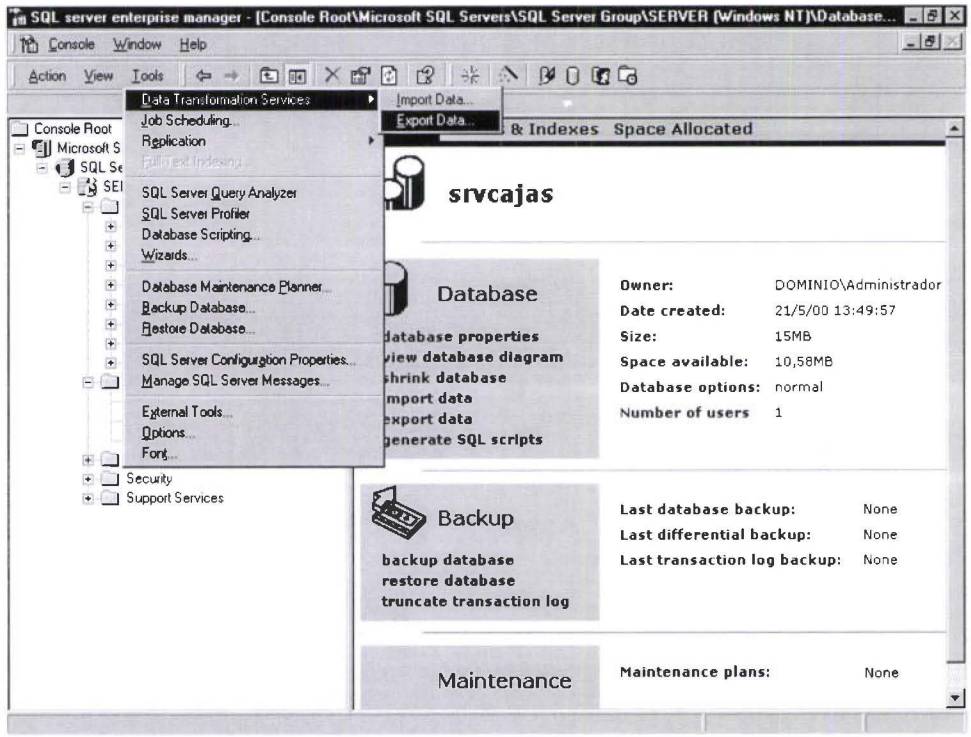


Gráfico 8.

2) Aparecerá la venta de inicio del asistente del DTS de Microsoft SQL Server 7.0

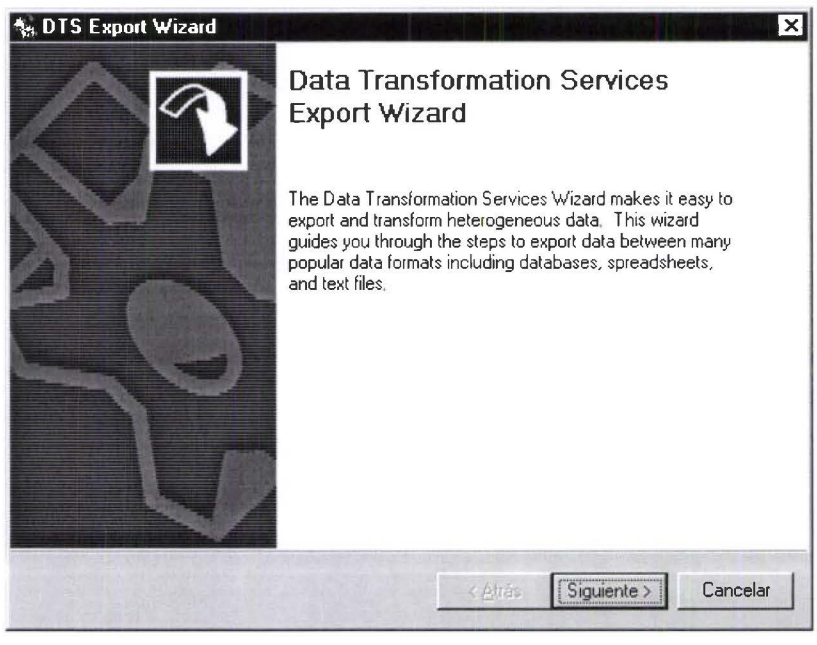


Gráfico 9.

3) Se escoge la fuente de datos, el servidor de origen y un login y contraseña de SQL Server con los permisos y accesos necesarios y la base de datos origen (srvcajas). Luego presionar el botón **Siguiente >**

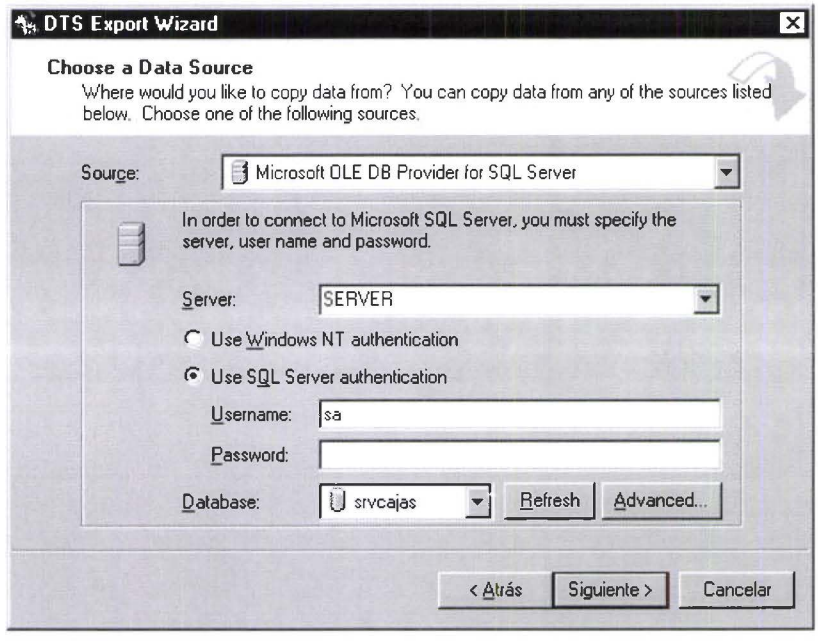


Gráfico 10.

4) Se escoge el destino de los datos, el servidor de destino y un login y contraseña de SQL Server con los permisos y accesos necesarios para crear objetos en SQL Server y la base data warehouse de destino. Luego presionar botón **Siguiente >**

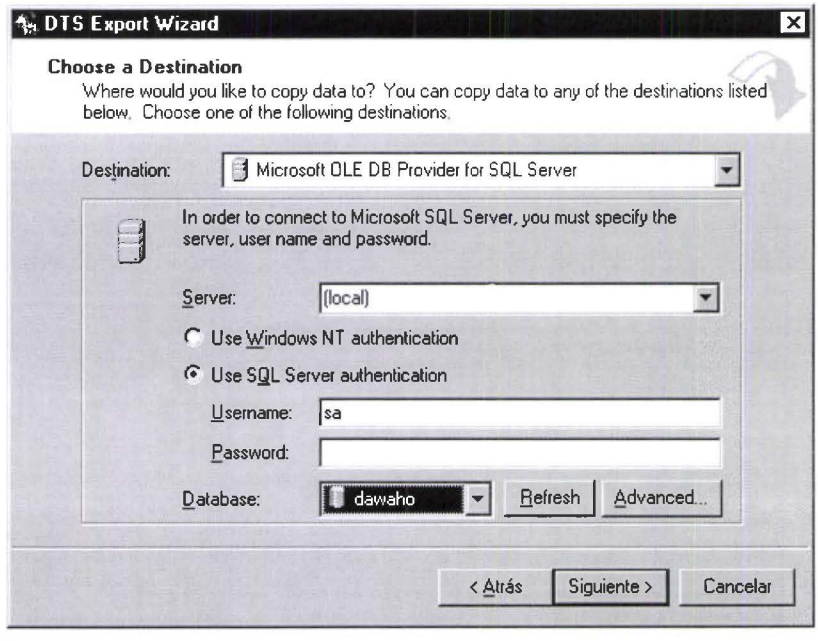


Gráfico 11.

5) Seleccionar las tablas de origen y las tablas destino relacionadas, de donde a donde se migraran los datos.

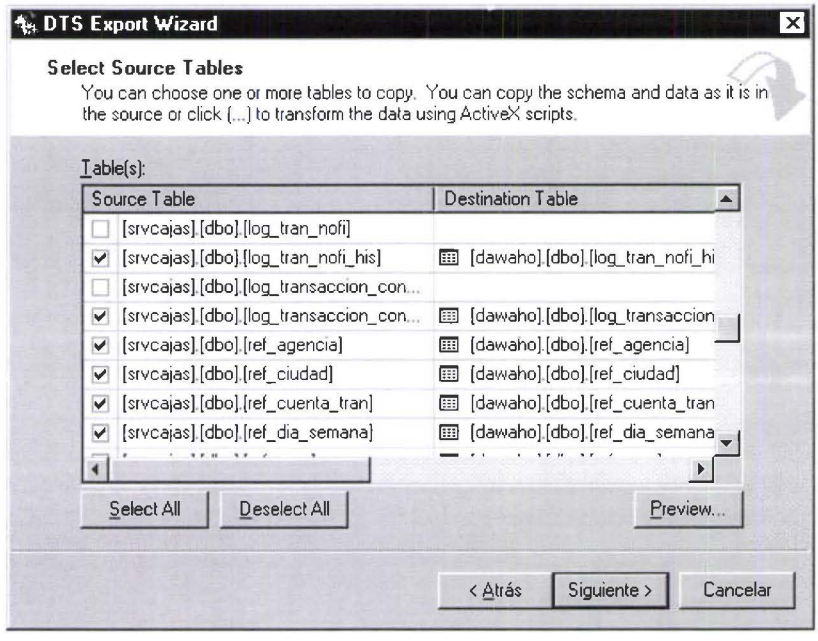


Gráfico 12.

6) En el costado derecho de la tabla destino, se presenta un botón, este nos permite editar la transformación que se llevará a cabo en el momento de la migración de los datos.

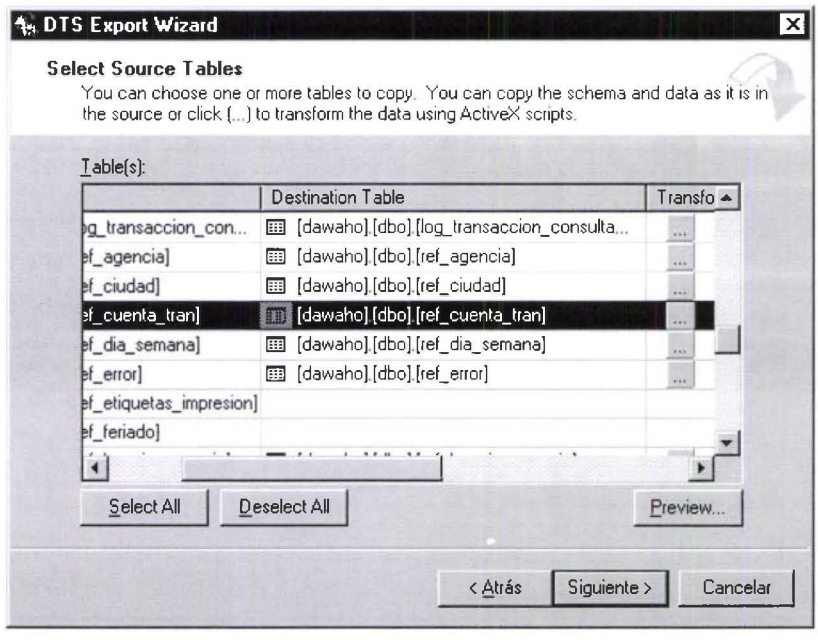


Gráfico 13.

7) Presionando el botón anteriormente mencionado, se puede observar la siguiente información, que indica la columna origen y la columna destino, el tipo de dato de cada campo en la tabla destino, en el caso de que ya se encuentre creada, caso contrario permite activar la opción de crearla en el momento mismo de la ejecución de la exportación de datos. Específicamente para nuestro caso de estudio, se escoge la opción **Añadir filas a la tabla destino**, porque la estructura de las tablas ya están creadas.

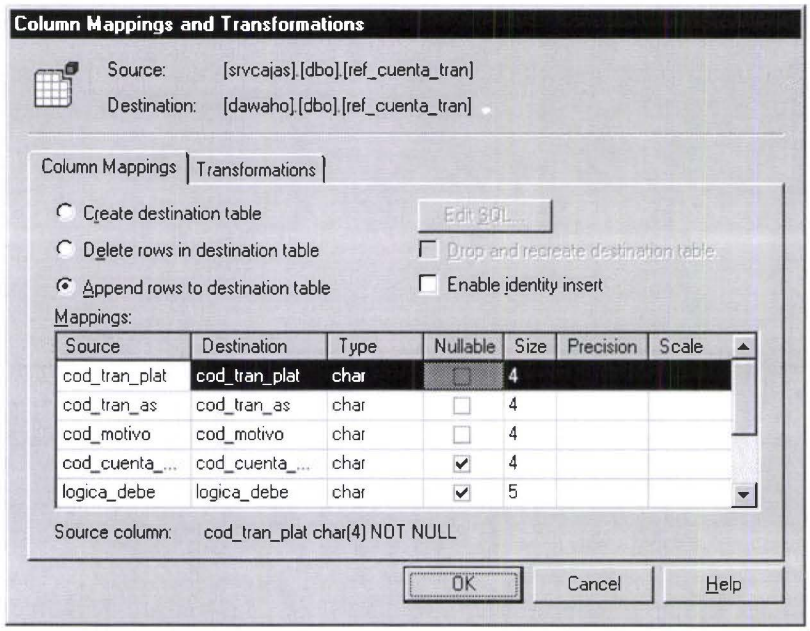


Gráfico 14.

8) Luego de haber realizado el paso anterior para cada una de las tablas, el siguiente paso es escoger cuando se ejecutará el proceso de exportación de datos, si se calendarizará su ejecución y la manera como se guardará este paquete en el Servidor.

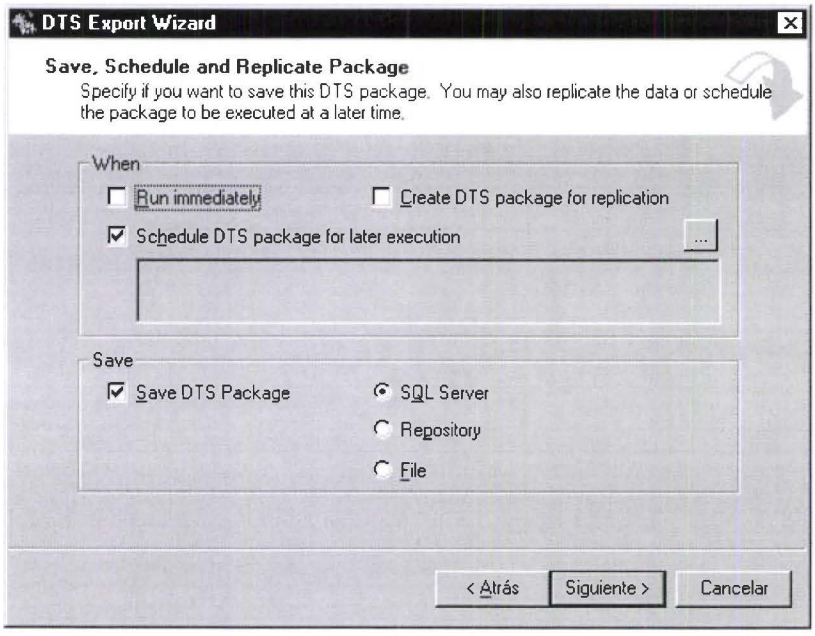


Gráfico 15.

9) Cuando se escoge la opción de calendarización, para la ejecución del paquete de DTS, aparecerá una ventana como la siguiente:

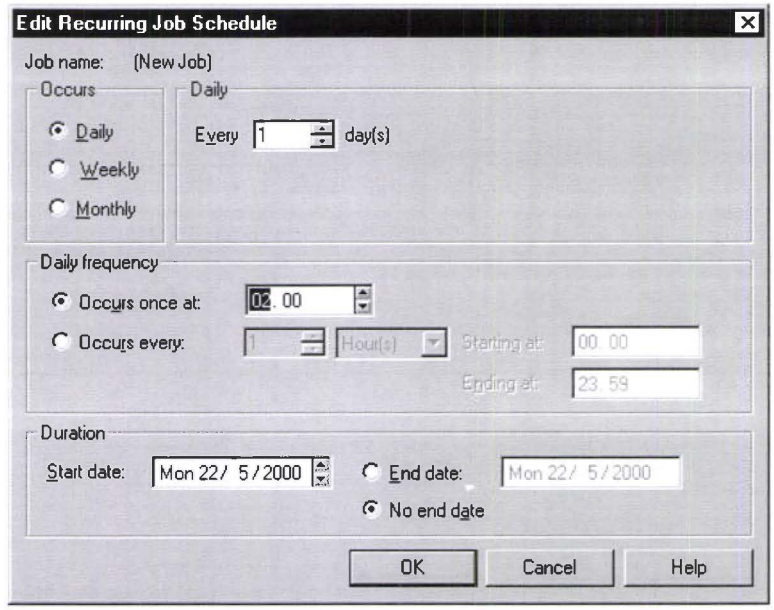


Gráfico 16.

10) Al escoger la opción de guardar el paquete de DTS, se ingresará un nombre con el que se identificara el objeto, una descripción del mismo, el servidor en el que se guardará y el login y password de autentificación para ese servidor, aparecerá una ventana como la siguiente:

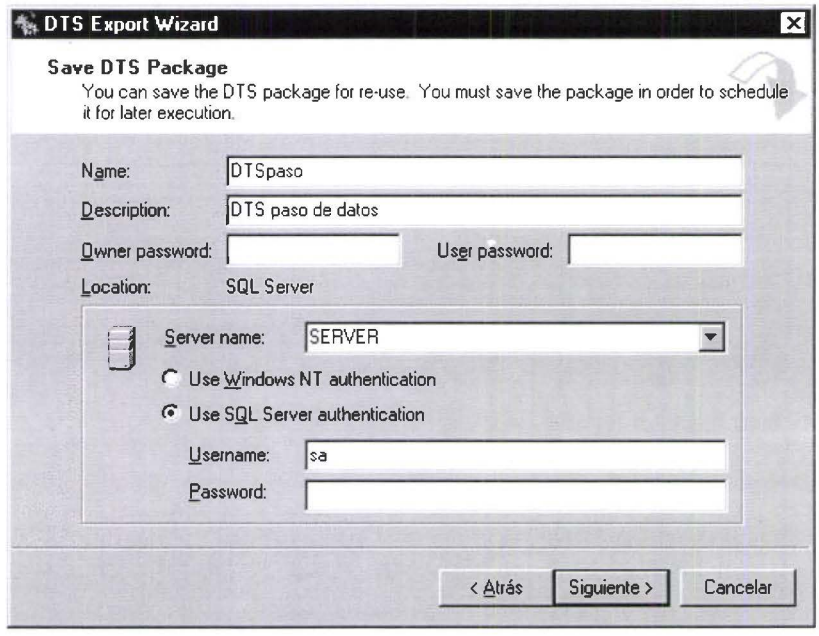


Gráfico 17.

11) Cuando se ha terminado el proceso de definición del paquete para exportación de datos, se despliega un resumen como el siguiente:

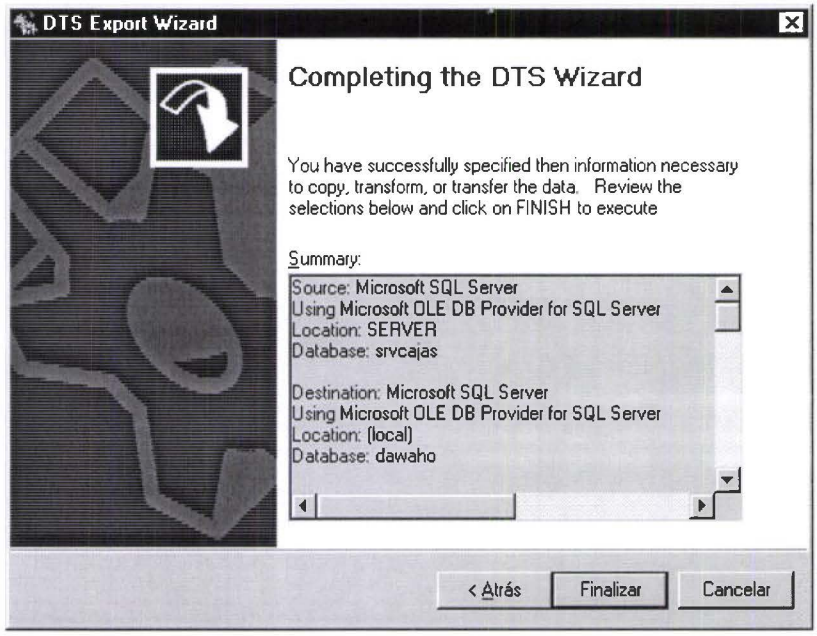


Gráfico 18.

12) Luego de presionar el botón **Finalizar** del gráfico 18, se presentará una pantalla como la del Gráfico 19, que indica que el proceso de creación del paquete se está realizando.

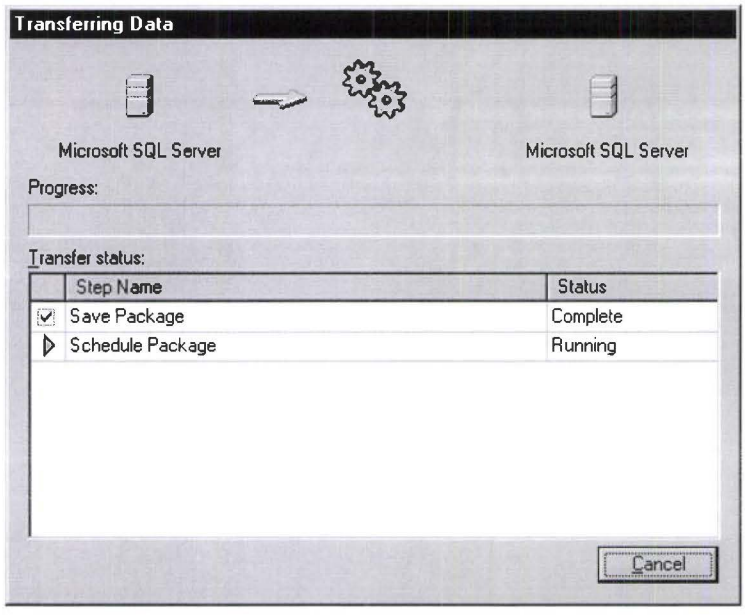


Gráfico 19.

13) Se observará un **Status** como **Complete** cuando las tareas de creación y definición del paquete de DTS se llevaron a cabo con éxito:

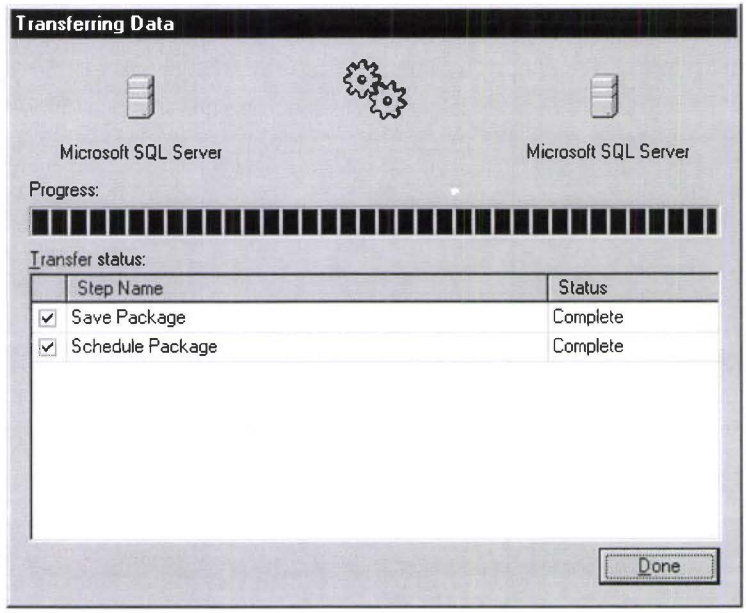


Gráfico 20.

14) Si en el **paso 8** se seleccionó guardar el paquete como un Paquete Local aparecerá de la forma siguiente dentro de **Data Transformation Services**.

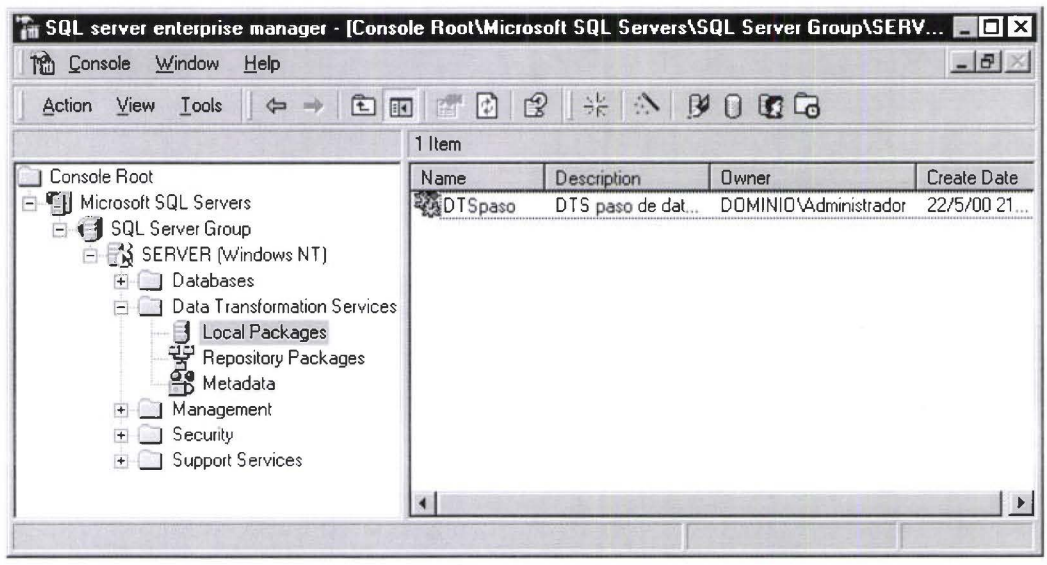


Gráfico 21.

15) En la pantalla anterior, se puede observar el paquete con el nombre que se le asignó, al dar doble clic sobre este aparecerá una ventana con el detalle del diseño del paquete construido, es decir, se mostrarán cada una de las transformaciones que existen entre la base operacional y la base de data warehouse, específicamente entre cada una de las tablas que intervienen en la exportación de datos.

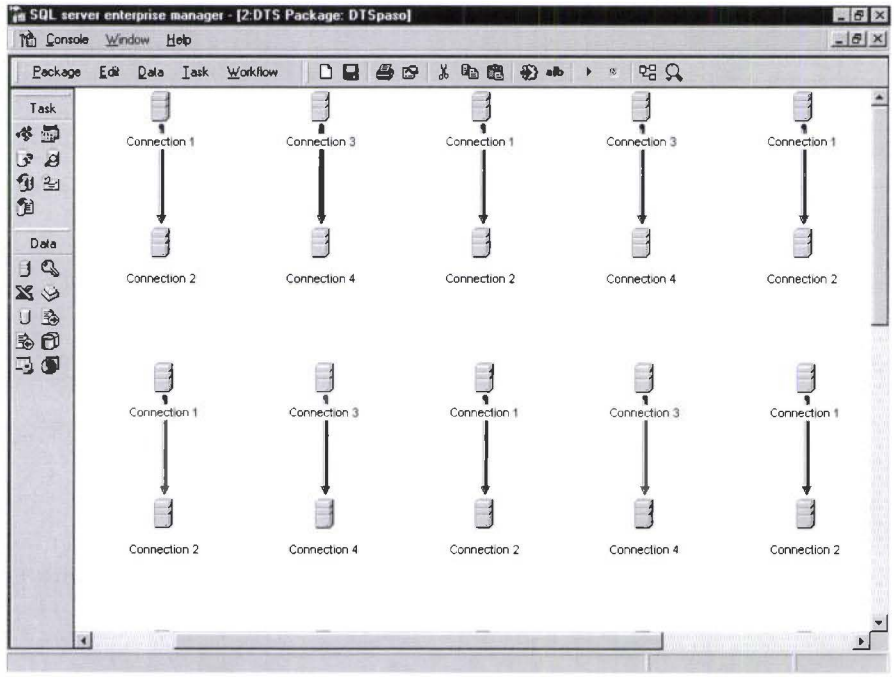


Gráfico 22.

16) Para modificar las propiedades de cada conexión (origen o destino de datos) se debe dar un doble click sobre la conexión, como se indica en los Gráficos 23 y 24:

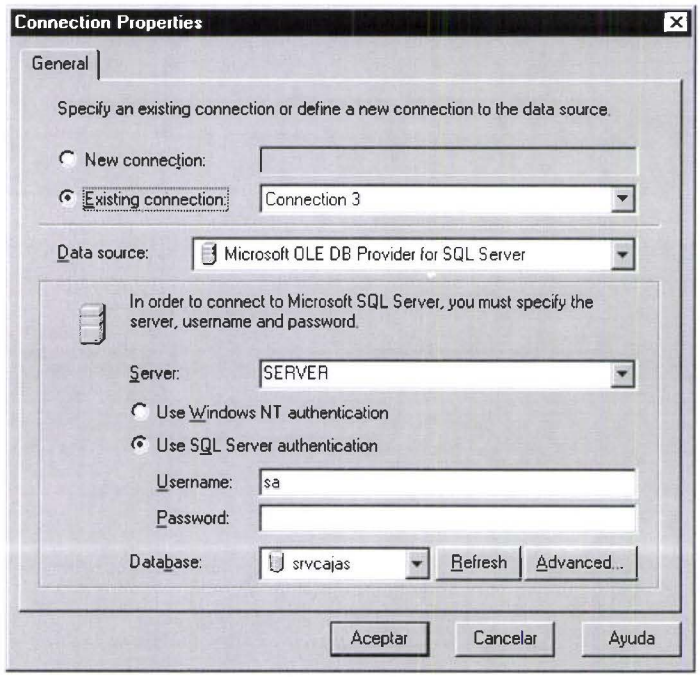


Gráfico 23. Conexión origen

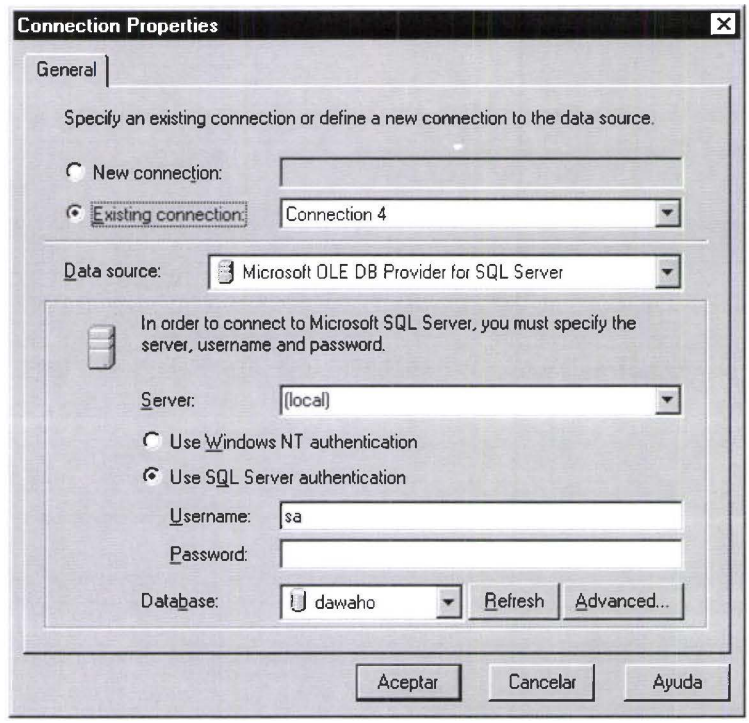


Gráfico 23. Conexión destino

17) Para modificar las propiedades de cada transformación de datos se debe dar un doble click sobre la flecha que se encuentra de conexión a conexión (Gráfico 22), la ventana siguiente ilustra la información desplegada para la pestaña **Fuente** al realizar este evento, se puede observar el SQL query que se ejecuta para la exportación de datos:

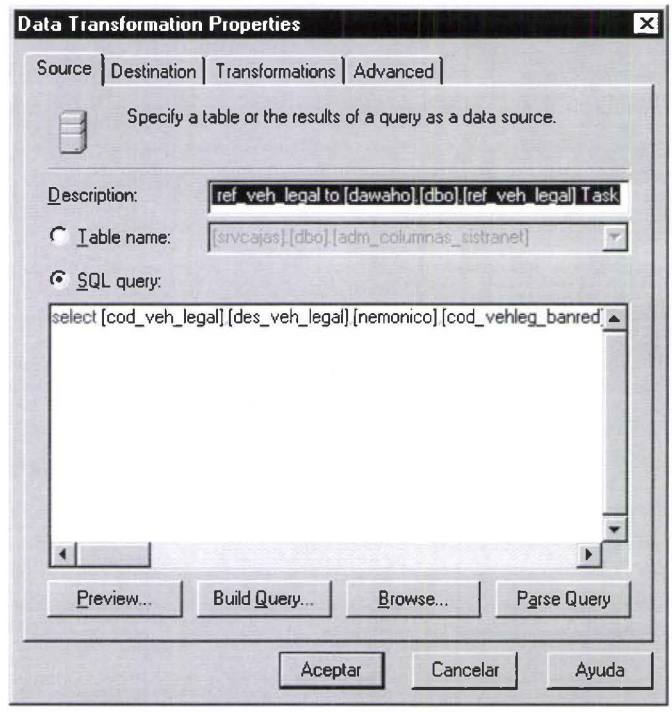


Gráfico 24.

18) En la pestaña **Destino** tenemos la información de los campos disponibles, miembros de la tabla destino, para la transformación de datos.

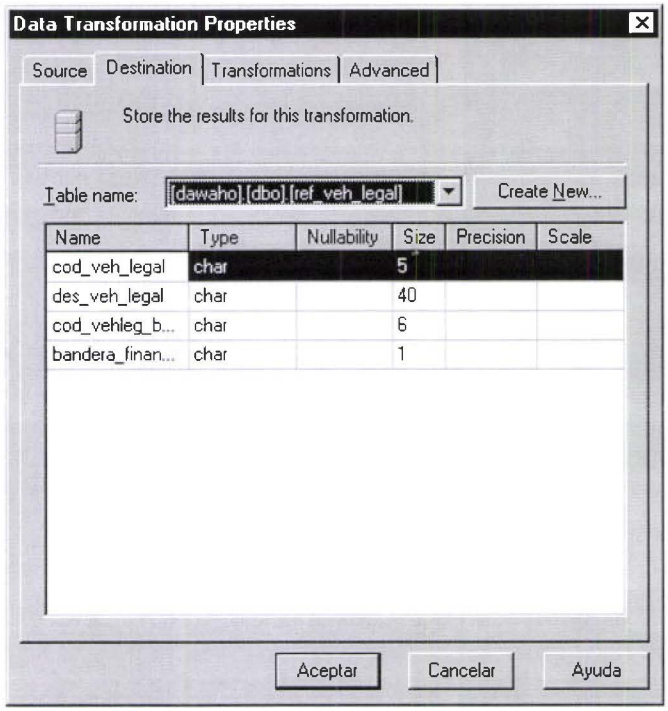


Gráfico 25.

19) En la pestaña **Transformaciones** se tiene en forma gráfica la relación entre los campos de la tabla origen y los campos de la tabla destino.

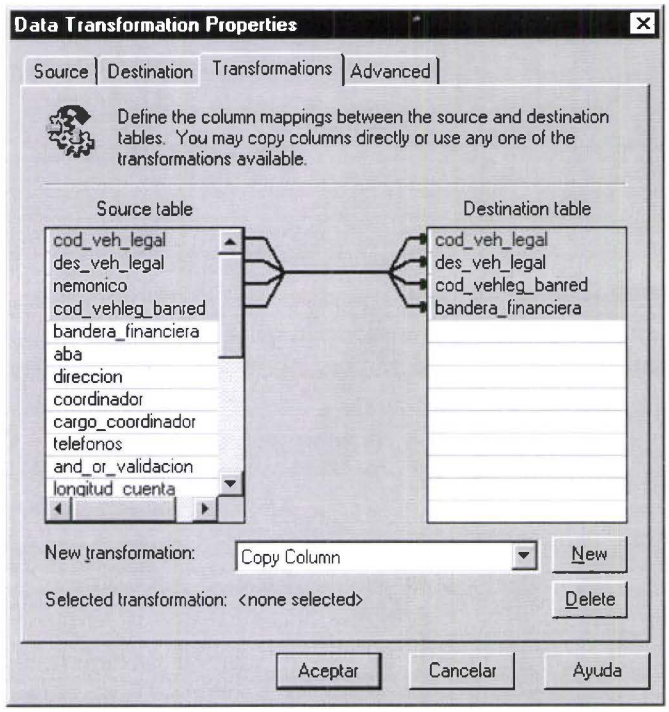


Gráfico 26.

20) Para modificar o eliminar cada una de las relaciones entre campos de las tablas se debe marcar el campo origen y el campo destino y presionar el botón **Delete.**

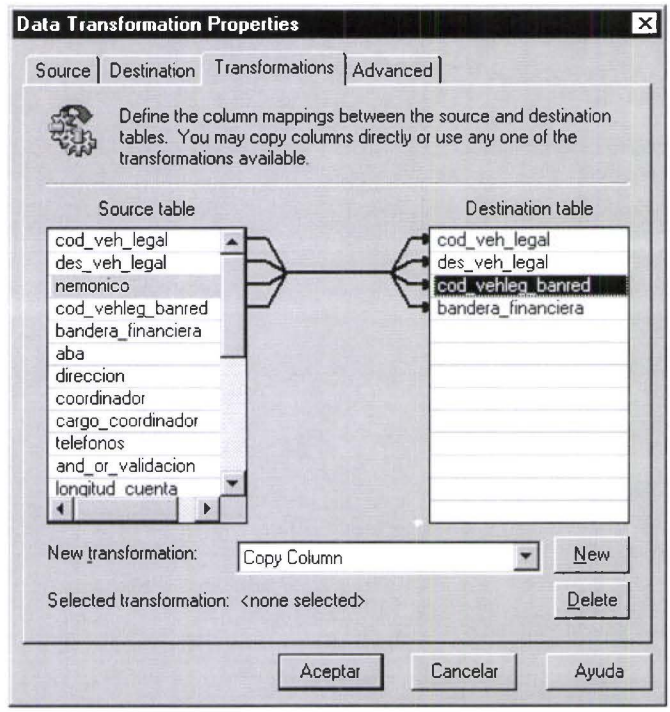


Gráfico 27

21) De manera similar al paso anterior se puede añadir una relación entre las columnas de las tablas origen y destino, presionando el botón **New**, para ello se debe escoger también el tipo de transformación a realizar, como en el Gráfico se indica **Copy Column**.

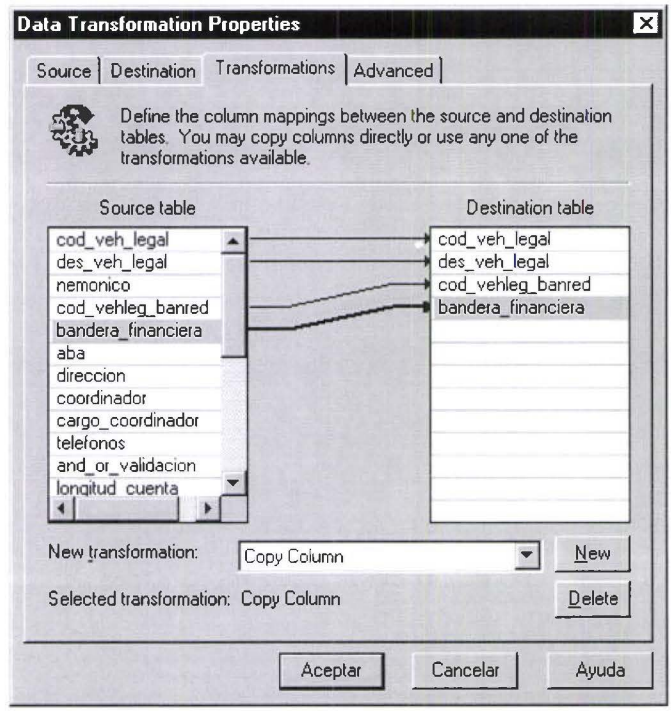


Gráfico 28.

22) Para pestaña de opciones avanzadas nos presenta información como la que se indica en el Gráfico 29.Para el proyecto estas opciones se las ha dejado con sus valores por default.

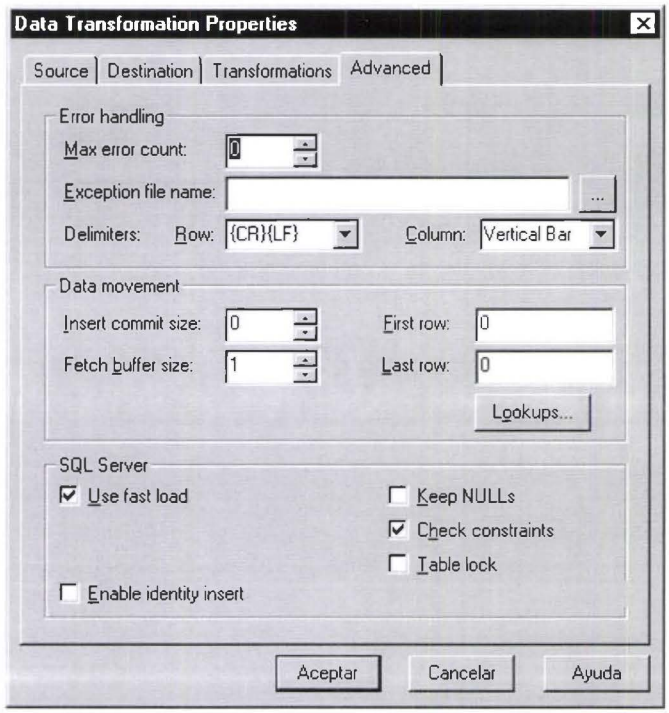


Gráfico 29.

23) Si en el paso 8 se escogió crear una tarea calendarizada para la ejecución del paquete DTS dentro de **Management, SQL Server Agent**, **Jobs** se tendrá creada una tarea como la que se muestra en el Gráfico 30

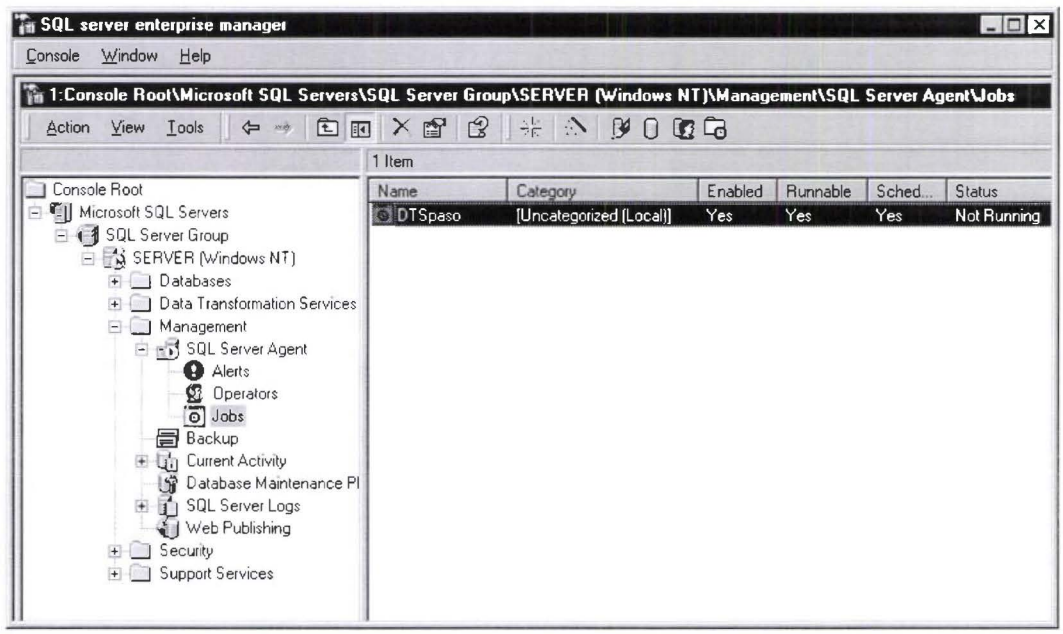


Gráfico 30.

24) Para modificar las propiedades de la tarea creada, se debe dar un doble click sobre la tarea. En la pestaña **general**, se presenta información como la que se muestra a continuación:

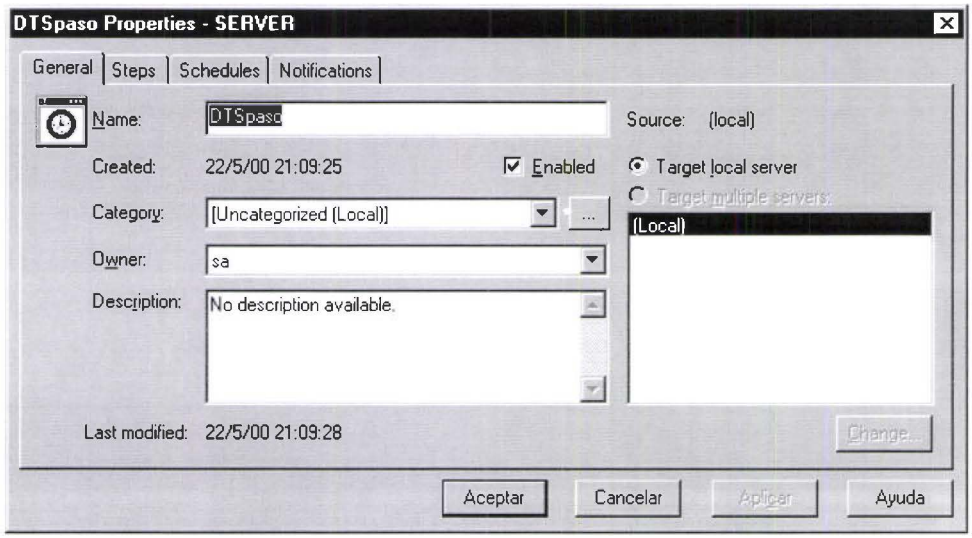


Gráfico 31.

25) En la pestaña **Steps**, se mostrará el o los pasos que se ejecutarán en el momento de la corrida de esta tarea, con el botón **Edit..** , se puede acceder al código que se ha creado para la tarea y la posibilidad de modificarla. Esto se muestra en los Gráficos 32 y 33.

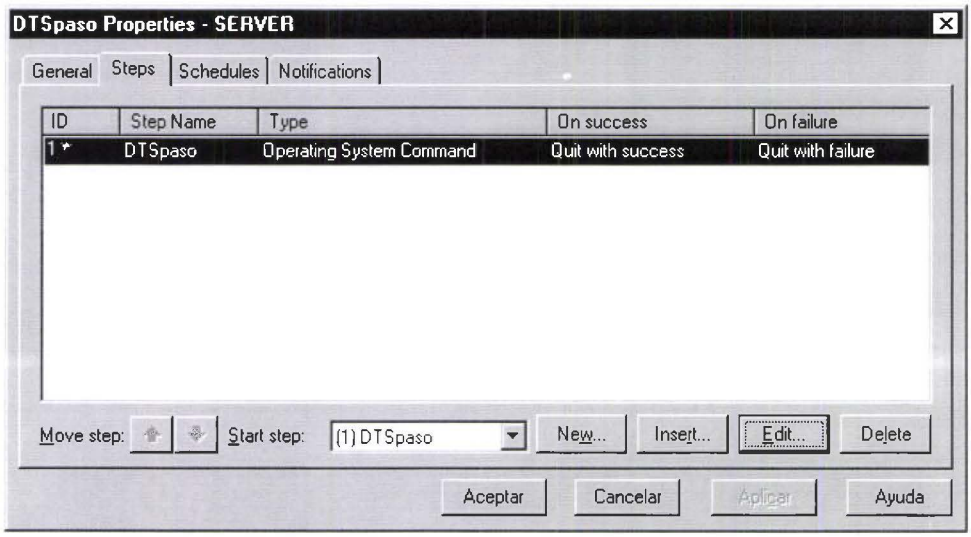


Gráfico 32.

26) En el gráfico 33 se puede observar que en el comando ingresado, se encuentra el programa **DTSRun**, para ejecutar esta tarea de transformación de datos:

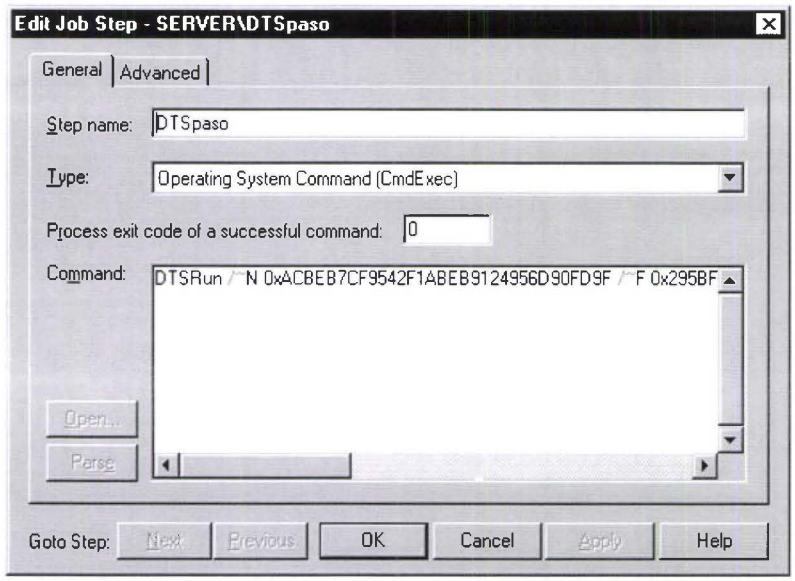


Gráfico 33.

27) En la pestaña **Schedules**, se puede modificar el cuando se ejecuta la tarea presionando el botón **Edit..**

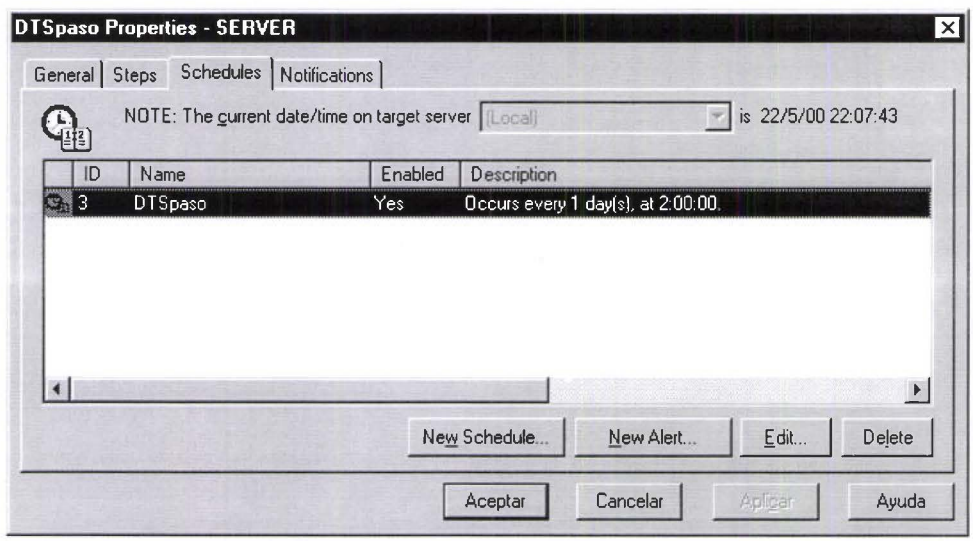


Gráfico 34.

28) En la pestaña **Notificaciones**, podemos configurar las alertas necesarias ha reportarse al personal encargado de verificar y controlar la correcta ejecución de las tareas que se encuentran vigentes en el Servidor. Se puede enviar un mensaje o e-mail al operador definido cuando ocurra un evento en el proceso, sea que la tarea se realizo satisfactoriamente o cuando la tarea tuvo problemas en su ejecución.

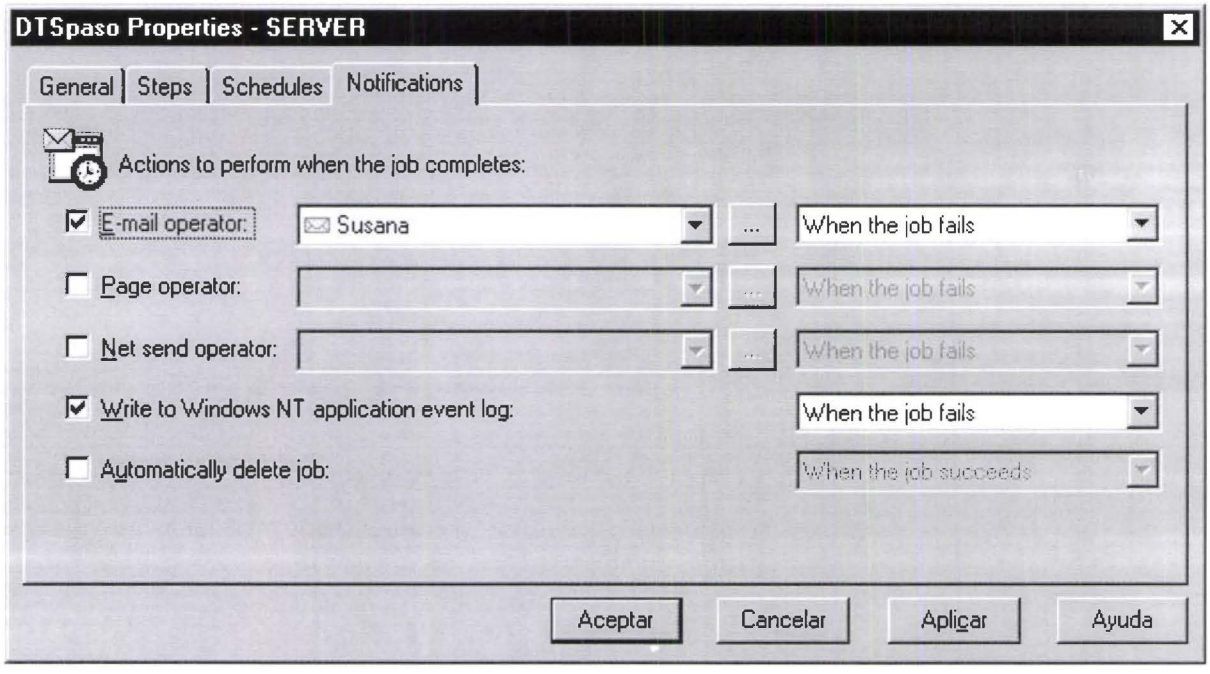


Gráfico 35.

DISEÑAR Y PROCESAR AGREGADOS

Normalmente, las herramientas OLAP se usan para crear y administrar datos de resumen. Los Servicios OLAP de Microsoft SQL Server permiten que los agregados se almacenen en varios formatos y ubicaciones, con conexiones dinámicas a los detalles esenciales del Data warehouse. A menudo, los datos resumidos se generan para satisfacer las consultas ejecutadas con más frecuencia en el Data warehouse. El almacenamiento de los datos previamente agregados aumenta el rendimiento de las consultas y reduce la carga del Data warehouse.

Si un Data warehouse se crea de tal forma que los datos del mismo no cambian, agregar previamente los datos en la tabla de hechos sólo ahorra el espacio en disco necesario para la tabla de hechos. Los Servicios OLAP utilizan el tiempo de proceso que se hubiera utilizado para agregar previamente los datos en la tabla de hechos cuando la procesa para crear un cubo. Sin embargo, los agregados precalculados se almacenan en el cubo y no es necesario que se vuelvan a calcular para cada consulta. Si se utiliza un cubo OLAP híbrido (HOLAP) u OLAP relacional (ROLAP), la tabla de hechos no se copia en el cubo como se hace en los cubos OLAP multidimensionales (MOLAP), con lo que la sobrecarga necesaria para mantener la disponibilidad de los datos de detalle se produce sólo por el tamaño de la tabla de hechos, no por el tiempo de proceso o el tiempo de respuesta de la consulta.

La estrategia de utilizar agregación previa al diseñar un Data warehouse para que sea utilizado por los Servicios OLAP depende de las siguientes variables:

- Estabilidad de los datos.
  Si cambian los datos de origen, las agregaciones previas tienen que ser realizadas cada vez, tanto si se han agregado previamente en la tabla de hechos o en los cubos OLAP que se tienen que volver a crear a partir de la tabla de hechos.

- Tiempo de respuesta de la consulta.
  Si los cubos OLAP se diseñan de manera adecuada, la granularidad del detalle de la tabla de hechos no tendrá efecto en el tiempo de respuesta de aquellas consultas que no tienen acceso a hechos detallados.

- Requisitos de almacenamiento.
  Un nivel más preciso de granularidad en la tabla de hechos requiere más espacio de almacenamiento para la tabla de hechos y para los cubos MOLAP. Es un equilibrio entre aumentar la disponibilidad de detalles y elegir el modo de almacenamiento de los cubos OLAP. Los cubos OLAP tienden a ser grandes independientemente del tipo de almacenamiento; por tanto, el almacenamiento necesario para obtener una granularidad adecuada de detalle en la tabla de hechos puede que no sea especialmente significativo en comparación con las necesidades de almacenamiento de OLAP.

Cuando se diseña el Data warehouse para OLAP, las necesidades del usuario deben guiar la estrategia de agregación previa. La tabla de hechos sólo debe agregarse previamente hasta alcanzar un nivel de granularidad por debajo del cual ningún usuario desearía tener acceso a los detalles.

6. CONSULTAR Y MANTENER EL DATA WAREHOUSE Y LAS BASES DE DATOS OLAP.

Desde el punto de vista del usuario, el único proceso visible es la explotación del data warehouse, aunque el éxito del Data Warehouse radica en los procesos iniciales que alimentan la información del mismo y suponen el mayor porcentaje de esfuerzo (en torno a un 80%) a la hora de desarrollar el almacén. Por ello se debe tomar en cuenta el rendimiento de data warehouse.

CONSIDERACIONES DE RENDIMIENTO

El rendimiento es una área que necesitara ser atendida continuamente. El flujo de los datos de warehouse hacia el usuario no debe ser entorpecido por un sistema que no pueda realizar fácil y rápidamente las consultas.

El impacto que una consulta puede causar en el rendimiento del data warehousing debe ser transparente para el usuario. El analista de la información no debería esperar horas para obtener una respuesta, o trabajar mas tarde cuando el sistema ya no este ocupado. Por lo tanto es imperativo que los cuellos de botellas sean manejados antes de implementar una solución de data warehousing.

Generalmente, los datos de un data warehouse alcanzan rápidamente el punto donde la búsqueda por un mejor rendimiento y escalabilidad se convierten en una verdadera necesidad.

Esta búsqueda persigue dos objetivos:

- Aumento de velocidad - la habilidad de ejecutar el mismo requerimiento sobre el mismo volumen de datos en un tiempo menor.
- Aumento de la escalabilidad - la habilidad de obtener el mismo rendimiento con el mismo requerimiento conforme el tamaño de la base de datos aumenta.

Existen varias opciones que están siendo utilizadas en una solución de data warehousing para controlar los cuellos de botella y su impacto en el rendimiento. Algunas son nuevas tecnologías, otras han estado ya por algún tiempo estas son:

Opciones que impactan en Hardware

- procesamiento paralelo
- warehouse distribuidos
- servidores multidimensionales

Opciones que impactan en software

- OLAP (o consultas multidimensionales)
- Data marts
- Regulación de consultas
- Índices avanzados

Los factores que determinan la utilización de una u otra tecnología son el precio, las plataformas actuales de las organizaciones, políticas tecnologías, organizacionales así como también la proyección del crecimiento de datos pues cada tecnología tiene sus limitaciones; todos estos factores tienen distintos pesos para cada organización.

TÉCNICAS DE EXPLOTACIÓN DE LA INFORMACIÓN

Dentro del esquema de Gestión y Explotación del Data Warehouse que se muestra en el gráfico, pasamos a detallar las posibilidades que nos ofrecen las distintas tecnologías incluida la tecnología OLAP, las que revisaremos son:

- OLAP, ROLAP Y MOLAP
- Query & Reporting
- Data Mining o Minería de datos
- Websousing

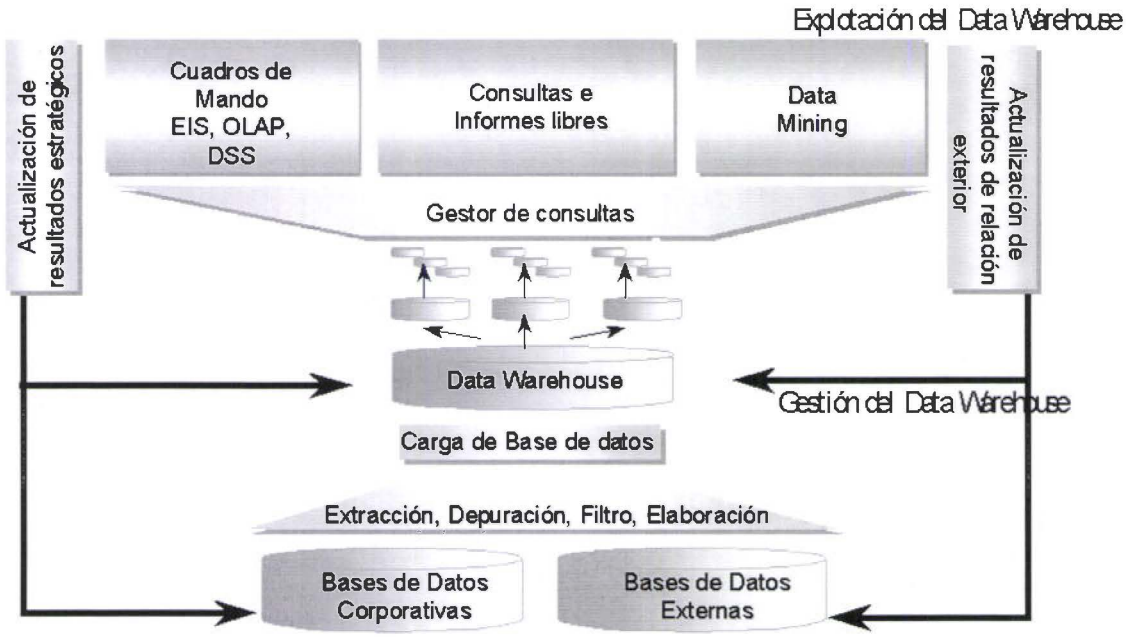


Gráfico 36.

Aquí examinaremos:

- El uso que se puede realizar de las utilidades OLAP del Data Warehouse para análisis multidimensionales
- Las facilidades de obtención de información mediante consultas e informes libre, y el uso de técnicas de Data Mining que nos permitan descubrir "información oculta" en los datos mediante el uso de técnicas estadísticas.[12]

[12] Tomado de http://www.map.es/csi/silice/DW225.html

TECNOLOGIA MULTIDIMENSIONAL

Las bases de datos multidimensionales son consideradas como el último paso en la evolución de las bases de daos. A pesar de que las bases de daos multidimensionales no son nuevas, pues desde la década de los 60's ha ido madurando, es actualmente cuando, para alcanzar el objetivo de toda organización moderna(utilizar sus datos), surge como la alternativa ideal para análisis de la información de los negocios.

La tecnología de base de datos multidimensionales esta optimizada para un alto rendimiento en el acceso, manipulación y análisis de los datos, no ha sido diseñada para tomar el lugar de otras tecnologías como la relacional, empleadas para aplicaciones transaccionales intensivas. De hecho el proveer un punto para la integración, manipulación y análisis de datos, las tecnologías de base de datos multidimensionales pueden ser vistas como el complemento de la tecnología de base de datos relacionales.

Los datos en la tabla relacional están guardados en registros y cada registro esta dividido en campos.

En base de datos multidimensionales los datos son almacenados lógicamente en arreglos (componente fundamental de una estructura multidimensional)|, utilizando una de dos opciones: hipercubos o multicubos

OLAP, ROLAP, MOLAP

La explotación del Data Warehouse mediante información de gestión, se fundamenta básicamente en los niveles agrupados o calculados de información.

La información de gestión se compone de conceptos de información y coeficientes de gestión, que los cuadros directivos de la empresa pueden consultar según las dimensiones de negocio que se definan.

Los sistemas de soporte a la decisión usando tecnologías de Data Warehouse, se llaman sistemas OLAP (siglas de On Line Analytical Processing (OLAP). En general, estos sistemas OLAP deben:

- Soportar requerimientos complejos de análisis
- Analizar datos desde diferentes perspectivas
- Soportar análisis complejos contra un volumen ingente de datos

La funcionalidad de los sistemas OLAP se caracteriza por ser un análisis multidimensional de datos corporativos, que soportan los análisis del usuario y unas posibilidades de navegación, seleccionando la información a obtener.

Normalmente este tipo de selecciones se ve reflejada en la visualización de la estructura multidimensional, en unos campos de selección que nos permitan elegir el nivel de agregación (jerarquía) de la dimensión, y/o la elección de un dato en concreto, la visualización de los atributos del sujeto, frente a una(s) dimensiones en modo tabla, pudiendo con ello realizar, entre otras las siguientes acciones:

Acción	Descripción
Rotar (*Swap*)	alterar las filas por columnas (permutar dos dimensiones de análisis)
Bajar (*Down*)	bajar el nivel de visualización en las filas a una jerarquía inferior
Detallar (*Drilldown*)	informar para una fila en concreto, de datos a un nivel inferior
Expandir (*Expand*)	id. anterior sin perder la información a nivel superior para éste y el resto de los valores
Colapsar (*Collapse*)	operación inversa de la anterior.

Los términos que se indican en la tabla son los más comunes, pero existen muchos más.[13]

En forma general, existen dos arquitecturas diferentes para los sistemas OLAP: OLAP multidimensional (MOLAP) y OLAP relacionales (ROLAP).

Sistemas MOLAP

La arquitectura MOLAP usa unas bases de datos multidimensionales para proporcionar el análisis, su principal premisa es que el OLAP está mejor implantado almacenando los datos multidimensionalmente. Por el contrario, la arquitectura ROLAP cree que las capacidades OLAP están perfectamente implantadas sobre bases de datos relacionales.

Un sistema MOLAP usa una base de datos propietaria multidimensional, en la que la información se almacena multidimensionalmente, para ser visualizada multidimensionalmente.

El sistema MOLAP utiliza una arquitectura de dos niveles: La bases de datos multidimensionales y el motor analítico.

- La base de datos multidimensional es la encargada del manejo, acceso y obtención del dato.
- El nivel de aplicación es el responsable de la ejecución de los requerimientos OLAP. El nivel de presentación se integra con el de aplicación y proporciona un interfaz a través del cual los usuarios finales visualizan los análisis OLAP. Una arquitectura cliente/servidor permite a varios usuarios acceder a la misma base de datos multidimensional.

La información procedente de los sistemas operacionales, se carga en el sistema MOLAP, mediante una serie de rutinas batch. Una vez cargado el dato elemental en la Base de Datos multidimensional (MDDB), se realizan una serie de cálculos en batch, para calcular los datos agregados, a través de las dimensiones de negocio, rellenando la estructura MDDB.

[13] Para ampliar el glosario sobre términos OLAP visitar http://www.kenan.com/acumate/olaptrms.htm

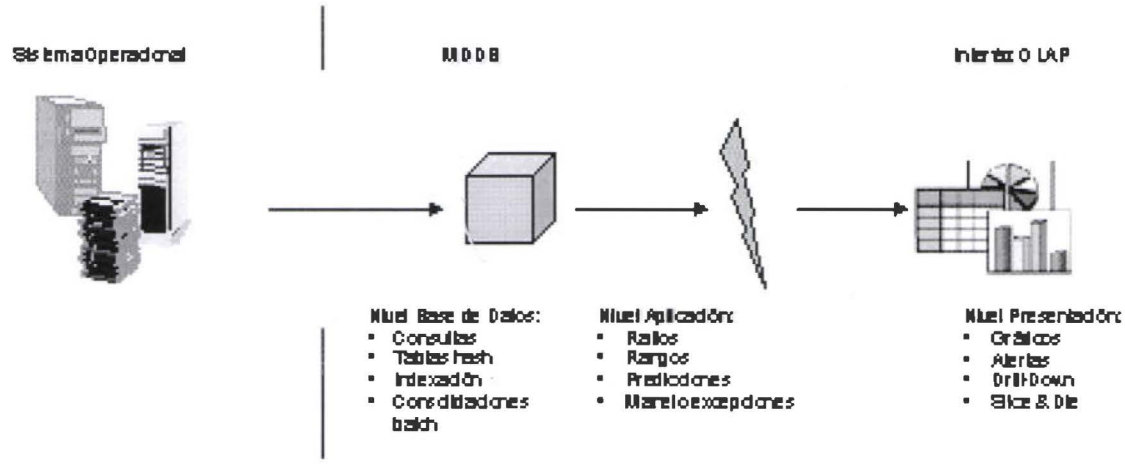


Gráfico 37.

Tras rellenar esta estructura, se generan unos índices y algoritmos de tablas hash para mejorar los tiempos de accesos a las consultas.

Una vez que el proceso de compilación se ha acabado, la MDDB está lista para su uso. Los usuarios solicitan informes a través del interfase, y la lógica de aplicación de la MDDB obtiene el dato.

La arquitectura MOLAP requiere unos cálculos intensivos de compilación. Lee de datos precompilados, y tiene capacidades limitadas de crear agregaciones dinámicamente o de hallar ratios que no se hayan precalculados y almacenados previamente.

Sistemas ROLAP

La arquitectura ROLAP, accede a los datos almacenados en un Data Warehouse para proporcionar los análisis OLAP. La premisa de los sistemas ROLAP es que las capacidades OLAP se soportan mejor contra las bases de datos relacionales.

El sistema ROLAP utiliza una arquitectura de tres niveles. La base de datos relacional maneja los requerimientos de almacenamiento de datos, y el motor ROLAP proporciona la funcionalidad analítica.

- El nivel de base de datos usa bases de datos relacionales para el manejo, acceso y obtención del dato.
- El nivel de aplicación es el motor que ejecuta las consultas multidimensionales de los usuarios.
- El motor ROLAP se integra con niveles de presentación, a través de los cuales los usuarios realizan los análisis OLAP.

Después de que el modelo de datos para el Data Warehouse se ha definido, los datos se cargan desde el sistema operacional. Se ejecutan rutinas de bases de datos para agregar el dato, si así es requerido por el modelos de datos.

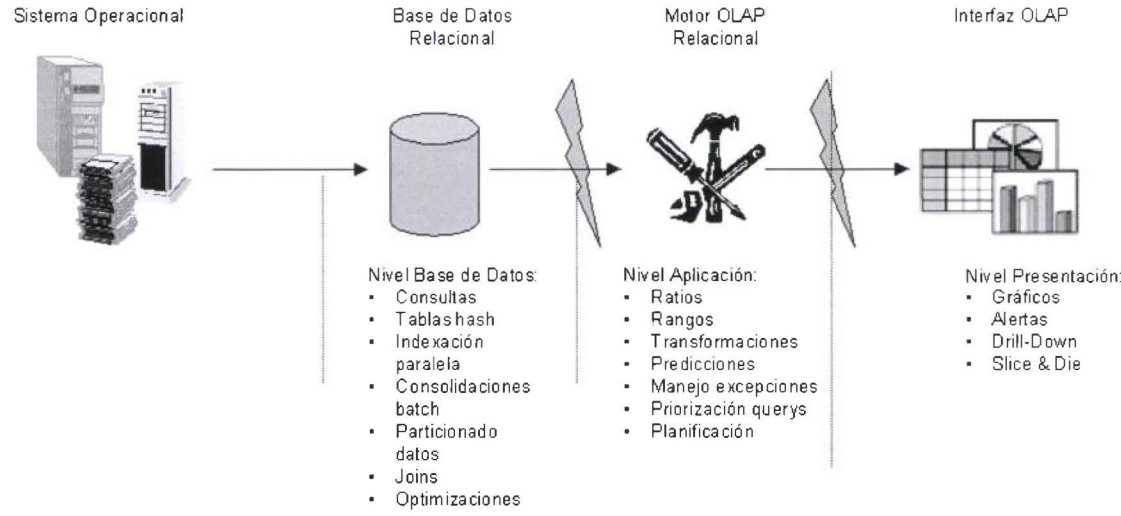


Gráfico 38.

Se crean entonces los índices para optimizar los tiempos de acceso a las consultas.
Los usuarios finales ejecutan sus análisis multidimensionales, a través del motor ROLAP, que transforma dinámicamente sus consultas a consultas SQL. Se ejecutan estas consultas SQL en las bases de datos relacionales, y sus resultados se relacionan mediante tablas cruzadas y conjuntos multidimensionales para devolver los resultados a los usuarios.

La arquitectura ROLAP es capaz de usar datos precalculados si estos están disponibles, o de generar dinámicamente los resultados desde los datos elementales si es preciso. Esta arquitectura accede directamente a los datos del Data Warehouse, y soporta técnicas de optimización de accesos para acelerar las consultas. Estas optimizaciones son, entre otras, particionado de los datos a nivel de aplicación, soporte a la desnormalización y joins múltiples.

ROLAP vs. MOLAP (Comparativa)

Cuando se comparan las dos arquitecturas, se pueden realizar las siguientes observaciones:

- El ROLAP delega la negociación entre tiempo de respuesta y el proceso batch al diseño del sistema. Mientras, el MOLAP, suele requerir que sus bases de datos se precompilen para conseguir un rendimiento aceptable en las consultas, incrementando, por tanto los requerimientos batch.

- Los sistemas con alta volatilidad de los datos (aquellos en los que cambian las reglas de agregación y consolidación), requieren una arquitectura que pueda realizar esta consolidación ad-hoc. Los sistemas ROLAP soportan bien esta consolidación dinámica, mientras que los MOLAP están más orientados hacia consolidaciones batch.

- Los ROLAP pueden crecer hasta un gran número de dimensiones, mientras que los MOLAP generalmente son adecuados para diez o menos dimensiones.

- Los ROLAP soportan análisis OLAP contra grandes volúmenes de datos elementales, mientras que los MOLAP se comportan razonablemente en volúmenes más reducidos (menos de 5 Gb)

Por ello, y resumiendo, el ROLAP es una arquitectura flexible y general, que crece para dar soporte a amplios requerimientos OLAP. El MOLAP es una solución particular, adecuada para soluciones departamentales con unos volúmenes de información y número de dimensiones más modestos.

Luego de haber analizado la tecnología OLAP en forma genérica, se presenta un análisis en forma particular de cómo Microsoft SQL Server 7.0 implementa esta tecnología en sus herramientas.

Servicios OLAP de Microsoft SQL Server 7.0

OLAP se centra en buscar tendencias en los datos agregados o resumidos. Los principales objetos utilizados por los programas OLAP son cubos multidimensionales. Un cubo multidimensional guarda un conjunto de datos derivados de tablas de hechos y de dimensiones. Una tabla de hechos guarda datos acerca de un conjunto de transacciones. Las medidas son columnas numéricas de la tabla de hechos que resultan interesantes para los usuarios. Un cubo representa la forma en que estas medidas varían en varias dimensiones.

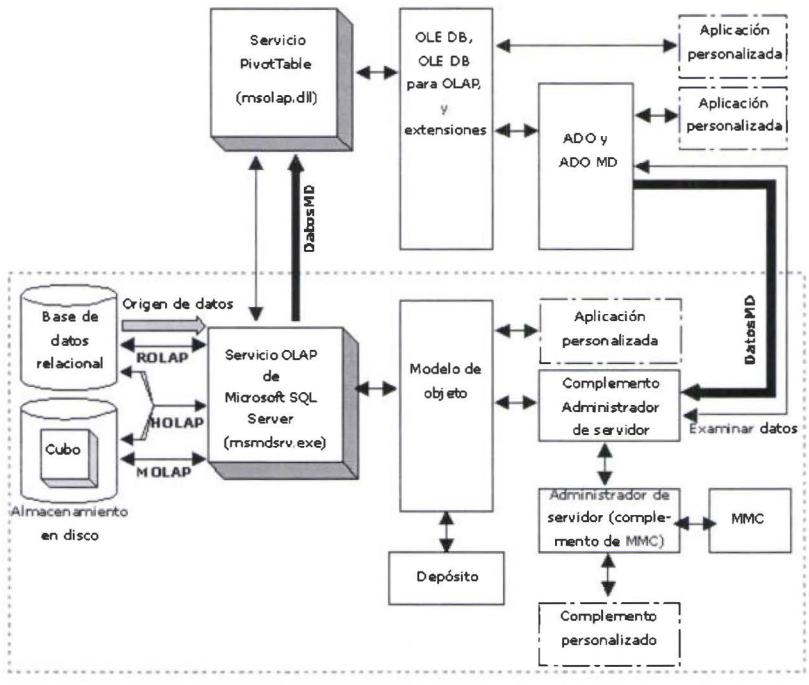


Gráfico 39.Arquitectura OLAP

Los Servicios OLAP proporcionan la capacidad de diseñar, crear y administrar cubos de un Data warehouse y, a continuación, ponerlos a disposición de las aplicaciones cliente escritas utilizando las extensiones OLAP 2.0 de OLE DB, o los Objetos multidimensionales de Microsoft ActiveX Data Objects 2.0 (ADO MD).

El servidor de OLAP realiza consultas multidimensionales de los datos y almacena los resultados en su almacenamiento multidimensional. Aumenta la velocidad de análisis de las tablas de hechos al organizarlas en cubos, almacena los cubos mientras sea necesario y, posteriormente, devuelve los datos rápidamente a los clientes.

El servidor de OLAP se administra a través de una API llamada Objetos de apoyo a la toma de decisiones de Microsoft (DSO, *Decision Support Objects*). Los servicios OLAP proporcionan un complemento para Microsoft Management Console (MMC). Este complemento de MMC utiliza DSO para ofrecer a los administradores una interfaz gráfica sencilla para definir, crear y administrar los cubos creados por el servidor de OLAP. A DSO se le puede llamar también desde las aplicaciones personalizadas, que a su vez se pueden agregar al Administrador de OLAP como un complemento.

Los Servicios OLAP pasan sus datos multidimensionales a un Servicio PivotTable de nivel intermedio. El servicio PivotTable funciona como un proveedor de OLE DB para OLAP. Expone los datos multidimensionales a las aplicaciones usando las extensiones OLAP 2.0 de OLE DB, o la API ADO MD que engloba las extensiones OLAP de OLE DB.

Los Servicios OLAP de Microsoft SQL Server es un nuevo servidor de nivel intermedio para el proceso analítico en línea (OLAP). El sistema de Servicios OLAP incluye un eficaz servidor que construye cubos multidimensionales de datos para su análisis y proporciona al cliente rápido acceso a la información almacenada en el cubo. Microsoft Excel y aplicaciones desarrolladas por otros fabricantes utilizan el servicio PivotTable, que es el proveedor incluido compatible con OLE DB, para recuperar datos multidimensionales del servidor y mostrarlos al usuario.

Los Servicios OLAP organizan los datos de un Data warehouse en cubos multidimensionales con una información de resumen calculada previamente para proporcionar respuestas rápidas a las consultas analíticas complejas.

Entre las características clave de Servicios OLAP están las siguientes:

- Facilidad de uso gracias a su interfaz de usuario y a los asistentes.
- Un modelo flexible y eficaz de datos para la definición y almacenamiento de cubos
- Cubos habilitados para escritura para realizar análisis de escenarios "qué ocurriría si".
- Una arquitectura con capacidad de crecimiento que proporciona varios escenarios de almacenamiento y una solución automatizada al "síndrome de explosión de datos" que existe en las tecnologías OLAP tradicionales.
- Integración de herramientas de administración, seguridad, orígenes de datos y caché de cliente-servidor.
- API ampliamente compatibles y arquitectura abierta para dar soporte a las aplicaciones personalizadas.

Características de los Servicios OLAP

Los Servicios OLAP de Microsoft SQL Server incorporan muchas características para superar la complejidad de la tecnología del Proceso analítico en línea (OLAP). Las características específicas se encuentran agrupadas en los siguientes temas:

- Facilidad de uso
- Modelo flexible de datos
- Escalabilidad
- Integración
- API ampliamente compatibles y arquitectura abierta

QUERY & REPORTING

Las consultas o informes libres trabajan tanto sobre el detalle como sobre las agregaciones de la información.

Realizar este tipo de explotación en un almacén de datos supone una optimización del tradicional entorno de informes (reporting), dado que el Data Warehouse mantiene una estructura y una tecnología mucho más apropiada para este tipo de solicitudes.

Los sistemas de "Query & Reporting", no basados en almacenes de datos se caracterizan por la complejidad de las consultas, los altísimos tiempos de respuesta y la interferencia con otros procesos informáticos que compartan su entorno.

La explotación del Data Warehouse mediante "Query & Reporting" debe permitir una gradación de la flexibilidad de acceso, proporcional a la experiencia y formación del usuario. A este respecto, se recomienda el mantenimiento de al menos tres niveles de dificultad:

- Los usuarios poco expertos podrán solicitar la ejecución de informes o consultas predefinidas según unos parámetros predeterminados.
- Los usuarios con cierta experiencia podrán generar consultas flexibles mediante una aplicación que proporcione una interfaz gráfica de ayuda.
- Los usuarios altamente experimentados podrán escribir, total o parcialmente, la consulta en un lenguaje de interrogación de datos.

Hay una extensa gama de herramientas en el mercado para cumplir esta funcionalidad sobre entornos de tipo Data Warehouse, por lo que se puede elegir el software más adecuado para cada problemática empresarial concreta.

DATA MINING O MINERÍA DE DATOS

El Data Mining es un proceso que, a través del descubrimiento y cuantificación de relaciones predictivas en los datos, permite transformar la información disponible en conocimiento útil de negocio. Esto es debido a que no es suficiente "navegar" por los datos para resolver los problemas de negocio, sino que se hace necesario seguir una

metodología ordenada que permita obtener rendimientos tangibles de este conjunto de herramientas y técnicas de las que dispone el usuario.

Constituye por tanto una de las vías clave de explotación del Data Warehouse, dado que es este su entorno natural de trabajo.

Se trata de un concepto de explotación de naturaleza radicalmente distinta a la de los sistemas de información de gestión, dado que no se basa en coeficientes de gestión o en información altamente agregada, sino en la información de detalle contenida en el almacén. Adicionalmente, el usuario no se conforma con la mera visualización de datos, sino que trata de obtener una relación entre los mismos que tenga repercusiones en su negocio.

Técnicas de Data Mining

Para soportar el proceso de Data Mining, el usuario dispone de una extensa gama de técnicas que le pueden ayudar en cada una de las fases de dicho proceso, las cuales pasamos a describir:

Análisis estadístico

Utilizando las siguientes herramientas:

ANOVA: o Análisis de la Varianza, contrasta si existen diferencias significativas entre las medidas de una o más variables continuas en grupo de población distintos.

Regresión: define la relación entre una o más variables y un conjunto de variables predictoras de las primeras.

Ji cuadrado: contrasta la hipótesis de independencia entre variables.

Componentes principales: permite reducir el número de variables observadas a un menor número de variables artificiales, conservando la mayor parte de la información sobre la varianza de las variables.

Análisis cluster: permite clasificar una población en un número determinado de grupos, en base a semejanzas y desemejanzas de perfiles existentes entre los diferentes componentes de dicha población.

Análisis discriminante: método de clasificación de individuos en grupos que previamente se han establecido, y que permite encontrar la regla de clasificación de los elementos de estos grupos, y por tanto identificar cuáles son las variables que mejor definan la pertenencia al grupo.

Métodos basados en árboles de decisión

El método Chaid (Chi Squared Automatic Interaction Detector) es un análisis que genera un árbol de decisión para predecir el comportamiento de una variable, a partir de una o más variables predictoras, de forma que los conjuntos de una misma rama y un

mismo nivel son disjuntos. Es útil en aquellas situaciones en las que el objetivo es dividir una población en distintos segmentos basándose en algún criterio de decisión.

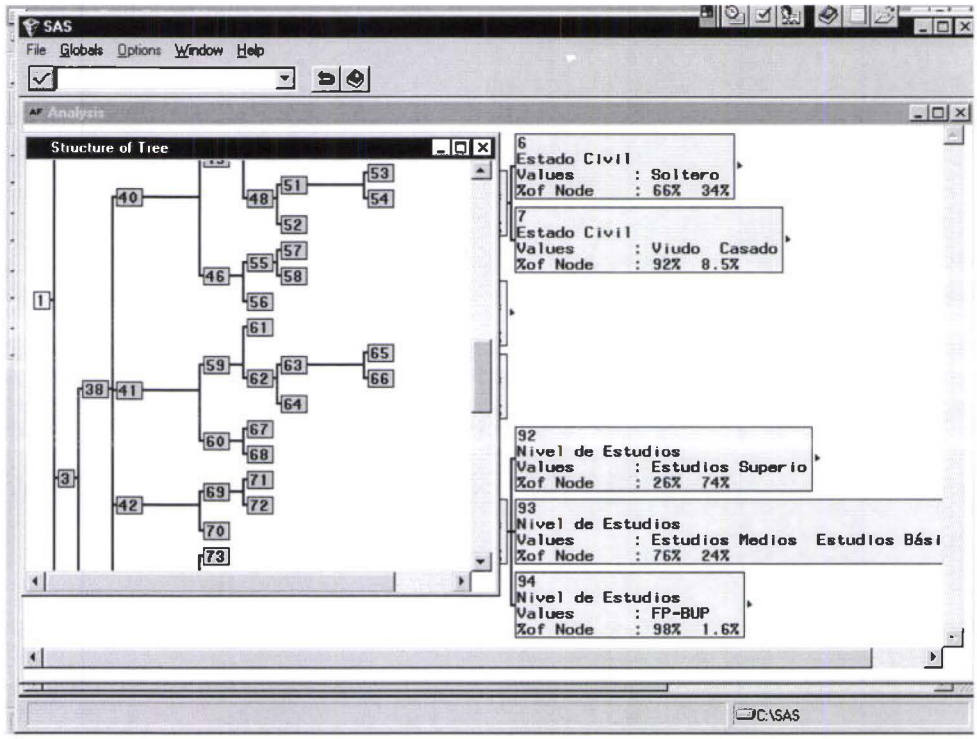


Gráfico 40

El árbol de decisión se construye partiendo el conjunto de datos en dos o más subconjuntos de observaciones a partir de los valores que toman las variables predictoras. Cada uno de estos subconjuntos vuelve después a ser particionado utilizando el mismo algoritmo. Este proceso continúa hasta que no se encuentran diferencias significativas en la influencia de las variables de predicción de uno de estos grupos hacia el valor de la variable de respuesta.

La raíz del árbol es el conjunto de datos íntegro, los subconjuntos y los subsubconjuntos conforman las ramas del árbol. Un conjunto en el que se hace una partición se llama nodo.

El número de subconjuntos en una partición puede ir de dos hasta el número de valores distintos que puede tomar la variable usada para hacer la separación. La variable de predicción usada para crear una partición es aquella más significativamente relacionada con la variable de respuesta de acuerdo con test de independencia de la Chi cuadrado sobre una tabla de contingencia.

Algoritmos genéticos

Son métodos numéricos de optimización, en los que aquella variable o variables que se pretenden optimizar junto con las variables de estudio constituyen un segmento de información. Aquellas configuraciones de las variables de análisis que obtengan mejores valores para la variable de respuesta, corresponderán a segmentos con mayor capacidad reproductiva. A través de la reproducción, los mejores segmentos perduran y su proporción crece de generación en generación. Se puede además introducir elementos

aleatorios para la modificación de las variables (mutaciones). Al cabo de cierto número de iteraciones, la población estará constituida por buenas soluciones al problema de optimización.

Redes neuronales:

Genéricamente son métodos de proceso numérico en paralelo, en el que las variables interactúan mediante transformaciones lineales o no lineales, hasta obtener unas salidas. Estas salidas se contrastan con los que tenían que haber salido, basándose en unos datos de prueba, dando lugar a un proceso de retroalimentación mediante el cual la red se reconfigura, hasta obtener un modelo adecuado.

Lógica difusa:

Es una generalización del concepto de estadística. La estadística clásica se basa en la teoría de probabilidades, a su vez ésta en la técnica conjuntista, en la que la relación de pertenencia a un conjunto es dicotómica (el 2 es par o no lo es). Si establecemos la noción de conjunto borroso como aquel en el que la pertenencia tiene una cierta graduación (¿un día a 20ºC es caluroso?), dispondremos de una estadística más amplia y con resultados más cercanos al modo de razonamiento humano.

Series temporales:

Es el conocimiento de una variable a través del tiempo para, a partir de ese conocimiento, y bajo el supuesto de que no van a producirse cambios estructurales, poder realizar predicciones. Suelen basarse en un estudio de la serie en ciclos, tendencias y estacionalidades, que se diferencian por el ámbito de tiempo abarcado, para por composición obtener la serie original. Se pueden aplicar enfoques híbridos con los métodos anteriores, en los que la serie se puede explicar no sólo en función del tiempo sino como combinación de otras variables de entorno más estables y, por lo tanto, más fácilmente predecibles.

Metodología de aplicación.

Para utilizar estas técnicas de forma eficiente y ordenada es preciso aplicar una metodología estructurada, al proceso de Data Mining. A este respecto proponemos la siguiente metodología, siempre adaptable a la situación de negocio particular a la que se aplique:

Muestreo

Extracción de la población muestral sobre la que se va a aplicar el análisis. En ocasiones se trata de una muestra aleatoria, pero puede ser también un subconjunto de datos del Data Warehouse que cumplan unas condiciones determinadas. El objeto de trabajar con una muestra de la población en lugar de toda ella, es la simplificación del estudio y la disminución de la carga de proceso. La muestra más óptima será aquella que teniendo un error asumible contenga el número mínimo de observaciones.

En el caso de que se recurra a un muestreo aleatorio, se debería tener la opción de elegir El nivel de confianza de la muestra (usualmente el 95% o el 99%).

El tamaño máximo de la muestra (número máximo de registros), en cuyo caso el sistema deberá informar del el error cometido y la representatividad de la muestra sobre la población original.

El error muestral que está dispuesto a cometer, en cuyo caso el sistema informará del número de observaciones que debe contener la muestra y su representatividad sobre la población original.

Para facilitar este paso s debe disponer de herramientas de extracción dinámica de información con o sin muestreo (simple o estratificado). En el caso del muestreo, dichas herramientas deben tener la opción de, dado un nivel de confianza, fijar el tamaño de la muestra y obtener el error o bien fijar el error y obtener el tamaño mínimo de la muestra que nos proporcione este grado de error.

Exploración

Una vez determinada la población que sirve para la obtención del modelo se deberá determinar cuales son las variables explicativas que van a servir como "inputs" al modelo. Para ello es importante hacer una exploración por la información disponible de la población que nos permita eliminar variables que no influyen y agrupar aquellas que repercuten en la misma dirección.

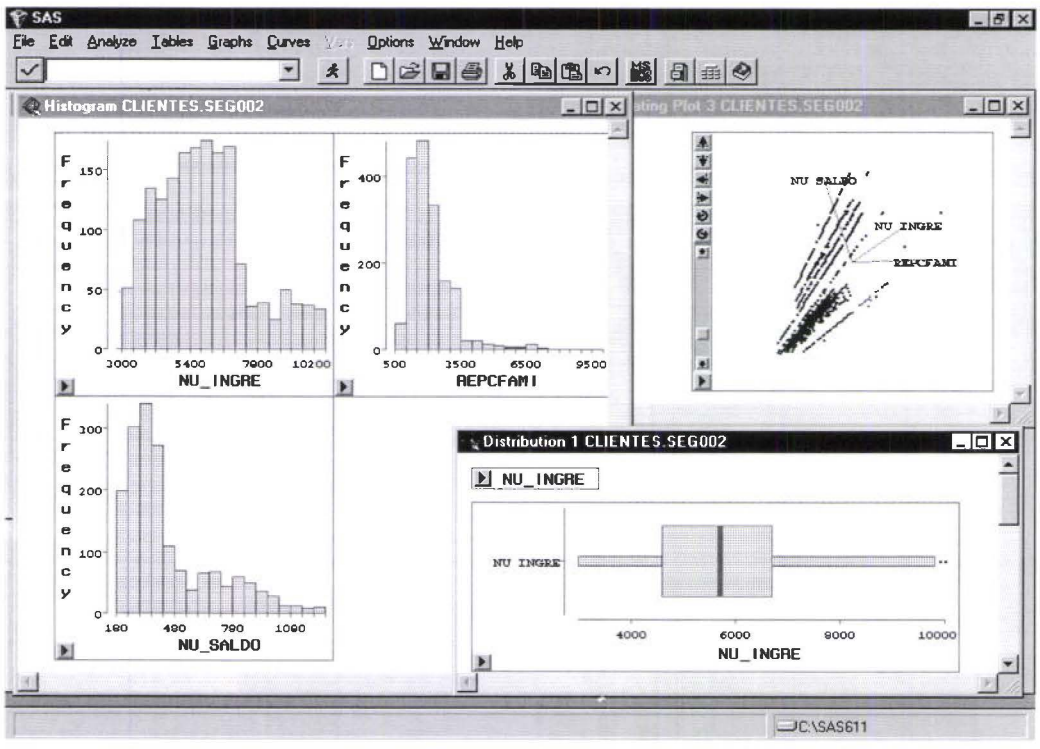


Gráfico 41.

El objetivo es simplificar en lo posible el problema con el fin de optimizar la eficiencia del modelo. En este paso se pueden emplear herramientas que nos permitan visualizar de forma gráfica la información utilizando las variables explicativas como dimensiones.

También se pueden emplear técnicas estadísticas que nos ayuden a poner de manifiesto relaciones entre variables. A este respecto resultará ideal una herramienta que permita la visualización y el análisis estadístico integrados.

Manipulación

Tratamiento realizado sobre los datos de forma previa a la modelización, en base a la exploración realizada, de forma que se definan claramente los inputs del modelo a realizar (selección de variables explicativas, agrupación de variables similares, etc.).

Modelización

Permite establecer una relación entre las variables explicativas y las variables objeto del estudio, que posibilitan inferir el valor de las mismas con un nivel de confianza determinado.

Valoración

Análisis de la bondad del modelo contrastando con otros métodos estadísticos o con nuevas poblaciones muestrales.

WEBHOUSING

La popularización de Internet y la tecnología Web, ha creado un nuevo esquema de información en el cual los clientes tienen a su disposición unas cantidades ingentes de información. La integración de las tecnologías Internet y Data Warehouse tienen una serie de ventajas como son:

Consistencia: toda la organización accede al mismo conjunto de datos y ve los informes que reflejan sus necesidades. Hay una "única versión de la verdad".

Accesibilidad: la empresa accede a la información a través de un camino común (el browser de Internet), simplificando el proceso de búsqueda de la información.

- Disponibilidad, la información es accesible en todo momento, independientemente de los sistemas operacionales.
- Bajos costes de desarrollo y mantenimiento, debidos a la estandarización de las aplicaciones de consultas basadas en Internet, independientemente del sistema operativo que soporte el browser, y de la reducción de los costes de distribución de software en los puestos clientes.
- Protección de los datos, debido al uso de tecnologías consolidadas de protección en entornos de red (firewalls).
- Bajos costes de formación, debido al uso de interfaces tipo Web.

La interactividad de las aplicaciones en este entorno pueden tener varios niveles:

- Publicación de datos: las páginas distribuyen información obtenida del Data Warehouse, volcada en las páginas intra/internet.

- Distribución de reportes: dando soporte a consultas simples elaboradas por los usuarios.
- Aplicaciones dinámicas: sirviendo de soporte de decisión a servicios solicitados desde el puesto cliente, ejecutando la petición en el servidor y devolviéndolas al cliente, vía el browser de Internet o haciendo uso de "*applets*" de Java.

Las arquitecturas base de una implantación de Data Warehouse en Internet, pueden tener las siguientes alternativas:

1. Usar el Servidor Internet como router, y ejecutar la petición desde el cliente al servidor directamente.
2. Hacer uso del navegador para visualizar una página Internet residente en el servidor de Internet. Esta página contendría información que se actualizaría en el servidor Internet, desde el servidor DW, a petición del usuario haciendo uso de CGI's.
3. El cliente podría lanzar su consulta directamente al servidor de DW, con "applets" de Java, haciendo el servidor Internet únicamente de encaminamiento (router).
4. El cliente podría ejecutar la aplicación DW desde el navegador, pero con un plug-in, que haría que se tuvieran las mismas opciones que la aplicación DW.
5. Realizar una descarga masiva de datos con un protocolo de transferencia de ficheros (FTP), para su proceso en local.

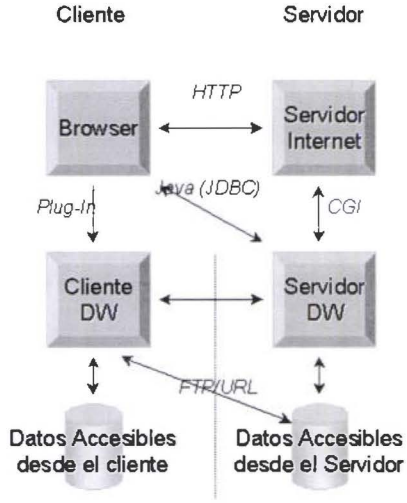


Gráfico 42.

El alcance funcional de la implantación del Data Warehouse, basado en tecnologías Internet, puede ser la misma que la realizada sin su uso. En este sentido las críticas que se le pueden achacar en la actualidad, provienen de la baja velocidad de las líneas actuales, que se solventa parcialmente mediante el uso de aplicaciones Java, en lugar de hacer uso de páginas HTML, o CGI. Solución parcial, mientras la velocidad de transferencia se incrementa día a día mediante nuevos algoritmos de compresión de datos o el uso de líneas de alta capacidad RDSI.

CREACIÓN DE LOS CUBOS MULTIDIMENSIONALES

Utilizando la tecnología multidimensional de Microsoft SQL Ser ver 7.0, los servicios OLAP, se presenta paso a paso la generación de los cubos multidimensionales del data warehouse para el sistema transaccional.

1. Iniciar el Administrador de OLAP, desde menú Inicio de Windows, Programas, Microsoft SQL Server 7.0, OLAP Services y escoja OLAP Manager.

Ahora, estamos listos para trabajar con el Administrador de OLAP, se debe configurar la estructura de la base de datos para ello son los siguientes pasos:

- En la vista de árbol del Administrador OLAP, expandir **OLAP servers**.
- Dar un clic en el nombre del servidor. Se establecerá una conexión con el servidor OLAP.
- Dar un clic derecho sobre el nombre del servidor (SUSANA), seleccionar **New Database.**
- En la caja de dialogo **Database**, digitar el nombre de la base de datos del data warehouse (Dawaho) y clic en OK.
- En el árbol del Administrador OLAP, expandir el Servidor y entonces expandir la base **Dawaho** creada.

La figura siguiente ilustra la conexión creada, la base de datos Dawaho contiene tres carpetas que son: **Cubes, Virtual Cubes y Library.**

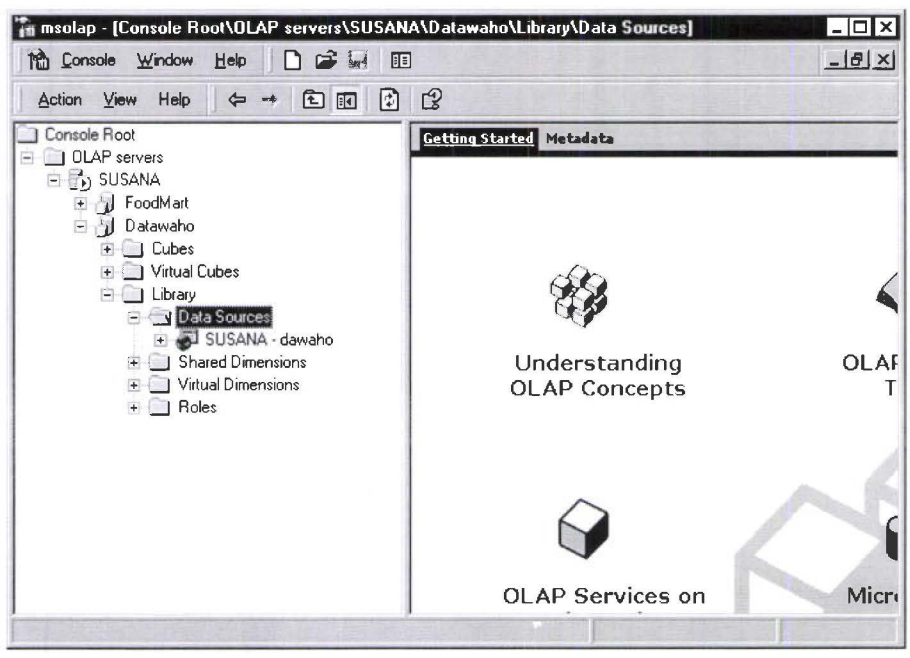


Gráfico 43

A continuación usaremos el Asistente para creación de cubos, para ello se debe hacer lo siguiente:

- En el Administrador OLAP, bajo la base de datos Dawaho, dar un clic derecho sobre la carpeta Cubes, escoger Nuevo Cubo y entonces escoger el Wizard desde el menu.
- Lo que sigue es añadir medidas a el cubo, seleccionar la tabla para el cubo, en nuestro caso seleccionamos **log_cabecera_his**. Gráfico 44.
- Se debe definir las medidas para el cubo bajo las columnas numéricas, seleccionar las columnas requeridas y darles un nombre, si se desea, en nuestro caso se han creado 3 medidas: **Monto total, efectivo y documentos**, estas representan los montos de las transacciones financieras. Ver gráfico 45.

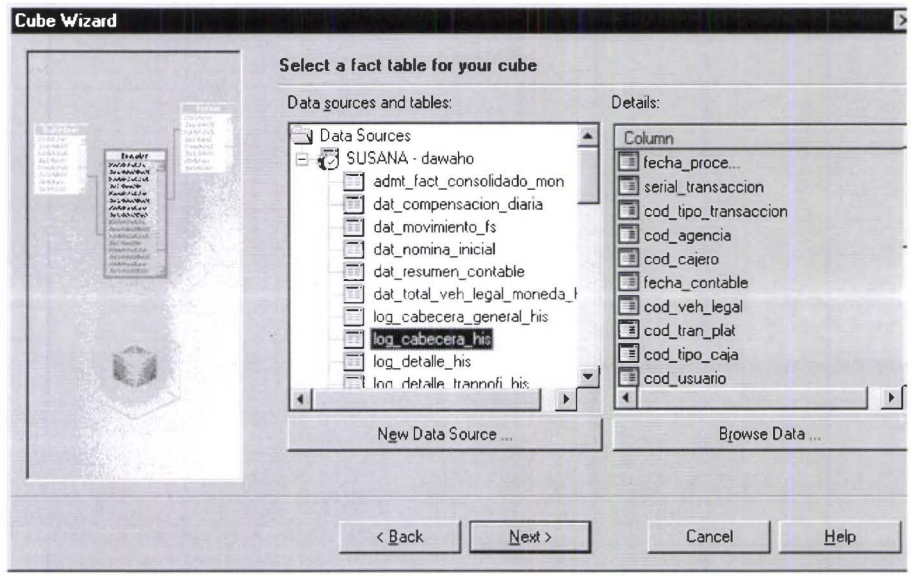


Gráfico 44.

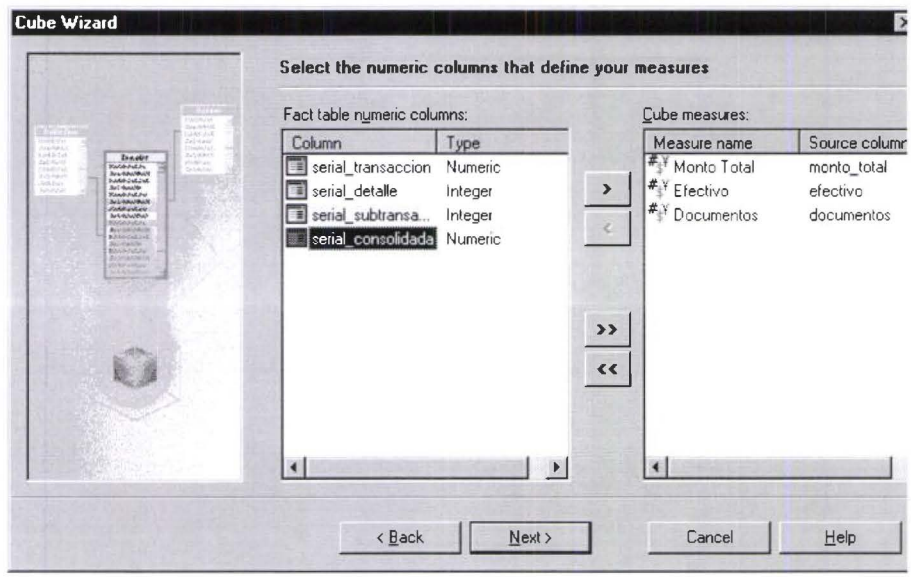


Gráfico 45.

El siguiente paso es la creación de las dimensiones para nuestro cubo, en la pantalla de Selección de dimensiones para el cubo, dar un clic en **New Dimension**.
Una de las primeras dimensiones que se deben crear es la dimensión de tiempo, para lo cual seleccionamos la columna fecha de la tabla log_cabecera_his, seguidamente aparecerá el asistente para creación de dimensión de tiempo, gráfico 46.

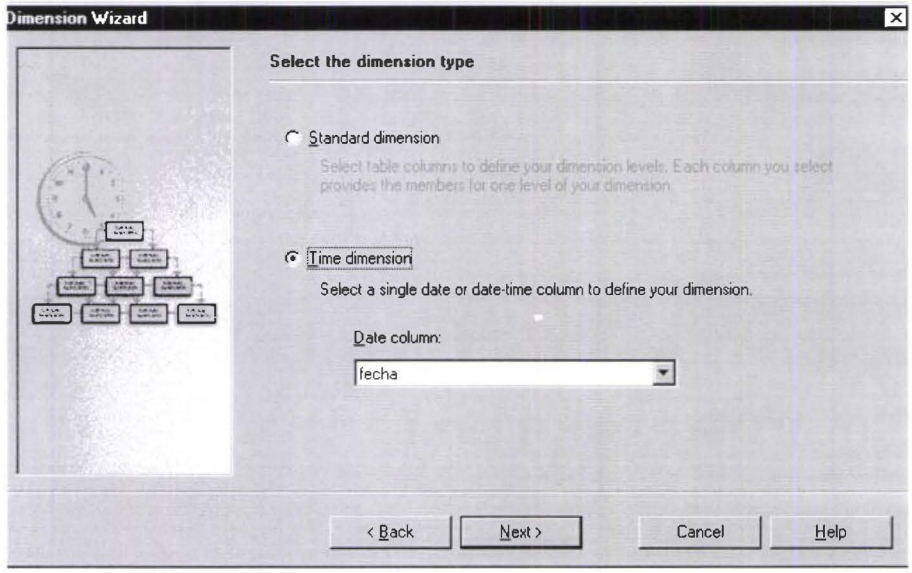


Gráfico 46.

Para seleccionar los niveles para esta dimension, dar un clic en la lista despleglable en Select time levels, escoger el nivel adecuado a las necesidades, en este caso se escogerá Year, Quarter, Month, Day, Hour, Minute para satisfacer el nivel de detalle del negocio. Este procedimiento se muestra en los Gráficos 47,48,49 y 50.

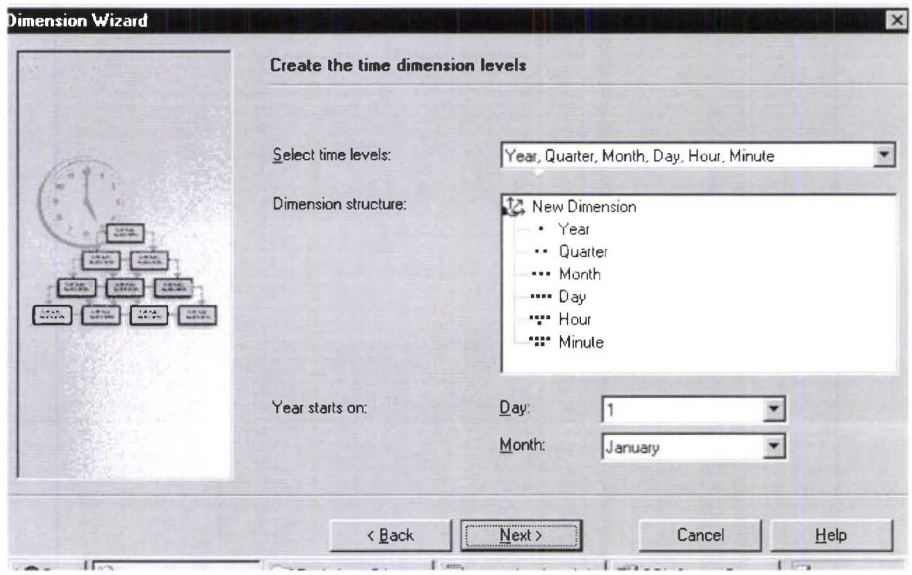


Gráfico 47.

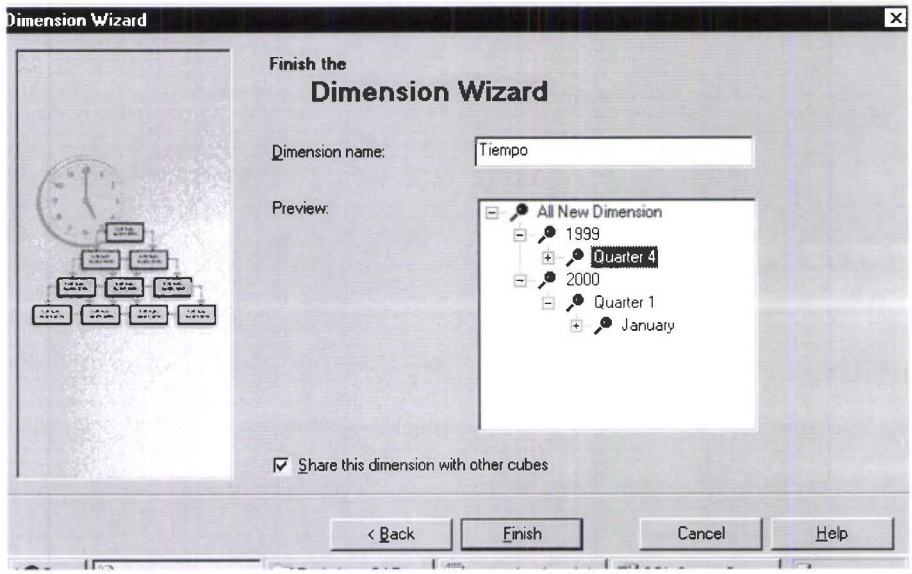


Gráfico 48.

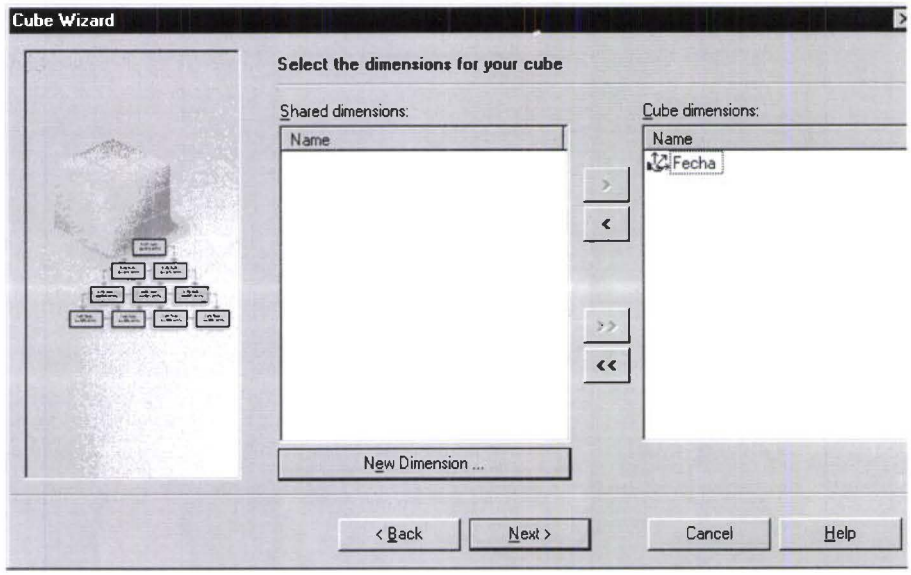


Gráfico 49.

Luego de haber generado las dimensiones necesarias para el cubo, presionamos el botón Finish. Se abrirá el Editor de Cubos, en el se puede mirar la tabla de hechos (con una barra de titulo amarilla) y las relaciones con las tablas de dimensiones (una barra de titulo azul). En la vista de árbol del editor de cubos, se puede ver previamente el cubo en un árbol jerárquico. Se pueden editar las propiedades del cubo haciendo clic en el botón **Properties**.

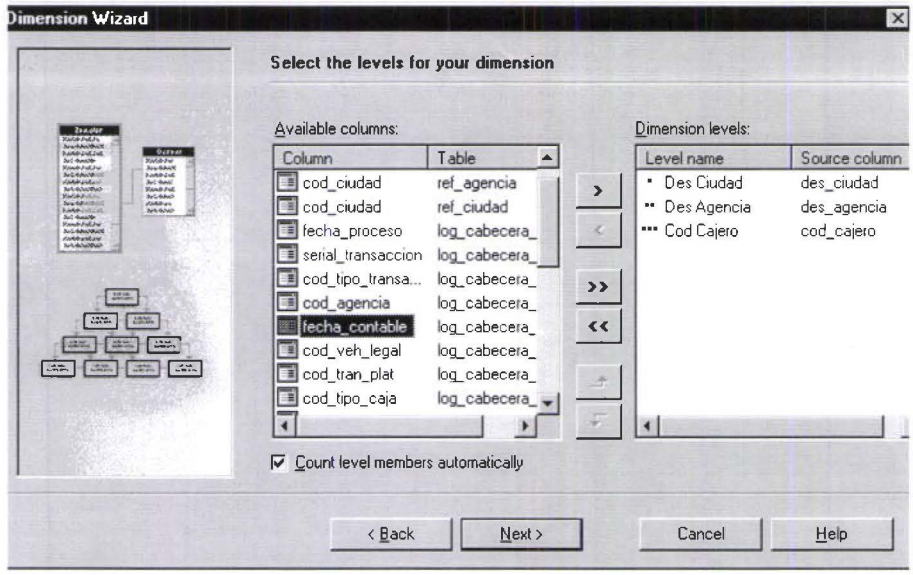


Gráfico 50.

En el gráfico 51 se presenta la tabla de hechos log_cabecera_his y la relación con la tabla ref_agencia, que contiene la dimensión agencia.

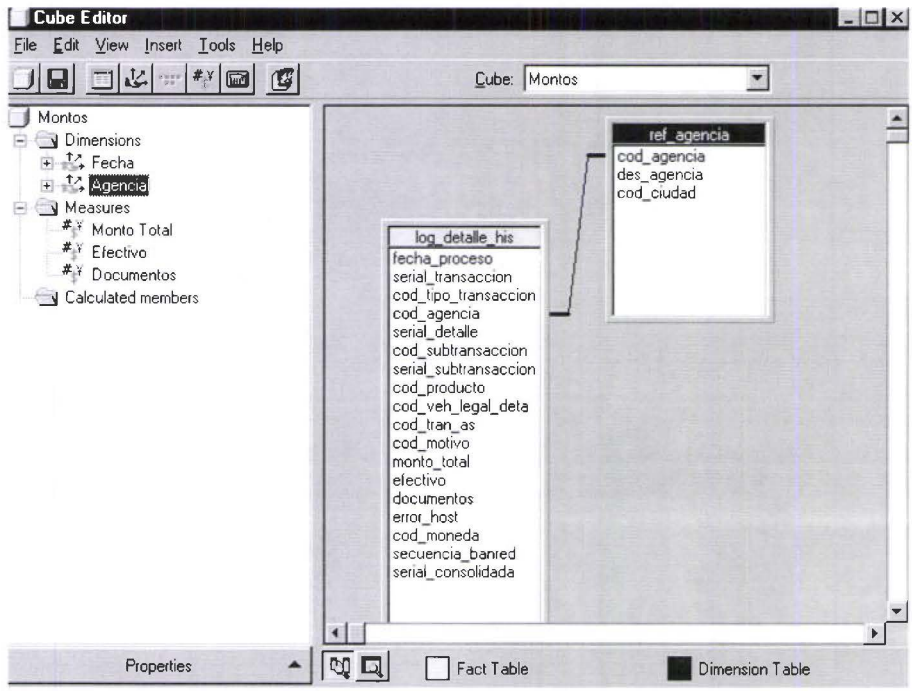


Gráfico 51.

El siguiente paso es el almacenamiento y procesamiento del cubo para ello se usa el Storage Design Wizard para especificar la manera como OLAP Services almacena las agregaciones y optimizar el rendimiento del procesamiento de consultas para el cubo.

En el Editor del cubo, desde el menú Tools, escoger Design Storage. Seleccione una de las tres opciones que se presentan en la ventana; para nuestro caso hemos escogido MOLAP, como se muestra en el gráfico 52.

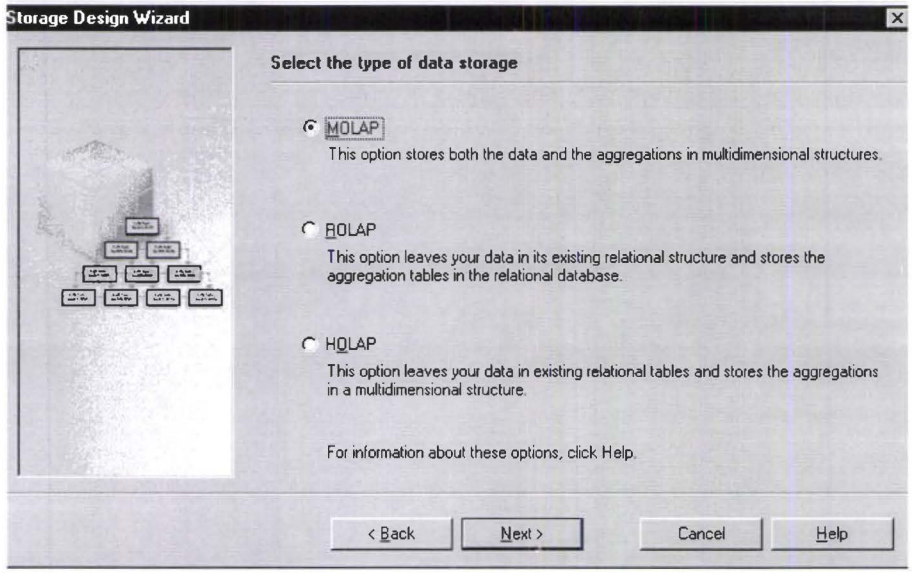


Gráfico 52.

En la siguiente pantalla se indicará las Opciones de agregación, se puede escoger de entre tres opciones: **Estimated storage reactives, Performance gain reaches** y **Until clic stop**. Los administradores pueden usar este afinamiento para balancear las necesidades para el rendimientos de las consultas contra el espacio en disco requerido para almacenar los datos agregados. En nuestro caso, como se presenta en el gráfico 53, se seleccionó **Estimated storage reaches**.

Al dar un clic en Start se puede observar el gráfico **Performance vs. Size**, es decir el rendimiento versus el tamaño de almacenamiento de los datos agregados.

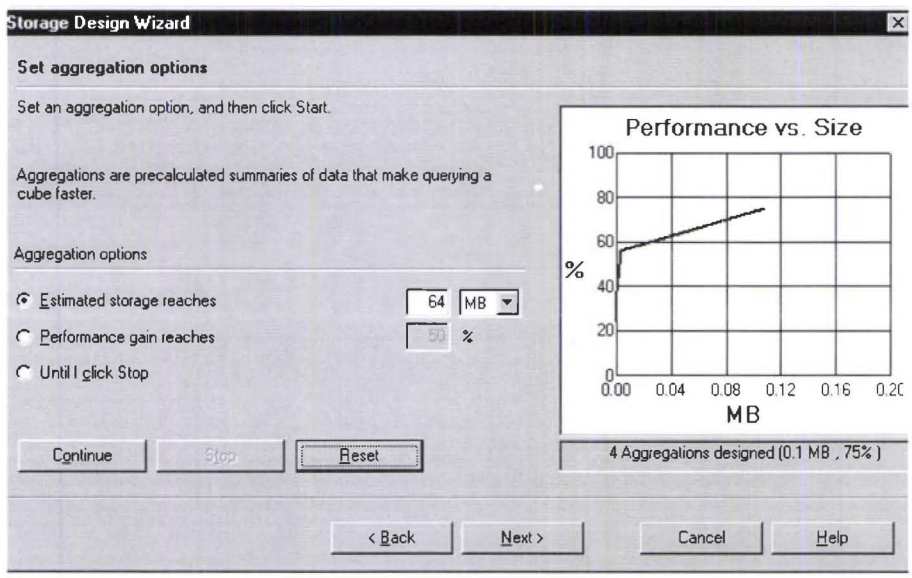


Gráfico 53.

Cuando el proceso de diseño de agregaciones este completo dar un clic en **Next**, seleccionar **Process Now** y luego dar clic en **Finish**. Una ventana aparecerá en la se puede observar tal como el cubo esta siendo procesado. Cuando el proceso está completo, aparecerá un mensaje, **Processing Completed Successfully**. Dar un clic en close y retornara al Administrador OLAP. Este proceso se lo puede observar en el gráfico 54.

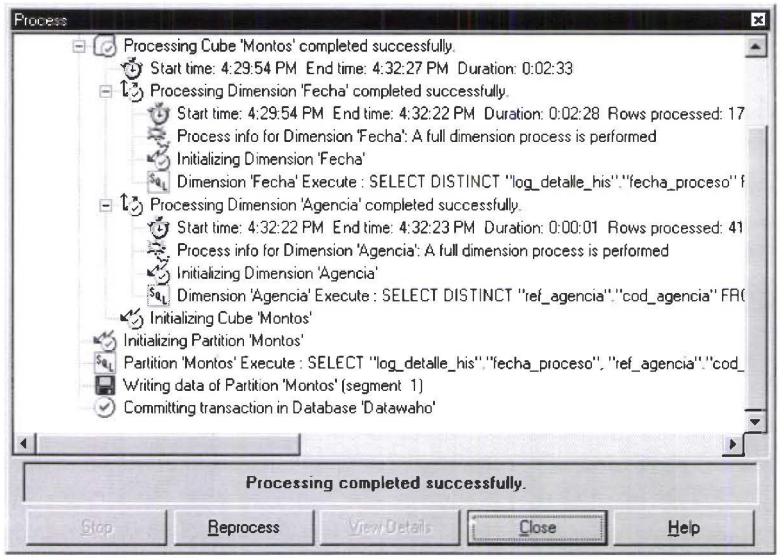


Gráfico 54.

Como se puede apreciar en la figura, bajo el árbol, en la carpeta Cubes se ha creado el cubo Montos, en el panel de la derecha en Metadata, se presenta toda la información del cubo generado.

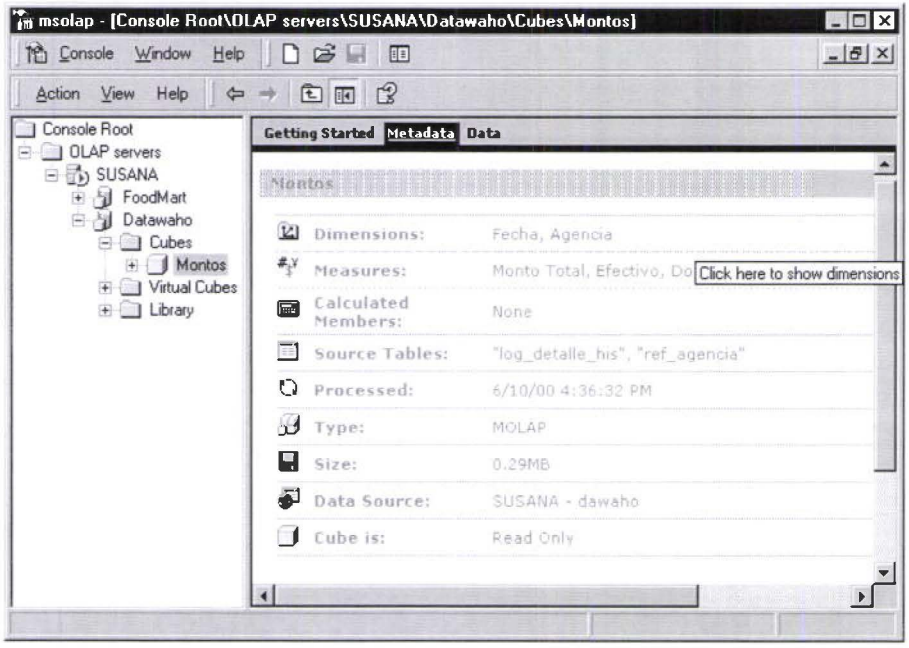


Gráfico 55.

No una parte importante que no se nos puede escapar es el definir Roles para el cubo, de tal manera que los usuarios o grupo de usuarios puedan consultar los datos en un cubo. Ahora que el cubo está creado (también se lo puede realizar en la etapa de diseño del cubo, en el editor de cubos), bajo la carpeta Library, dar un clic derecho sobre la carpeta Roles y seleccionar New Role, como se indica en el gráfico 56.

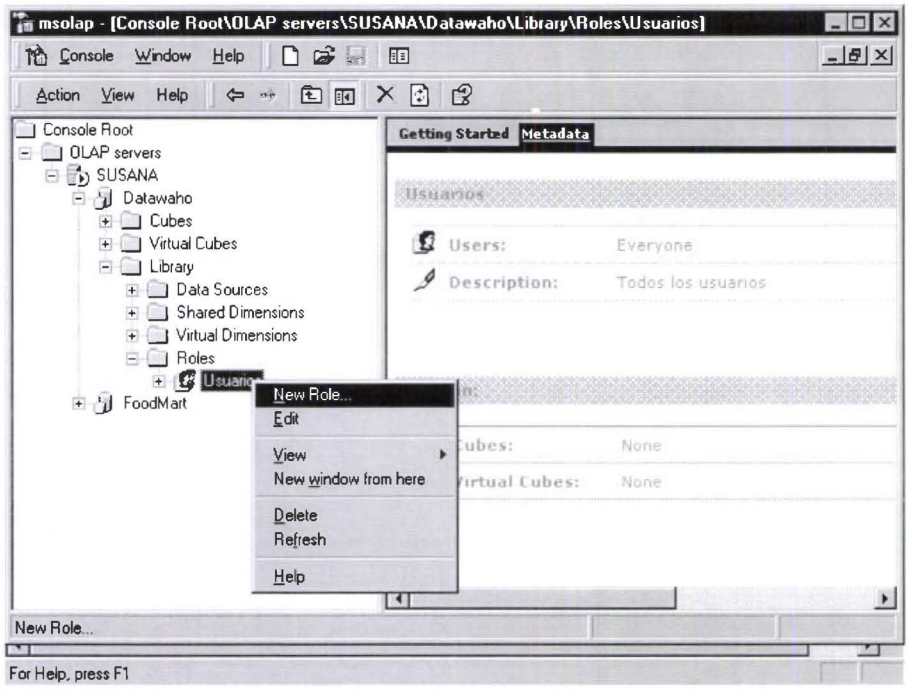


Gráfico 56.

A continuación se solicita el nombre para el nuevo rol, una descripción y seleccionamos los grupos y usuarios que formaran parte de este rol, para ello dar un clic en Groups and Users... como se muestra en el gráfico.

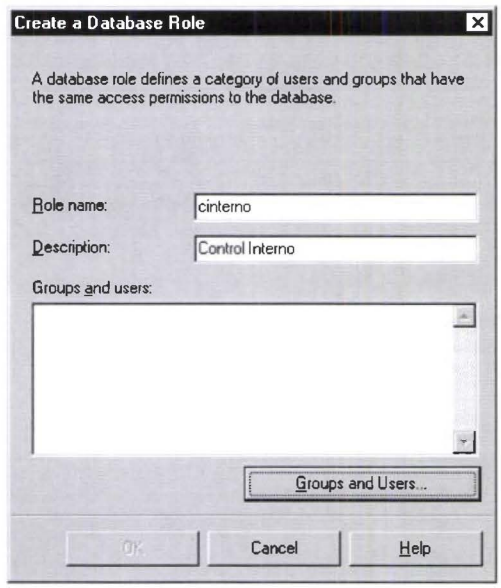


Gráfico 57.

Aparcera la lista de usuarios y grupos del dominio o dominios NT en el que estamos trabajando, añadimos los necesarios y presionamos OK.

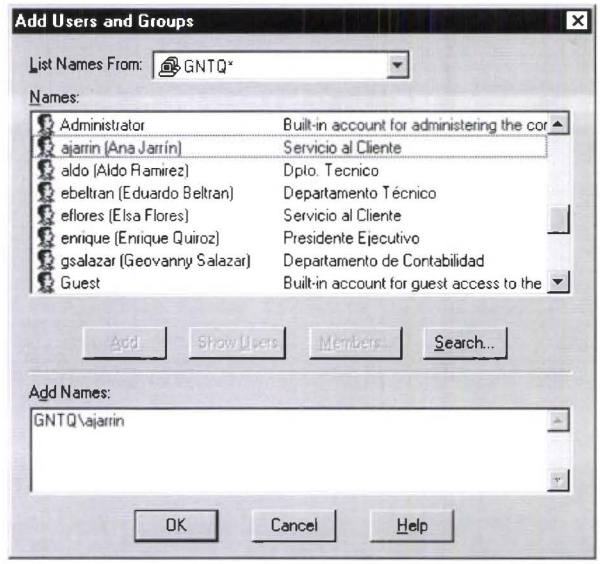


Gráfico 58.

Una vez creados los roles, se los asignará a cada uno de los cubos que se hayan creado, de acuerdo a las necesidades, en el gráfico 59, se observa dos roles creados Usuarios y cinterno. Para añadir el o los roles a los cubos se usa el administrador de roles para cubos, para ello bajo el cubo deseado, clic derecho sobre la carpeta roles, escoger **Manage Roles..** ,mirar gráfico 60.

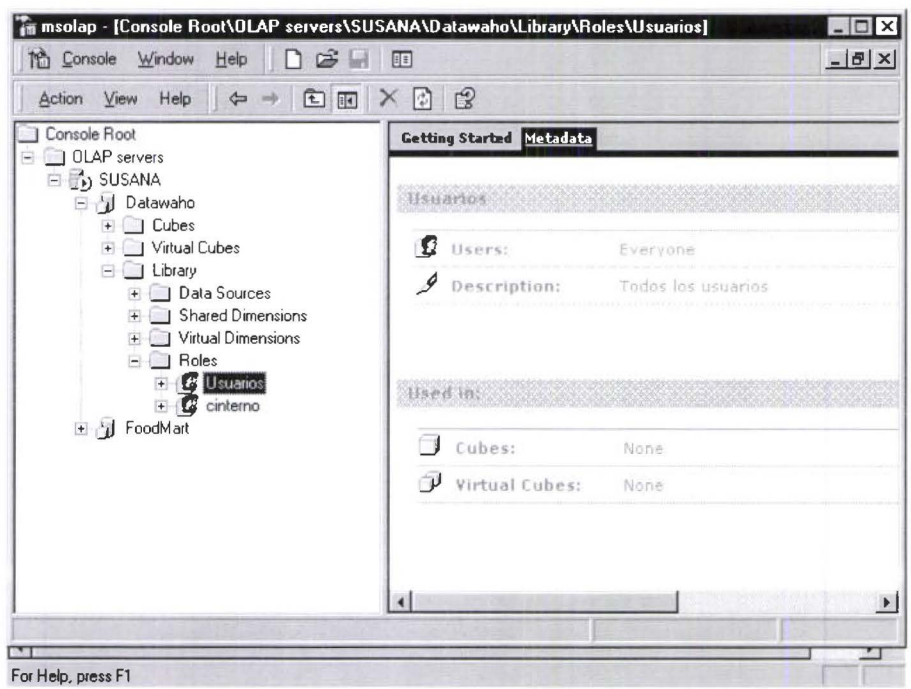


Gráfico 59.

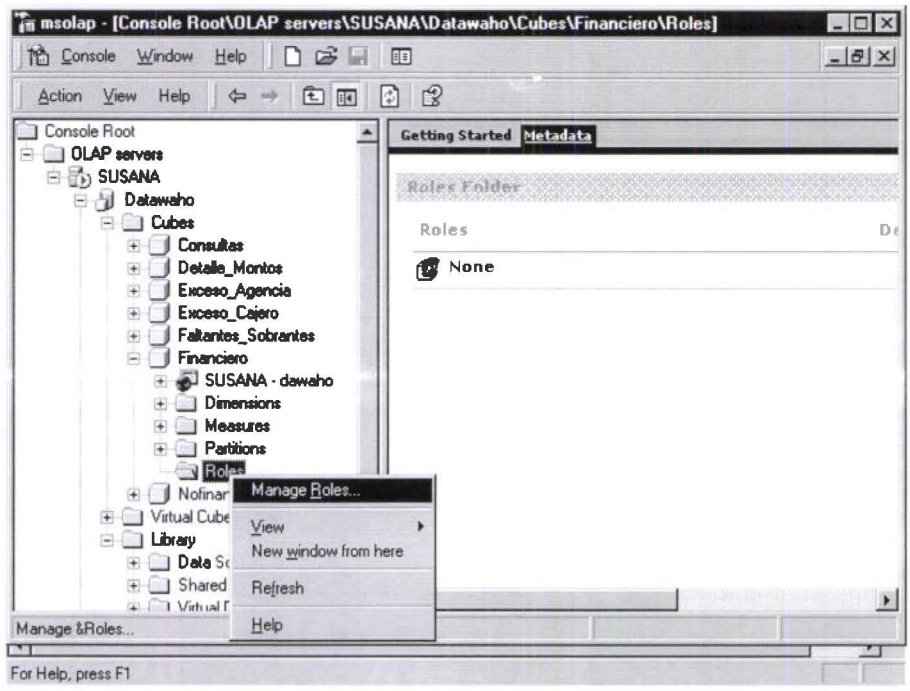


Gráfico 60.

En el **manage the cube roles** se puede seleccionar el rol y pasarlo como parte de los roles del cubo mediante la fecha >, este proceso se lo puede visualizar en los gráficos 61 y 62.

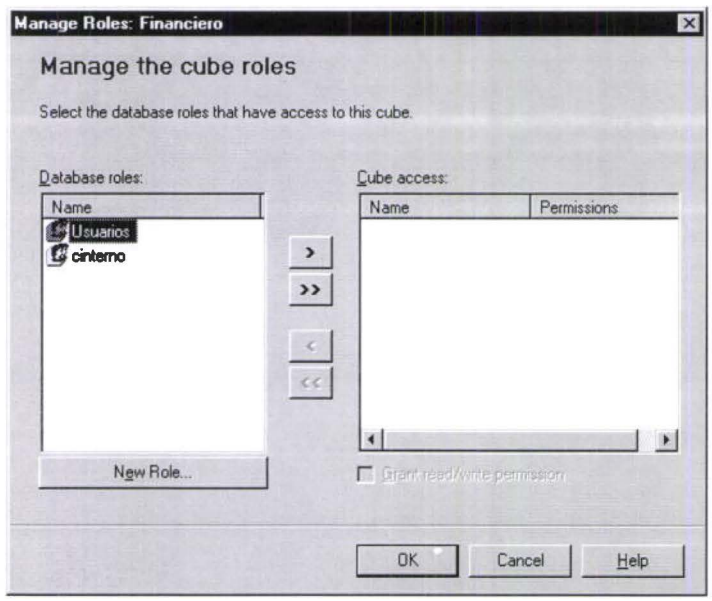


Gráfico 61.

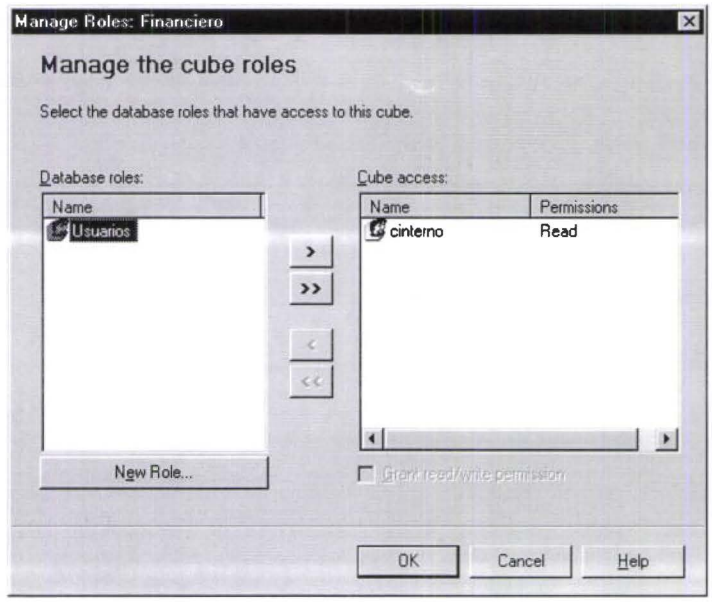


Gráfico 62.

Ahora ya se pueden visualizar los datos de los cubos. Usando el **Cube browser**, se puede mirar los datos de diferentes maneras, se puede filtrar la cantidad de dimensiones que están visibles y se puede mirar en diferentes grados de detalle.

Hasta aquí se ha presentado la creación de los cubos multidimensionales, utilizando las herramientas de Microsoft SQL Server 7.0 y específicamente OLAP Services, de una manera rápida y sencilla, pero existen muchas más opciones y actividades ha realizar que no son parte de este documento y que los interesados en ahondar en este tema lo pueden hacer, utilizando las facilidades que brinda la herramienta y los asistentes de OLAP Services.

A continuación se presenta como utilizar la herramienta Microsoft Excel 2000 para realizar las consultas de una manera sencilla y práctica.

CONSULTAS DEL DATA WAREHOUSE MEDIANTE MICROSOFT EXCEL 2000

Luego de haber generado los cubos multidimensionales, estamos listos para realizar las consultas del data warehouse, utilizando una herramienta cliente, en este caso especifico optamos por Microsoft Excel 2000.

A continuación, se presenta una guía, paso a paso, para realizar la conexión al data warehouse y visualizar la información multidimensional.

2. Abrir Microsoft Excel 2000, se presentará una pantalla como la siguiente:

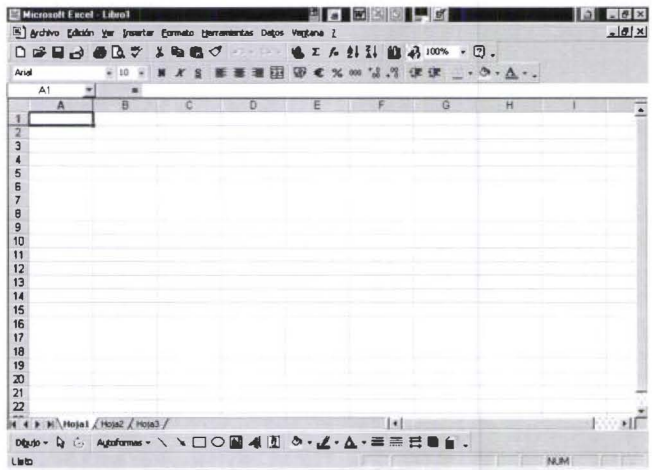


Gráfico 63.

3. Luego, en el menú escoger la opción **Datos, Informe de tablas y gráficos dinámicos...**, esto inicia el asistente para tablas y gráficos dinámicos que le guiará a través de la creación o de la modificación de un informe de tablas o de gráficos dinámicos, como se indica en la figura.

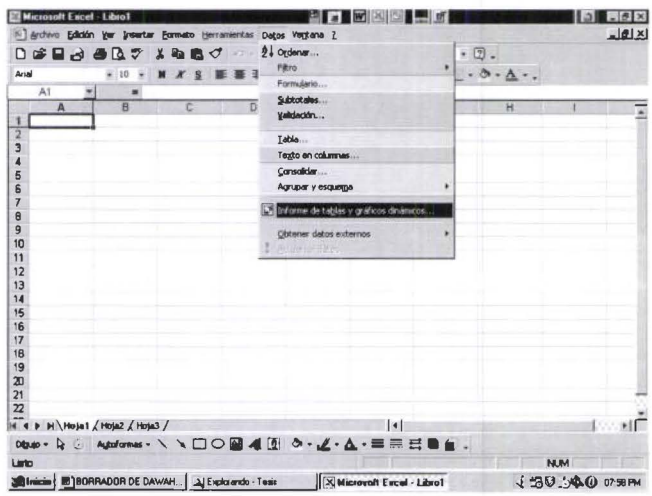


Gráfico 64.

4. En la ventana que aparecerá se debe elegir un origen de datos para el informe de tabla dinámica o gráfico dinámico. Una vez creado el informe, los datos pueden provenir de diversas fuentes. Así como el tipo de informe que se desea crear. Estas opciones se muestran en la figura siguiente:

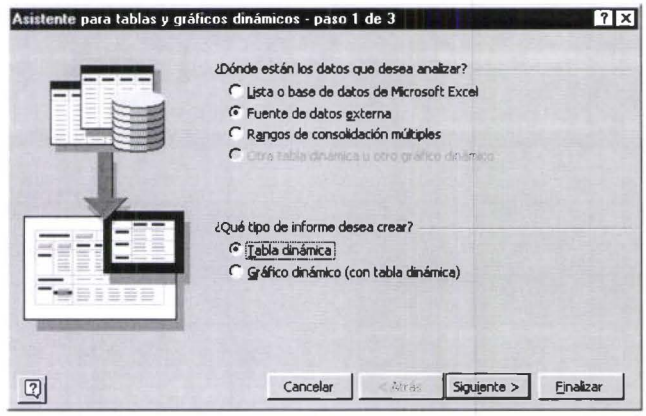


Gráfico 65.

5. Al presionar el botón siguiente de la pantalla anterior, se presenta una pantalla para indicar donde están guardados los datos externos, para lo cual damos un clic en **Obtener datos...**

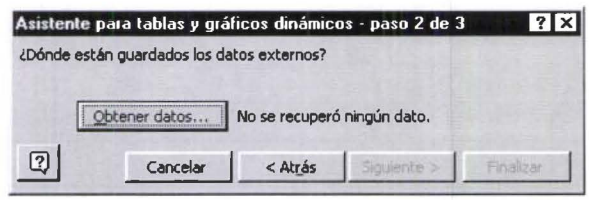


Gráfico 66.

6. Se debe seleccionar la pestaña **Cubos OLAP**, de la ventana que se muestra a continuación:

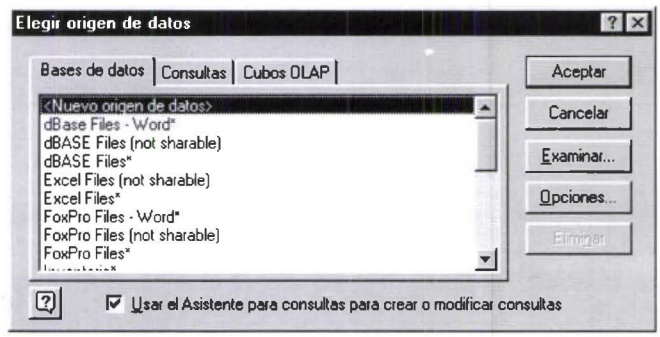


Gráfico 67.

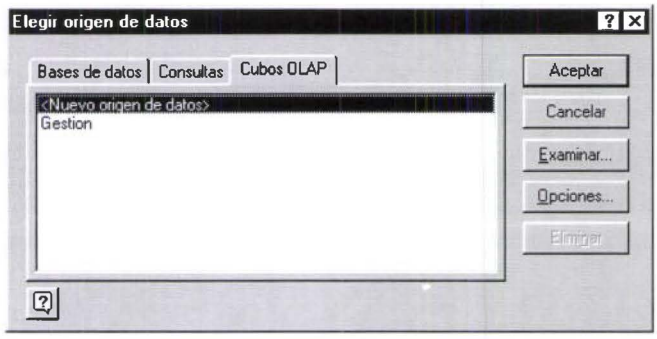


Gráfico 68.

7. Si el origen de datos que busca no esta definido haga doble clic en **<Nuevo origen de datos>** para que el asistente nos vaya solicitando la información necesaria para establecer el origen de datos, la figura muestra esta ventana:

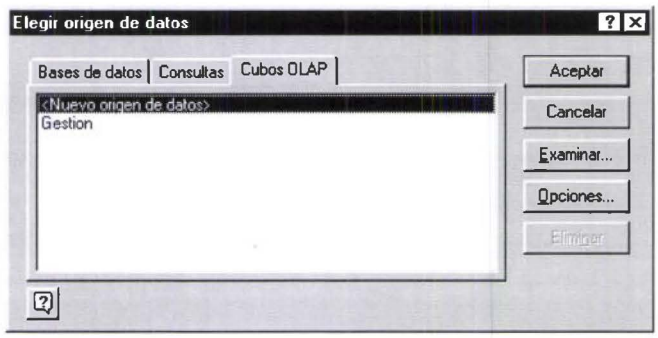


Gráfico 69.

Se requiere cuatro ingresos de información, como se puede apreciar en la figura 70:

1) Escriba un nombre que describa el origen de datos OLAP, se usará este nombre para el origen de datos cada vez que lo utilice para crear una consulta.
2) Seleccione el proveedor OLAP que desee utilizar, se seleccionará el tipo de base de datos o de servidor OLAP para los datos. Si el servidor no aparece en la lista, es posible que tenga que instalarlo.
3) Introduzca la información que necesita el proveedor acerca de la base de datos, se le pedirá que proporcione la ubicación del cubo OLAP y cualquier otro tipo de información necesaria para el proveedor. Si fuese necesario para obtener más información sobre el cubo, consulte al administrador de base de datos. La información solicita aparece en la figura 49 y 50.
4) Seleccione, de la lista, el cubo que contiene los datos que busca

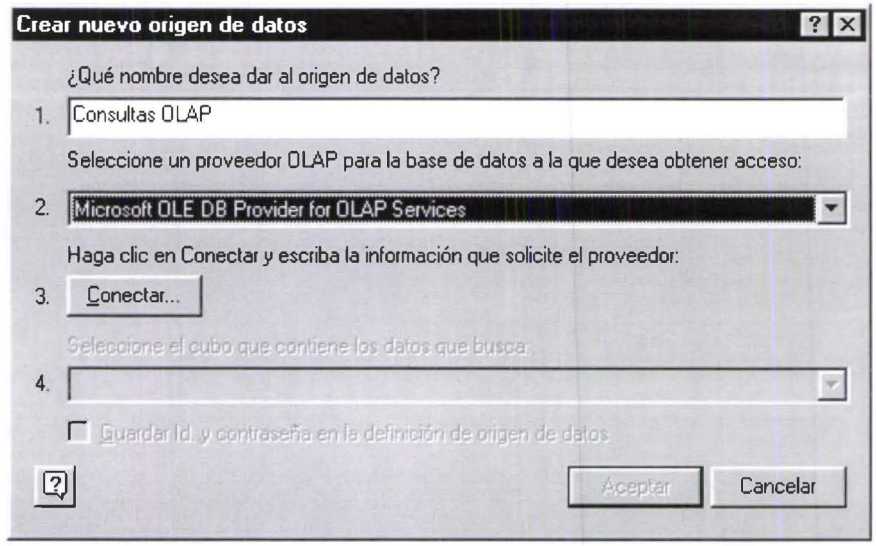


Gráfico 70.

Se puede activar la opción de guardar el Id y contraseña en la definición de origen de datos, dependiendo de las seguridades implementadas.

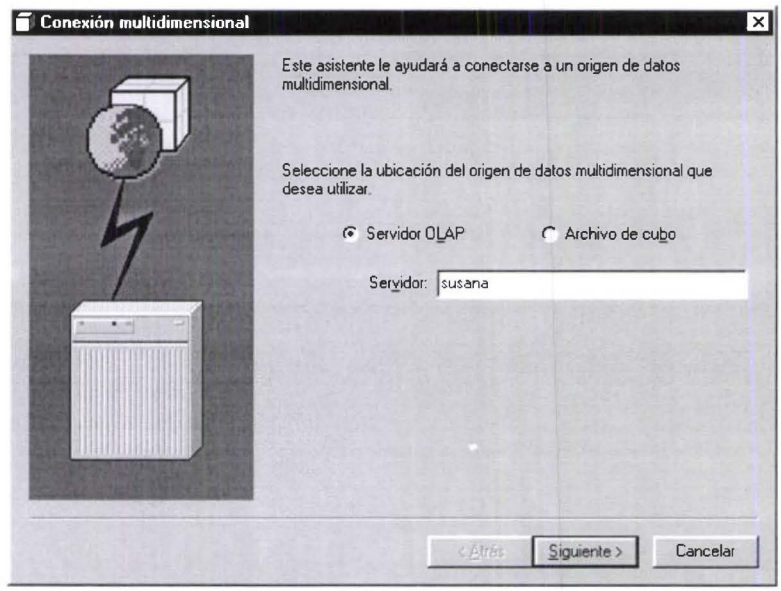


Gráfico 71.

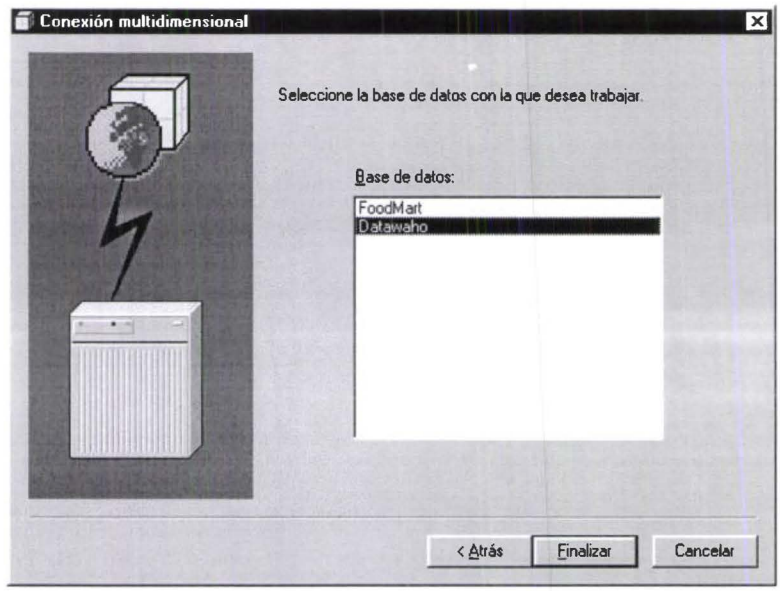


Gráfico 72.

En la figura 73 se muestra el origen de datos con toda la información que fue solicitada por el asistente.

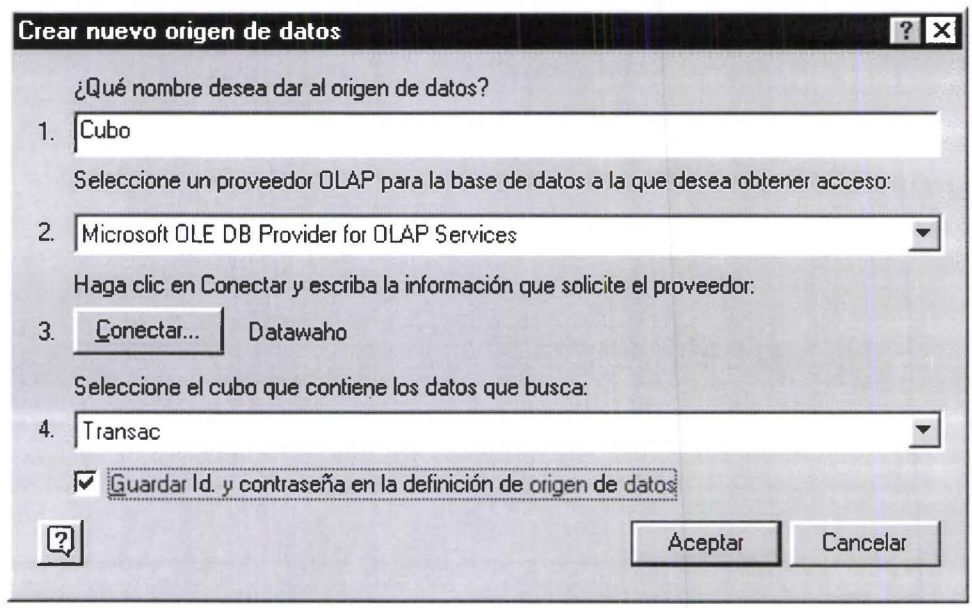


Gráfico 73.

8. En el siguiente paso del asistente se indica donde situar la tabla dinámica, ver figura 74 , se tiene dos opciones: en una hoja de calculo nueva o en una existente. Al presionar el botón Finalizar habremos concluido con la creación de la tabla dinámica.

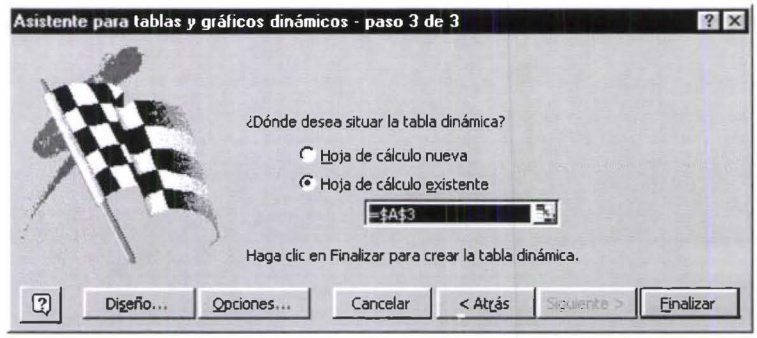


Gráfico 74.

9. Cuando se ha finalizado la creación de la tabla dinámica, se desplegará una pantalla como la que aparece a continuación, donde aparecen las dimensiones del cubo que seleccionamos en el paso 6.

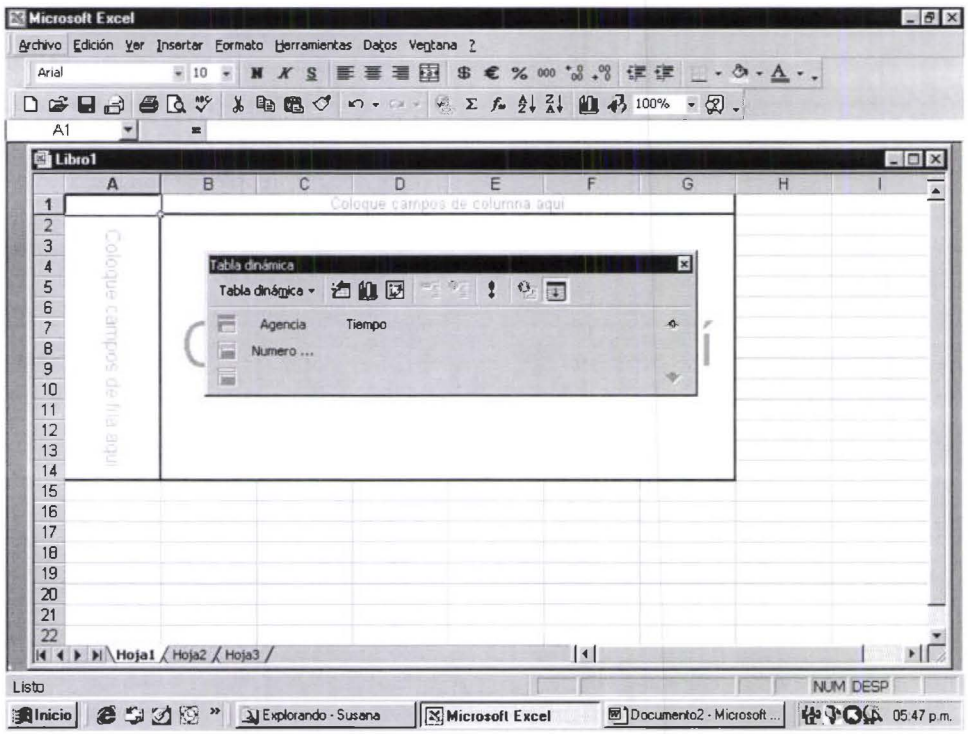


Gráfico 75.

Para diseñar el reporte, desde el grupo de botones de campo, arrastre los campos que desee a las áreas de FILA y de COLUMNA del diagrama. [14]

En el siguiente gráfico se puede observar el informe que resultaría al seleccionar algunas de las dimensiones del cubo que estamos consultando.

[14] Para mas información acerca de Diseño del reporte remitirse a Ayuda de Microsoft Excel 2000

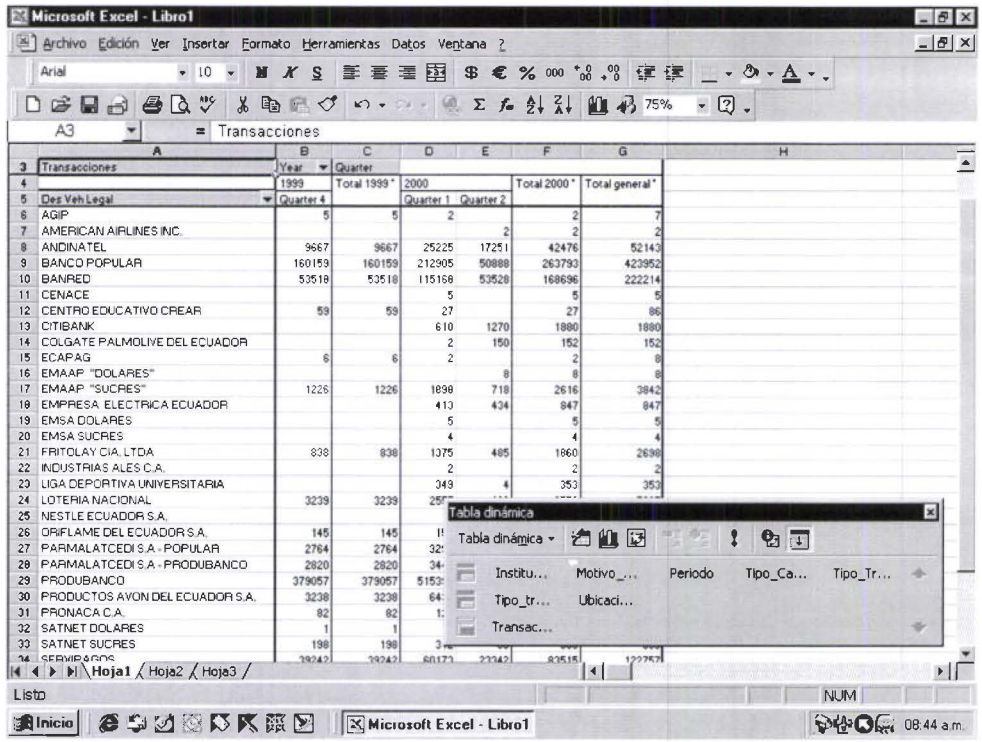


Gráfico 76.

En el gráfico 76 se presenta las transacciones por años y trimestres detallados por institución. A este reporte se ha agregado la dimensión de Ubicación y se genera un reporte como el que se muestra en el gráfico 77.

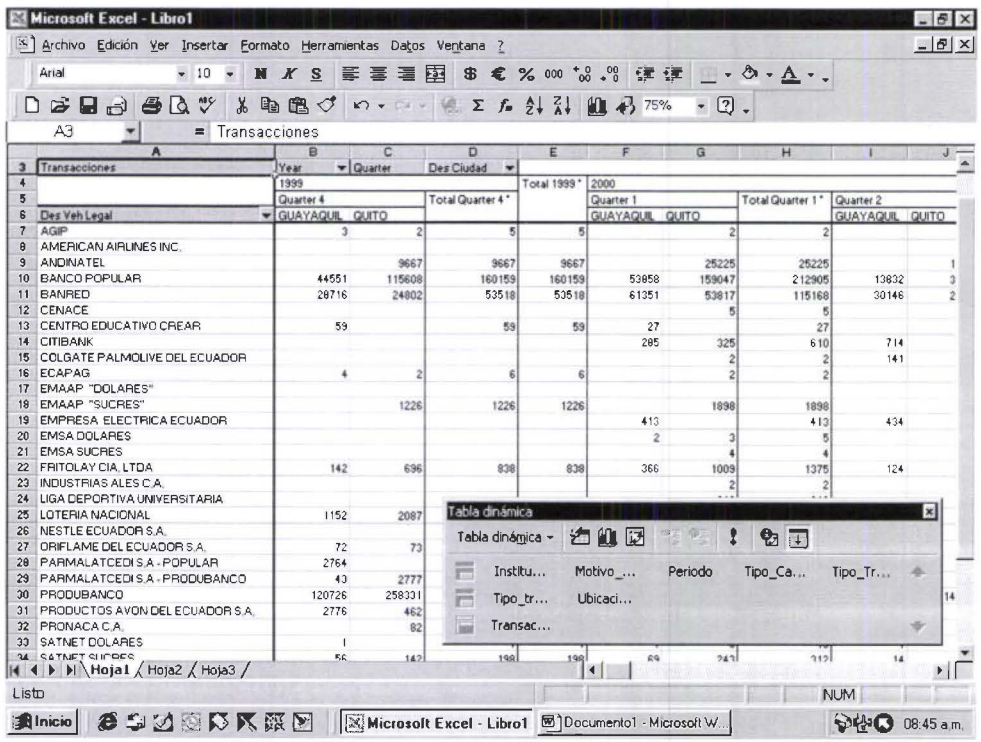


Gráfico 77.

De acuerdo al nivel de detalle y complejidad requerido para el reporte se puede seguir agregando dimensiones disponibles del cubo seleccionado como se muestra en el gráfico 78.

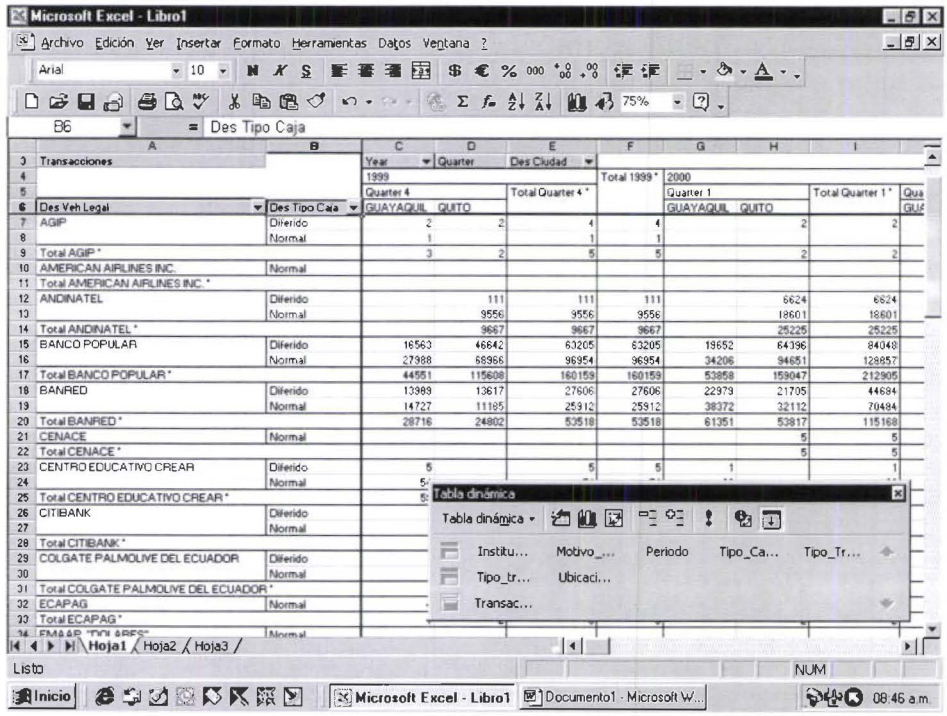


Gráfico 78.

Excel 2000 facilita la generación de un gráfico del reporte multidimensional, para ello seleccionamos toda el área del reporte y seleccionamos el botón de la barra de herramientas para el asistente de gráficos como se presenta en el gráfico 79.

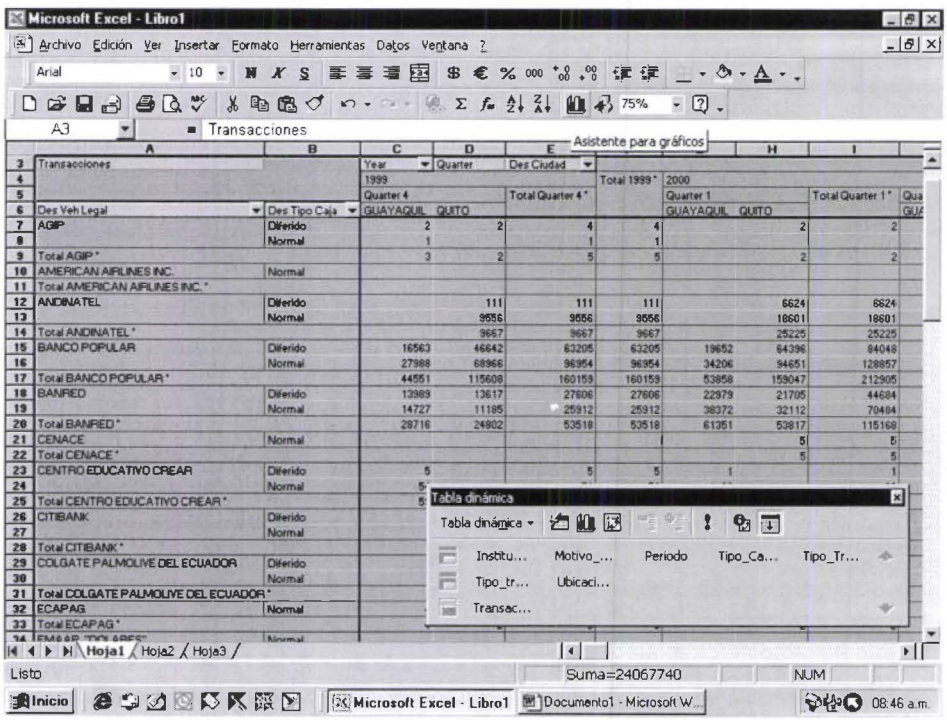


Gráfico 79.

En los informes de gráfico dinámico, los elementos de campo de fila se trazan en el eje de categorías y los elementos de campo de columna se trazan como series de datos. El gráfico generado a partir del informe anterior es el siguiente:

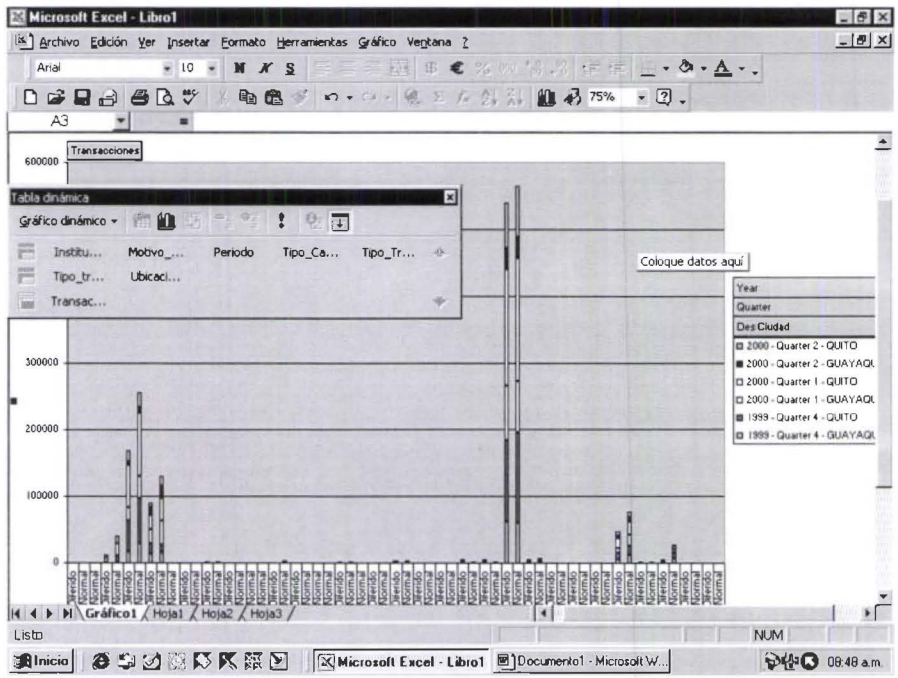


Gráfico 80.

Al igual que en el informe en forma de tabla dinámica, en el informe gráfico que estén basados en datos de origen procedentes de bases de datos OLAP, la flecha de campo aparece en el campo de nivel más alto de una dimensión y puede seleccionar elementos en diferentes niveles del campo. Haciendo clic en la flecha de la lista desplegable de un campo Año, como se aprecia en el gráfico a continuación:

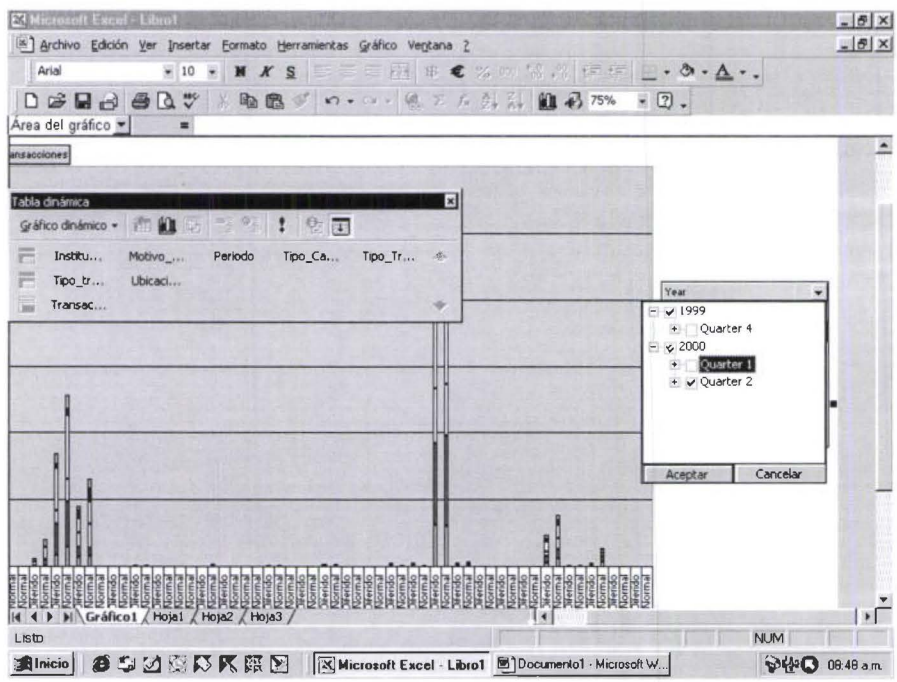


Gráfico 81.

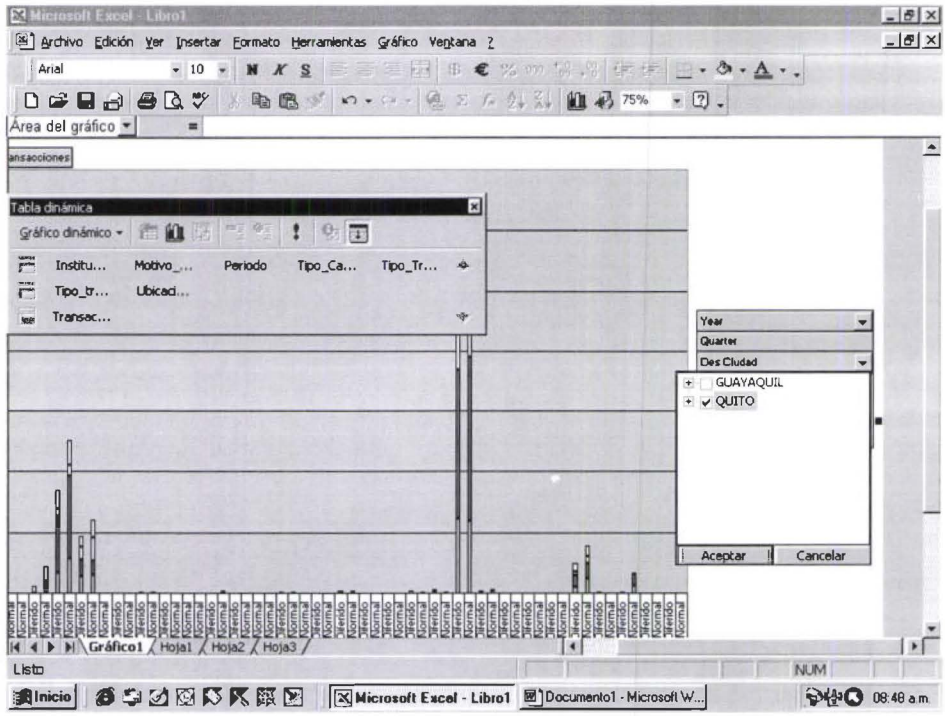


Gráfico 82.

Es muy sencillo seguir añadiendo o quitando dimensiones y detalles par el informe solo con arrastrar y seleccionar en niveles de árbol como se muestra en la figura 83 y 84.

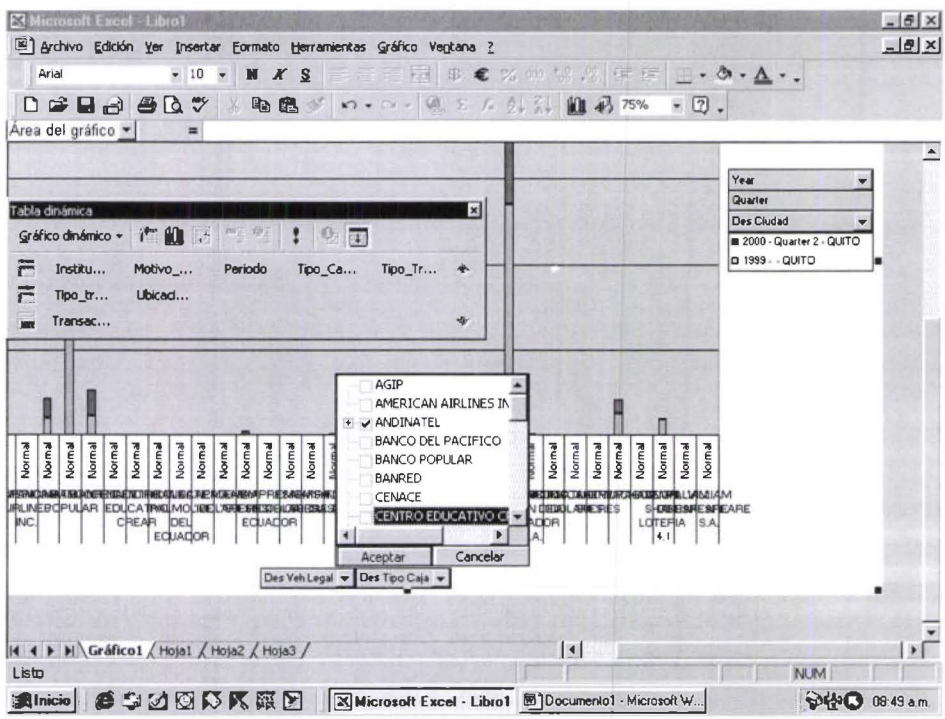


Gráfico 83.

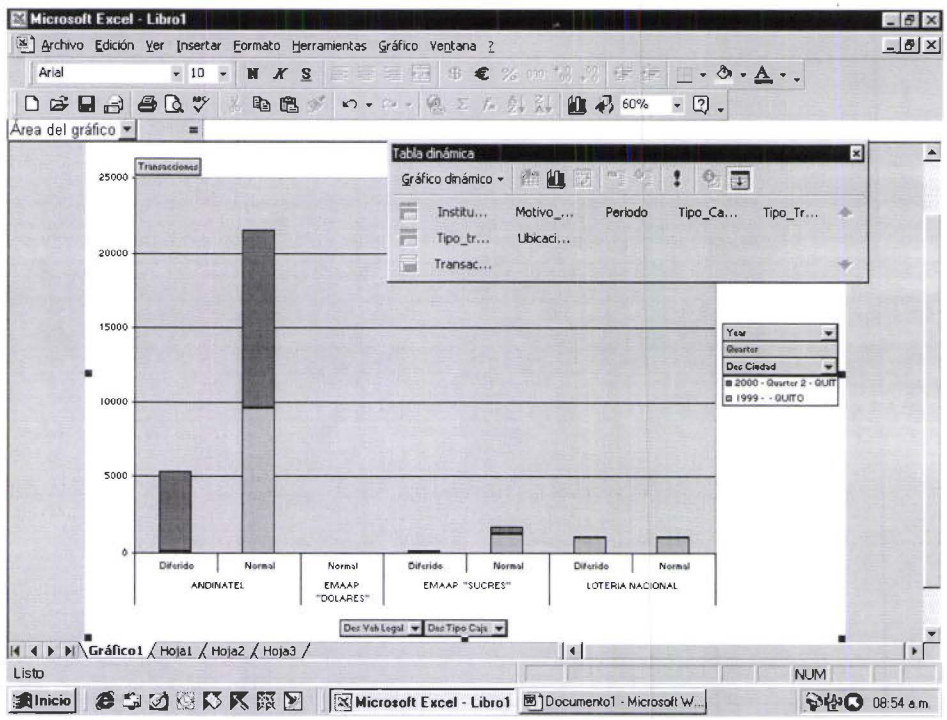


Gráfico 84.

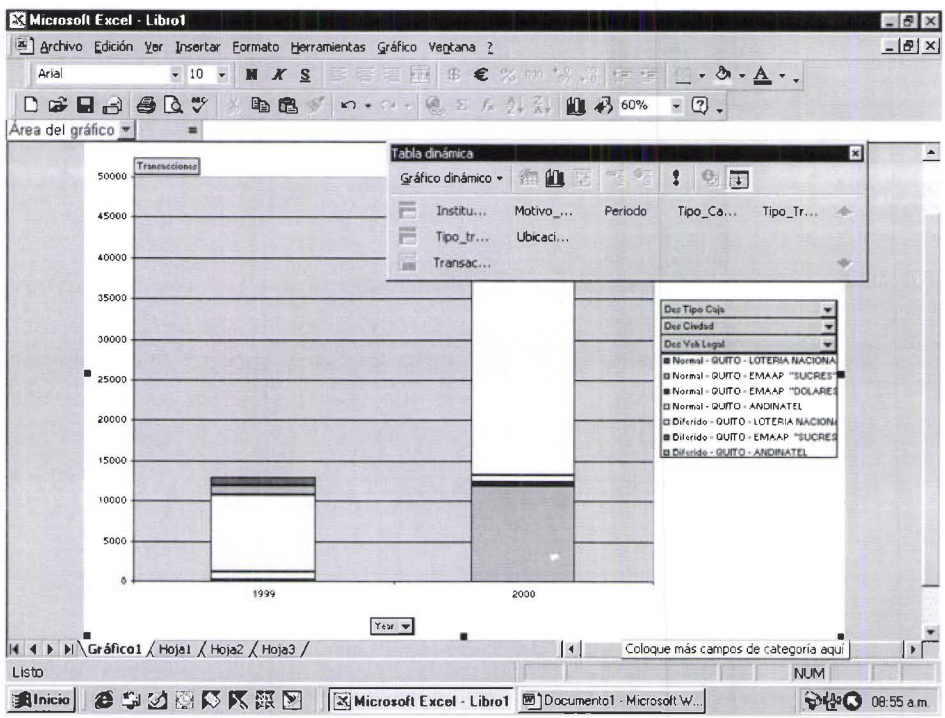


Gráfico 85.

Como cualquier gráfico de Excel se puede cambiar sus propiedades y seleccionar una presentación alternativa que se ajuste más a las necesidades del usuario. Gráfico 86 y 87.

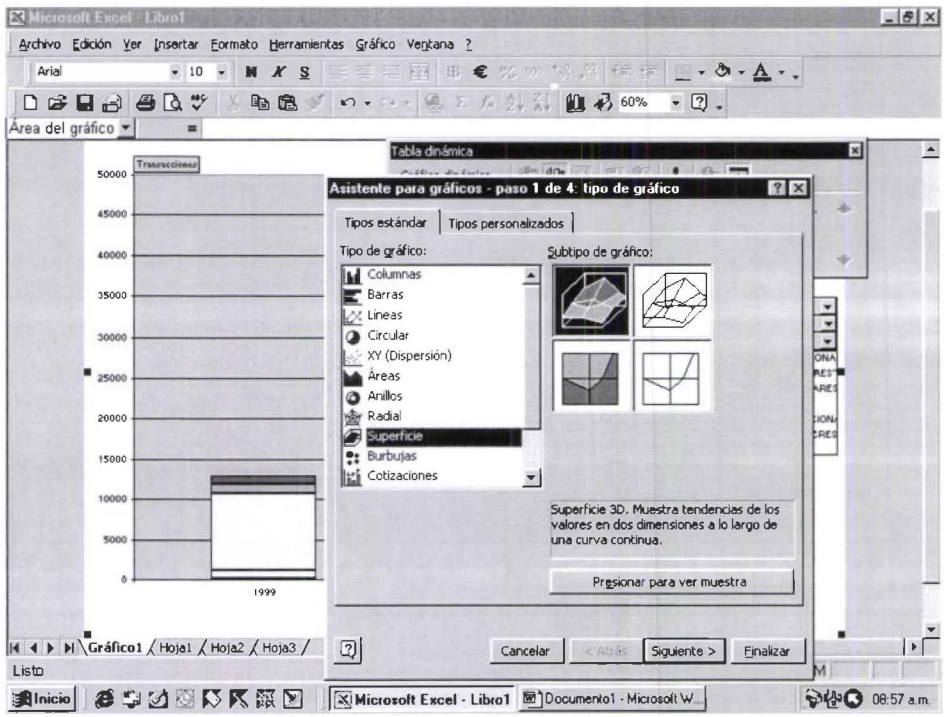


Gráfico 86.

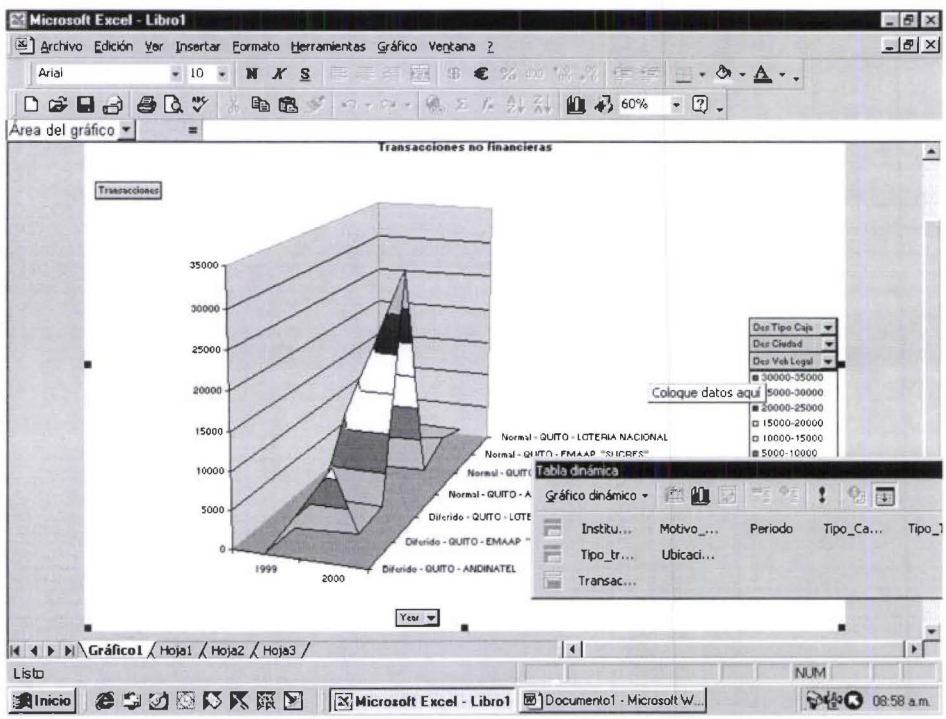


Gráfico 87.

Características de informes de tabla dinámica o de gráfico dinámico basados en OLAP

Las bases de datos OLAP proporcionan algunas de las funciones de análisis del servidor, incluidos el cálculo previo de los valores de resumen y la organización de los datos jerárquicamente.

Las bases de datos OLAP están organizadas para facilitar la recuperación y el análisis de grandes cantidades de datos. Antes de que Microsoft Excel muestre los datos resumidos en un informe de tabla dinámica o de gráfico dinámico, un servidor OLAP realiza cálculos para resumir los datos. Sólo se devuelven a Excel los datos resumidos, a medida que van siendo necesarios.

Los datos OLAP sólo pueden devolverse a Excel en forma de informe de tabla dinámica o de gráfico dinámico.

Si utiliza datos de origen OLAP, algunas funciones de análisis las proporciona directamente el servidor OLAP en vez de hacerse desde Excel. Por ejemplo, las funciones de resumen ya están definidas en el servidor. Si necesita otras funciones de una base de datos OLAP, tendrá que ponerse en contacto con el administrador del servidor OLAP.

Recuperación y la actualización de datos

Actualizar Los informes de tabla dinámica y de gráfico dinámico basados en datos de origen OLAP se actualizan automáticamente cada vez que se realiza un cambio, es decir, un servidor OLAP devuelve datos nuevos a Excel cada vez que se cambia la vista o el diseño del informe de tabla dinámica o de gráfico dinámico.

Valores del campo de página Los valores del campo de página no están disponibles en los informes. Los datos de origen OLAP siempre se recuperan para cada elemento según sea necesario, de forma que los informes pueden mostrar información de grandes bases de datos OLAP.

Consulta en segundo plano Esta función, que permite continuar trabajando en Excel mientras se recuperan datos para un informe de tabla dinámica o de gráfico dinámico, no está disponible para datos de origen OLAP.
Consultas de parámetros No se puede utilizar una consulta de parámetros para recuperar datos de una base de datos OLAP.

Opción Optimizar memoria Este valor (barra de herramientas **Tabla dinámica**, menú **Tabla dinámica**, comando **Opciones de tabla**, o menú **Gráfico dinámico**, comando **Opciones**) no está disponible para informes de tabla dinámica o de gráfico dinámico basados en datos de origen OLAP.

Diseño y presentación

Cambiar el nombre de campos y elementos Los campos y elementos cuyo nombre se ha cambiado recuperan su nombre original cuando se muestran de nuevo o se agregan al informe después de eliminarlos de la vista.

Tener acceso a datos de detalle Los valores de resumen se calculan en el servidor. El informe le permite mostrar el nivel de detalle más bajo disponible en el servidor OLAP, pero los registros de detalle subyacente que conforman los valores de resumen no están disponibles para mostrarlos en el informe.

Orden inicial Los elementos aparecen primero en el orden en que los devuelve el servidor OLAP. Puede ordenarlos o reorganizarlos manualmente si desea un orden distinto.

Cálculo

Funciones de resumen Los servidores OLAP proporcionan los valores resumidos directamente para el informe de tabla dinámica. No puede cambiar la función de resumen para un campo de datos si se trata de datos de origen OLAP y sólo puede utilizar varias funciones de resumen si el servidor OLAP proporciona varios campos de datos para distintos resúmenes.

Campos y elementos calculados No se pueden crear campos calculados ni elementos calculados en informes de tabla dinámica que estén basados en datos de origen OLAP.

Subtotales En informes de tabla dinámica que estén basados en datos de origen OLAP no pueden cambiar la función de resumen para subtotales ni mostrar subtotales para campos de fila o de columna interiores. Puesto que los totales se calculan en el servidor OLAP, no puede cambiar la configuración para incluir o excluir elementos de campos de página ocultos en los totales.

MANTENIMIENTO DEL SISTEMA DE DATA WAREHOUSE

El mantenimiento del Data warehouse es una tarea continuada que debe diseñarse antes de que el Data warehouse esté disponible para los usuarios. El mantenimiento implica:

- Implementar un mecanismo de copia de seguridad y recuperación para proteger los datos en caso de que se produzca un error del sistema u otros errores.
- Archivar la base de datos. Esto puede ser necesario para eliminar de la base de datos los datos históricos que no se utilizan y aumentar el espacio libre en disco.
- Ejecutar el Analizador de SQL Server para determinar los índices que hay que crear para mejorar el rendimiento de las consultas.

7. CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- Para una implementación exitosa de un data warehouse debe utilizarse una metodología que guíe el proceso. Se escogió la metodología propuesta por Microsoft para el desarrollo de un Data warehouse, por su costo, soporte y disponibilidad de todo el software como es sistema operativo, motor de base de datos, herramienta de desarrollo del front end de aplicaciones y software de oficina que es de este proveedor y para mantener la compatibilidad.

- Una concepción incorrecta de los términos y conceptos del mundo del data warehousing puede llevar al fracaso de su implementación.

- Una solución exitosa siempre debe involucrar a usuarios del negocio y a personal técnico de sistemas.

- Se define como servidor del Data warehouse un servidor diferente del servidor de aplicaciones o sistema OLTP, porque se necesita una base de datos independiente y con gran capacidad para las múltiples consultas que diariamente se realizaran contra este almacén. Además, para que no afecte al rendimiento del sistema operacional.

- El esquema de diseño en estrella es el que más se ajusta a las necesidades de nuestro Data warehouse, ya que del análisis se observa que no se requiere un grado tan alto de normalización.

- La infraestructura de una solución puede ser técnicamente la mejor, pero si esta no presenta datos relevantes y de calidad, esta no será una solución exitosa y en lugar de ser una ventaja de la organización, podría llevarla a un fracaso.

- Los factores críticos de éxito de un proyecto de data warehousing son:
  - Análisis de los beneficios reales que obtendría la organización (costo-beneficio)
  - Correcta interpretación de los requerimientos del usuario
  - Calidad de los datos
  - Creación y administración adecuada de meta datos
  - Elección de las herramientas adecuadas que satisfagan las necesidades de los usuarios

- Una solución de data warehousing cumple sus objetivos si una vez implementada brinda a la organización:
  - Respuesta a las preguntas del negocio a tiempo.
  - Permite que el analista del negocio tome decisiones en base a información de calidad que implique en ultima instancia una mayor rentabilidad para la organización.
  - Es base para futuros desarrollos de otras soluciones del negocio.

RECOMENDACIONES

A continuación se detallan varios de los puntos expuestos a través de todo el trabajo que se considera son claves para la implementación de data warehouse en una organización:

- Al igual que en el desarrollo de cualquier sistema transaccional, es de suma importancia utilizar una metodología de implementación de una solución de data warehousing

- Por la experiencia de varias compañías en este campo se recomienda ampliamente iniciar con proyectos pilotos que no involucren mas de dos sistemas y que además tengan una alta probabilidad de éxito.

- Es muy importante establecer expectativas reales en la comunidad de los usuarios. Típicamente los usuarios tienen la impresión de que un data warehouse será extremadamente rápido, y que responderá a la mayor parte de sus preguntas en pocos segundos. Este tipo de falsas expectativas necesitan ser restablecidas. Es extremadamente importante, y de no ser hechas en el momento preciso, puede llevar a la creencia de que el proyecto de DWH ha sido un fracaso

- Especial importancia debe prestarse a las herramientas de usuario final, pues a través de ellas es como se presenta nuestra solución al usuario final.

- Inicialmente se deben identificar a los usuarios claves y trabajar únicamente con sus definiciones, se debe congelar estas definiciones para en etapas posteriores de afinamiento satisfacer las necesidades de todos los usuarios.

- Nunca debemos olvidar que no se debe creer completamente en las promesas de capacidad y escalabilidad que promocionan los vendedores.

- Es importante tener en cuenta que no debemos cargar datos al data warehouse solo porque estos están disponibles.

Sobre las herramientas utilizadas:

- No es aconsejable trabajar con la versión Desktop de Microsoft SQL 7.0 para Windows 95,98 porque el Wizard para Import and Export de datos es demasiado pesado y en ocasiones estas plataformas no lo soportan.

- En plataformas Windows 95, 98 no se instalan los cuatro componentes de Microsoft OLAP Services, solamente se contará con los manuales en línea, pero no se tendrá a mano la consola de Administración de OLAP.

- Las tareas contenidas (de transformación de datos) en un paquete se ejecutan en el orden en que fueron creadas, por ello es aconsejable crear varios paquetes, dividiendo las tareas, en un orden adecuado, por ejemplo, primeramente se realiza la transformación de datos de las tablas de referencia o de dimensiones y posteriormente las de hechos, especialmente cuando se conserva la integridad referencial de las tablas.

8. LISTA DE LECTURAS Y BIBLIOGRAFÍA

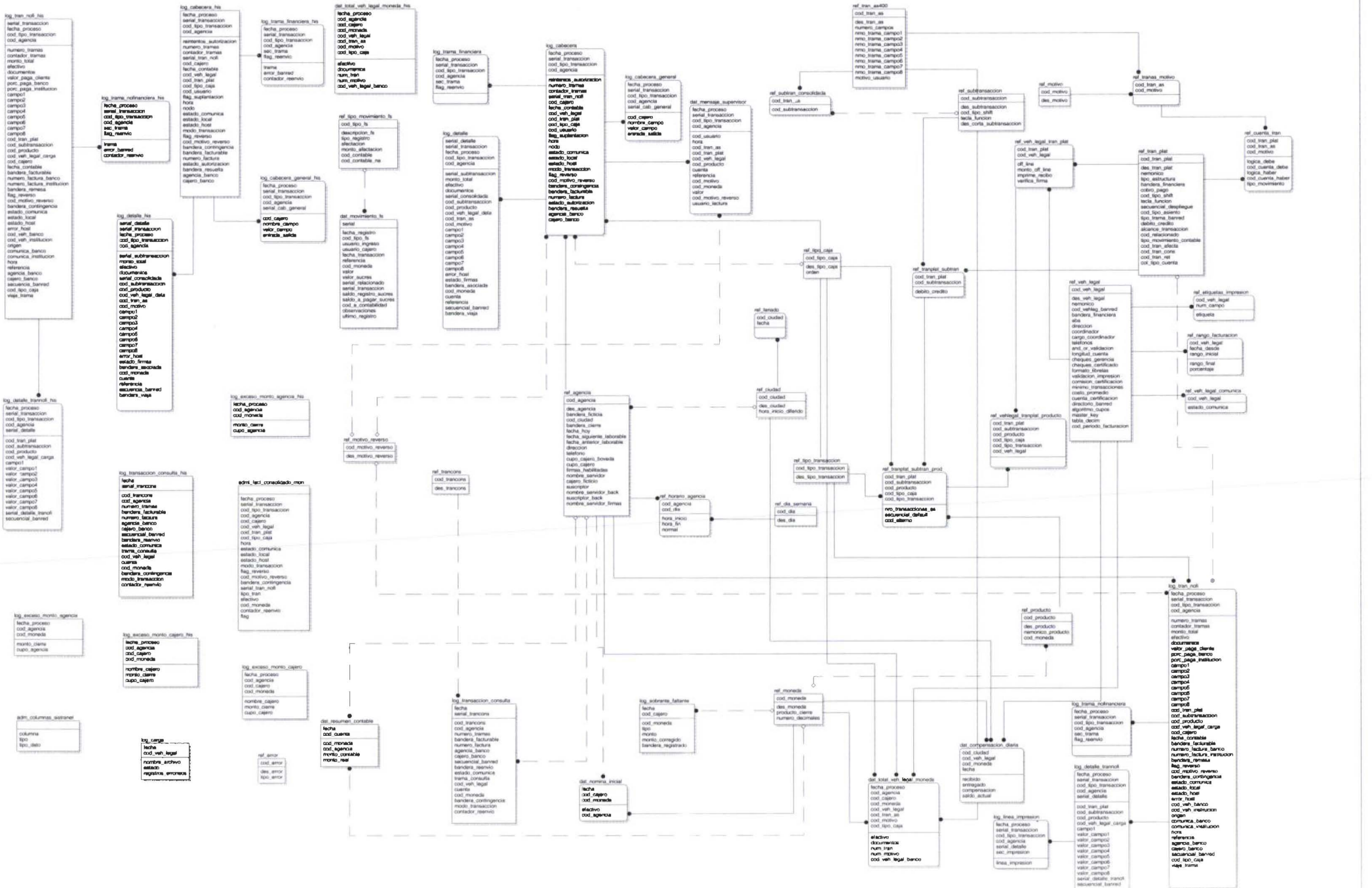
Hemos terminado, ahora, con nuestro estudio para la aplicación de una metodología para la implementación de un data warehouse en una empresa de servicios transaccionales financieros y no financieros, pero esto solo es el comienzo porque, muchos aspectos interesantes no han sido tratados con todo el detalle que merecen, y otros han sido totalmente descartados por falta de espacio. En este capítulo proporcionaré algunas referencias de lecturas posteriores, así como una bibliografía, para beneficio de aquellos lectores que deseen continuar con sus estudios sobre los data warehouse.

1. CONSEJO SUPERIOR DE INFORMATICA, http://www.map.es/csi/silice/DW1.html , España, 1999.
2. MICROSOFT, Corporation. Technical Information. http://www.microsoft.com/sql/techinfo/olap.htm, 2000.
3. MICROSOFT, Corporation. OLAP Services Cell-level Security. http://www.microsoft.com/sql/techinfo/CellLevelSecurity.doc, 2000.
4. MICROSOFT, Corporation. Guía del Administrador de SQL Server. Ed. Microsoft Corporation. Estados Unidos de América,1998.
5. MICROSOFT, Corporation. Guía del Programador de SQL Server. Ed. Microsoft Corporation. Estados Unidos de América,1998.
6. MICROSOFT, Corporation. SQL Server Books Online.
7. PASTOR, María de Lourdes. Metodología para una implementación exitosa de una solución de data warehousing. PUCE- Facultad de Ingeniería, Quito, 1997.
8. PILOT, Software. An Introduction to Data Mining, http://www.pilotsw.com/datamining/dmindex.htm, 1999.
9. PILOT, Software. An Introduction to OLAP Multidimensional Terminology and Technology. http://www.pilotsw.com/olap/olap.htm , 1999.

APENDICES

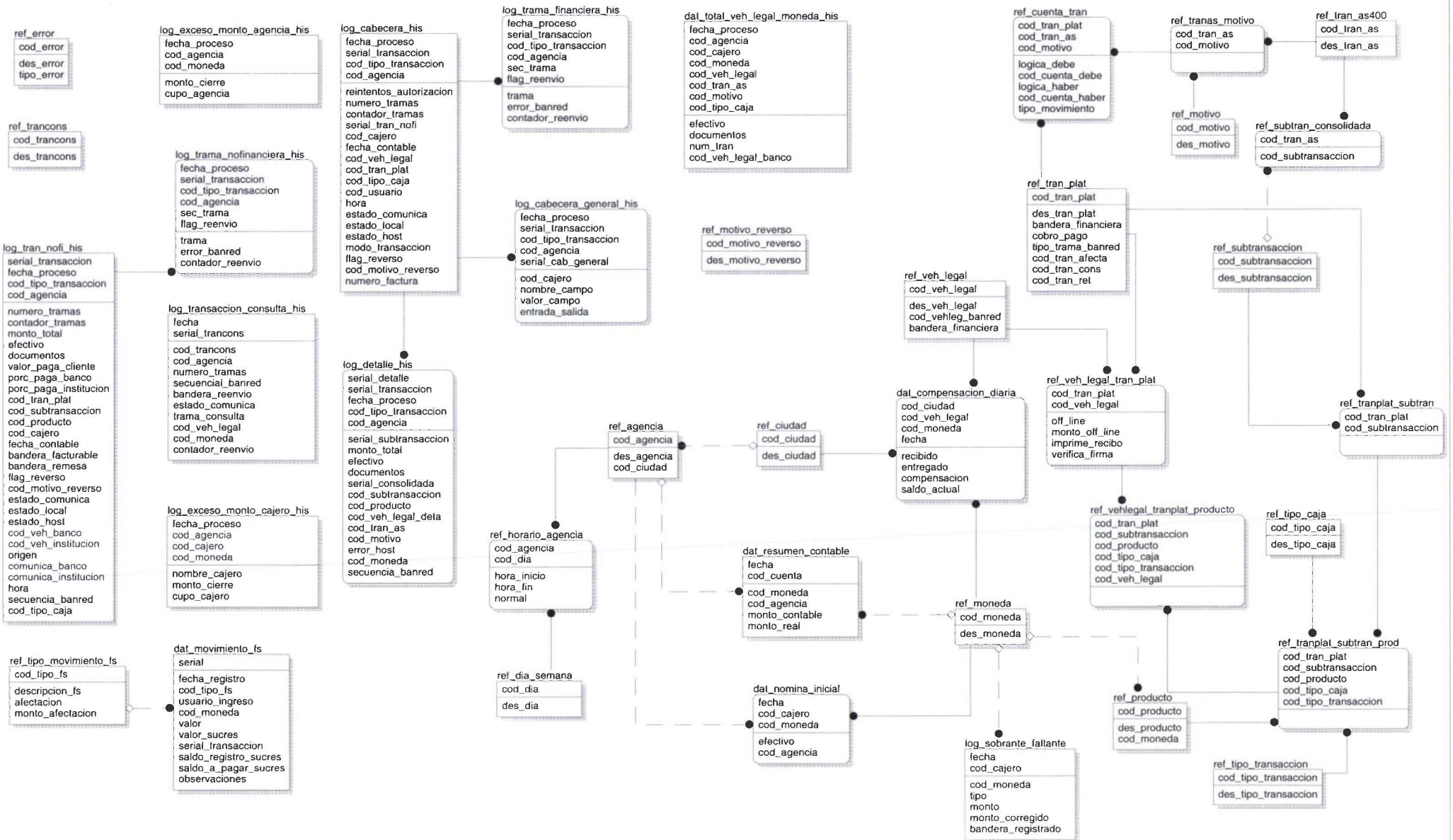
APENDICE 1

Parte central del modelo de la base de datos Operacional



APENDICE 2

Modelo de la Base de datos de data warehouse



APÉNDICE 3
Sentencias para la generación de la base de datos de data warehouse

```
CREATE TABLE ref_tipo_movimiento_fs (
     cod_tipo_fs        char(1) NOT NULL,
     descripcion_fs     varchar(20) NULL,
     afectacion         smallint NULL,
     monto_afectacion   money NULL,
     PRIMARY KEY (cod_tipo_fs) )

CREATE TABLE dat_movimiento_fs (
     serial             numeric(18,0) NOT NULL,
     fecha_registro     char(8) NULL,
     cod_tipo_fs        char(1) NULL,
     usuario_ingreso    char(10) NULL,
     cod_moneda         char(3) NULL,
     valor              money NULL,
     valor_sucres       money NULL,
     serial_transaccion numeric(18,0) NULL,
     saldo_registro_sucres money NULL,
     saldo_a_pagar_sucres money NULL,
     observaciones      varchar(184) NULL,
     PRIMARY KEY (serial),
     FOREIGN KEY (cod_tipo_fs)
                REFERENCES
ref_tipo_movimiento_fs )
 CREATE INDEX XIF104dat_movimiento_fs ON
dat_movimiento_fs
(
     cod_tipo_fs  )

 CREATE TABLE log_exceso_monto_agencia_his (
     fecha_proceso      char(8) NOT NULL,
     cod_agencia        char(3) NOT NULL,
     cod_moneda         char(3) NOT NULL,
     monto_cierre       money NULL,
     cupo_agencia       money NULL,
     PRIMARY KEY (fecha_proceso, cod_agencia,
cod_moneda)  )

 CREATE TABLE log_exceso_monto_cajero_his (
     fecha_proceso      char(8) NOT NULL,
     cod_agencia        char(3) NOT NULL,
     cod_cajero         char(4) NOT NULL,
     cod_moneda         char(3) NOT NULL,
     nombre_cajero      char(40) NULL,
     monto_cierre       money NULL,
     cupo_cajero        money NULL,
     PRIMARY KEY (fecha_proceso, cod_agencia,
cod_cajero,
          cod_moneda)  )

 CREATE TABLE log_tran_nofi_his (
     serial_transaccion numeric(15) NOT NULL,
     fecha_proceso      char(8) NOT NULL,
     cod_tipo_transaccion char(1) NOT NULL,
     cod_agencia        char(3) NOT NULL,
     numero_tramas      smallint NULL,
     contador_tramas    smallint NULL,
     monto_total        money NULL,
     efectivo           money NULL,
     documentos         money NULL,
     valor_paga_cliente money NULL,
     porc_paga_banco    numeric(5,2) NULL,
     porc_paga_institucion numeric(5,2) NULL,
     cod_tran_plat      char(4) NULL,
     cod_subtransaccion char(4) NULL,
     cod_producto       char(5) NULL,
     cod_cajero         char(4) NULL,
     fecha_contable     char(8) NULL,
     bandera_facturable char(1) NULL,
     bandera_remesa     char(1) NULL,
     flag_reverso       char(1) NULL,
     cod_motivo_reverso char(4) NULL,
     estado_comunica    char(1) NULL,
     estado_local       char(1) NULL,
     estado_host        char(1) NULL,
     cod_veh_banco      char(5) NULL,
     cod_veh_institucion char(5) NULL,
     origen             char(1) NULL,
     comunica_banco     char(1) NULL,
     comunica_institucion char(1) NULL,
     hora               char(8) NULL,
     secuencia_banred   char(6) NULL,
     cod_tipo_caja      char(1) NULL,
     PRIMARY KEY (serial_transaccion,
fecha_ proceso,
          cod_tipo_transaccion, cod_agencia)  )

CREATE TABLE log_transaccion_consulta_his (
     fecha              char(8) NOT NULL,
     serial_trancons    numeric(15) NOT NULL,
     cod_trancons       char(5) NOT NULL,
     cod_agencia        char(3) NULL,
     numero_tramas      tinyint NULL,
     secuencial_banred  char(6) NULL,
     bandera_reenvio    char(1) NULL,
     estado_comunica    char(1) NULL,
     trama_consulta     char(255) NULL,
     cod_veh_legal      char(5) NULL,
     cod_moneda         char(3) NULL,
     contador_reenvio   numeric NULL,
     PRIMARY KEY (fecha, serial_trancons)  )

CREATE TABLE ref_ciudad (
     cod_ciudad         char(3) NOT NULL,
     des_ciudad         varchar(12) NOT NULL,
     PRIMARY KEY (cod_ciudad)  )

CREATE TABLE ref_dia_semana (
     cod_dia            char(1) NOT NULL,
     des_dia            char(40) NULL,
     PRIMARY KEY (cod_dia) )

CREATE TABLE ref_error (
     cod_error          char(4) NOT NULL,
     des_error          char(40) NULL,
     tipo_error         char(1) NULL,
     PRIMARY KEY (cod_error)  )

CREATE TABLE ref_moneda (
     cod_moneda         char(3) NOT NULL,
     des_moneda         char(40) NULL,
     PRIMARY KEY (cod_moneda)  )

CREATE TABLE ref_motivo (
     cod_motivo         char(4) NOT NULL,
```

```
    des_motivo          char(40) NULL,
    PRIMARY KEY (cod_motivo)  )

CREATE TABLE ref_producto (
    cod_producto        char(5) NOT NULL,
    des_producto        char(40) NULL,
    cod_moneda          char(3) NULL,
    PRIMARY KEY (cod_producto),
    FOREIGN KEY (cod_moneda)
                REFERENCES ref_moneda )

CREATE TABLE ref_tipo_caja (
    cod_tipo_caja       char(1) NOT NULL,
    des_tipo_caja       char(40) NULL,
    PRIMARY KEY (cod_tipo_caja)  )

CREATE TABLE ref_tipo_transaccion (
    cod_tipo_transaccion char(1) NOT NULL,
    des_tipo_transaccion char(40) NULL,
    PRIMARY KEY (cod_tipo_transaccion)  )

CREATE TABLE ref_subtransaccion (
    cod_subtransaccion   char(4) NOT NULL,
    des_subtransaccion   char(40) NULL,
    PRIMARY KEY (cod_subtransaccion)  )

CREATE TABLE ref_tran_plat (
    cod_tran_plat       char(4) NOT NULL,
    des_tran_plat       varchar(24) NULL,
    bandera_financiera  char(1) NULL,
    cobro_pa            char(1) NULL,
    tipo_trama_banred   char(2) NULL,
    cod_tran_afecta     char(5) NULL,
    cod_tran_cons       char(5) NULL,
    cod_tran_ret        char(5) NULL,
    PRIMARY KEY (cod_tran_plat)  )

CREATE TABLE ref_tranplat_subtran (
    cod_tran_plat       char(4) NOT NULL,
    cod_subtransaccion  char(4) NOT NULL,
    PRIMARY KEY (cod_tran_plat,
cod_subtransaccion),
    FOREIGN KEY (cod_subtransaccion)
                REFERENCES ref_subtransaccion,
    FOREIGN KEY (cod_tran_plat)
                REFERENCES ref_tran_plat )

CREATE TABLE ref_tranplat_subtran_prod (
    cod_tran_plat       char(4) NOT NULL,
    cod_subtransaccion  char(4) NOT NULL,
    cod_producto        char(5) NOT NULL,
    cod_tipo_caja       char(1) NOT NULL,
    cod_tipo_transaccion char(1) NOT NULL,
    PRIMARY KEY (cod_tran_plat,
cod_subtransaccion, cod_producto,
        cod_tipo_caja, cod_tipo_transaccion),
    FOREIGN KEY (cod_producto)
                REFERENCES ref_producto,
    FOREIGN KEY (cod_tipo_caja)
                REFERENCES ref_tipo_caja,
    FOREIGN KEY (cod_tipo_transaccion)
                REFERENCES ref_tipo_transaccion,
    FOREIGN KEY (cod_tran_plat,
cod_subtransaccion)
                REFERENCES ref_tranplat_subtran )

CREATE TABLE ref_veh_legal (
    cod_veh_legal       char(5) NOT NULL,
    des_veh_legal       char(40) NULL,
    cod_vehleg_banred   char(6) NULL,
    bandera_financiera  char(1) NULL,
    PRIMARY KEY (cod_veh_legal) )

CREATE TABLE ref_veh_legal_tran_plat (
    cod_tran_plat       char(4) NOT NULL,
    cod_veh_legal       char(5) NOT NULL,
    off_line            char(1) NULL,
    monto_off_line      money NULL,
    imprime_recibo      char(1) NULL,
    verifica_firma      char(1) NULL,
    PRIMARY KEY (cod_tran_plat, cod_veh_legal),
    FOREIGN KEY (cod_tran_plat)
                REFERENCES ref_tran_plat,
    FOREIGN KEY (cod_veh_legal)
                REFERENCES ref_veh_legal )

CREATE TABLE ref_vehlegal_tranplat_producto (
    cod_tran_plat       char(4) NOT NULL,
    cod_subtransaccion  char(4) NOT NULL,
    cod_producto        char(5) NOT NULL,
    cod_tipo_caja       char(1) NOT NULL,
    cod_tipo_transaccion char(1) NOT NULL,
    cod_veh_legal       char(5) NOT NULL,
    PRIMARY KEY (cod_tran_plat,
cod_subtransaccion, cod_producto,
        cod_tipo_caja, cod_tipo_transaccion,
cod_veh_legal),
    FOREIGN KEY (cod_tran_plat,
cod_subtransaccion, cod_producto,
        cod_tipo_caja, cod_tipo_transaccion)
                REFERENCES
ref_tranplat_subtran_prod,
    FOREIGN KEY (cod_tran_plat, cod_veh_legal)
                REFERENCES
ref_veh_legal_tran_plat )

CREATE TABLE log_cabecera_his (
    fecha_proceso       char(8) NOT NULL,
    serial_transaccion  numeric(15) NOT NULL,
    cod_tipo_transaccion char(1) NOT NULL,
    cod_agencia         char(3) NOT NULL,
    reintentos_autorizacion tinyint NULL,
    numero_tramas       smallint NULL,
    contador_tramas     smallint NULL,
    serial_tran_nofi    numeric(15) NULL,
    cod_cajero          char(4) NOT NULL,
    fecha_contable      char(8) NULL,
    cod_veh_legal       char(5) NULL,
    cod_tran_plat       char(4) NULL,
    cod_tipo_caja       char(1) NULL,
    cod_usuario         char(10) NULL,
    hora                char(8) NULL,
    estado_comunica     char(1) NULL,
    estado_local        char(1) NULL,
    estado_host         char(1) NULL,
    modo_transaccion    char(1) NULL,
    flag_reverso        char(1) NULL,
    cod_motivo_reverso  char(4) NULL,
    numero_factura      char(15) NULL,
    PRIMARY KEY (fecha_proceso,
serial_transaccion,
        cod_tipo_transaccion, cod_agencia)  )

CREATE TABLE log_cabecera_general_his (
```

```
    fecha_proceso       char(8) NOT NULL,
    serial_transaccion  numeric(15) NOT NULL,
    cod_tipo_transaccion char(1) NOT NULL,
    cod_agencia         char(3) NOT NULL,
    serial_cab_general  int NOT NULL,
    cod_cajero          char(4) NULL,
    nombre_campo        char(20) NULL,
    valor_campo         varchar(120) NULL,
    entrada_salida      char(1) NOT NULL,
    PRIMARY KEY (fecha_proceso,
serial_transaccion,
        cod_tipo_transaccion, cod_agencia,
serial_cab_general),
    FOREIGN KEY (fecha_proceso,
serial_transaccion,
        cod_tipo_transaccion, cod_agencia)
            REFERENCES log_cabecera_his  )

CREATE TABLE log_detalle_his (
    serial_detalle      int NOT NULL,
    serial_transaccion  numeric(15) NOT NULL,
    fecha_proceso       char(8) NOT NULL,
    cod_tipo_transaccion char(1) NOT NULL,
    cod_agencia         char(3) NOT NULL,
    serial_subtransaccion int NULL,
    monto_total         money NULL,
    efectivo            money NULL,
    documentos          money NULL,
    serial_consolidada  numeric(15) NULL,
    cod_subtransaccion  char(4) NULL,
    cod_producto        char(5) NULL,
    cod_veh_legal_deta  char(5) NULL,
    cod_tran_as         char(4) NULL,
    cod_motivo          char(4) NULL,
    error_host          char(4) NULL,
    cod_moneda          char(3) NULL,
    secuencia_banred    char(6) NULL,
    PRIMARY KEY (serial_detalle, serial_transaccion,
        fecha_proceso, cod_tipo_transaccion,
cod_agencia),
    FOREIGN KEY (fecha_proceso,
serial_transaccion,
        cod_tipo_transaccion, cod_agencia)
            REFERENCES log_cabecera_his)

CREATE TABLE log_sobrante_faltante (
    fecha               char(8) NOT NULL,
    cod_cajero          char(4) NOT NULL,
    cod_moneda          char(3) NULL,
    tipo                char(1) NULL,
    monto_corregido     money NULL,
    bandera_registrado  char(1) NULL,
    PRIMARY KEY (fecha, cod_cajero),
    FOREIGN KEY (cod_moneda)
            REFERENCES ref_moneda )

CREATE TABLE log_trama_financiera_his (
    fecha_proceso       char(8) NOT NULL,
    serial_transaccion  numeric(15) NOT NULL,
    cod_tipo_transaccion char(1) NOT NULL,
    cod_agencia         char(3) NOT NULL,
    sec_trama           smallint NOT NULL,
    flag_reenvio        char(1) NOT NULL,
    trama               varchar(2000) NULL,
    error_banred        char(4) NULL,
    contador_reenvio    numeric(18,0) NULL,
    PRIMARY KEY (fecha_proceso,
serial_transaccion,
        cod_tipo_transaccion, cod_agencia, sec_trama,
        flag_reenvio),
    FOREIGN KEY (fecha_proceso,
serial_transaccion,
        cod_tipo_transaccion, cod_agencia)
            REFERENCES log_cabecera_his)

CREATE TABLE log_trama_nofinanciera_his (
    fecha_proceso       char(8) NOT NULL,
    serial_transaccion  numeric(15) NOT NULL,
    cod_tipo_transaccion char(1) NOT NULL,
    cod_agencia         char(3) NOT NULL,
    sec_trama           smallint NOT NULL,
    flag_reenvio        char(1) NOT NULL,
    trama               varchar(2000) NULL,
    error_banred        char(4) NULL,
    contador_reenvio    numeric(18,0) NULL,
    PRIMARY KEY (fecha_proceso,
serial_transaccion,
        cod_tipo_transaccion, cod_agencia, sec_trama,
        flag_reenvio),
    FOREIGN KEY (serial_transaccion,
fecha_proceso,
        cod_tipo_transaccion, cod_agencia)
            REFERENCES log_tran_nofi_his )

CREATE TABLE ref_agencia (
    cod_agencia         char(3) NOT NULL,
    des_agencia         varchar(24) NULL,
    cod_ciudad          char(3) NULL,
    PRIMARY KEY (cod_agencia),
    FOREIGN KEY (cod_ciudad)
            REFERENCES ref_ciudad )

CREATE TABLE ref_tran_as400 (
    cod_tran_as         char(4) NOT NULL,
    des_tran_as         char(40) NULL,
    PRIMARY KEY (cod_tran_as) )

CREATE TABLE ref_subtran_consolidada (
    cod_tran_as         char(4) NOT NULL,
    cod_subtransaccion  char(4) NULL,
    PRIMARY KEY (cod_tran_as),
    FOREIGN KEY (cod_subtransaccion)
            REFERENCES ref_subtransaccion,
    FOREIGN KEY (cod_tran_as)
            REFERENCES ref_tran_as400 )

CREATE TABLE ref_tranas_motivo (
    cod_tran_as         char(4) NOT NULL,
    cod_motivo          char(4) NOT NULL,
    PRIMARY KEY (cod_tran_as, cod_motivo),
    FOREIGN KEY (cod_motivo)
            REFERENCES ref_motivo,
    FOREIGN KEY (cod_tran_as)
            REFERENCES ref_tran_as400 )

CREATE TABLE dat_compensacion_diaria (
    cod_ciudad          char(3) NOT NULL,
    cod_veh_legal       char(5) NOT NULL,
    cod_moneda          char(3) NOT NULL,
    fecha               char(8) NOT NULL,
    recibido            money NULL,
    entregado           money NULL,
    compensacion        money NULL,
```

```
    saldo_actual        money NULL,
    PRIMARY KEY (cod_ciudad, cod_veh_legal,
cod_moneda, fecha),
    FOREIGN KEY (cod_ciudad)
                REFERENCES ref_ciudad,
    FOREIGN KEY (cod_moneda)
                REFERENCES ref_moneda,
    FOREIGN KEY (cod_veh_legal)
)

CREATE TABLE ref_motivo_reverso (
    cod_motivo_reverso   char(4) NOT NULL,
    des_motivo_reverso   char(40) NULL,
    PRIMARY KEY (cod_motivo_reverso) )

CREATE TABLE ref_trancons (
    cod_trancons        char(5) NOT NULL,
    des_trancons        char(40) NULL,
    PRIMARY KEY (cod_trancons) )

CREATE TABLE dat_nomina_inicial (
    fecha               char(8) NOT NULL,
    cod_cajero          char(4) NOT NULL,
    cod_moneda          char(3) NOT NULL,
    efectivo            money NULL,
    cod_agencia         char(3) NOT NULL,
    PRIMARY KEY (fecha, cod_cajero, cod_moneda),
    FOREIGN KEY (cod_moneda)
                REFERENCES ref_moneda,
    FOREIGN KEY (cod_agencia)
                REFERENCES ref_agencia )

CREATE TABLE dat_resumen_contable (
    fecha               char(8) NOT NULL,
    cod_moneda          char(3) NULL,
    cod_agencia         char(3) NULL,
    monto_contable      money NULL,
    monto_real          money NULL,
    PRIMARY KEY (fecha, cod_cuenta),
    FOREIGN KEY (cod_agencia)
                REFERENCES ref_agencia,
    FOREIGN KEY (cod_moneda)
                REFERENCES ref_moneda )

CREATE TABLE ref_cuenta_tran (
    cod_tran_plat       char(4) NOT NULL,
    cod_tran_as         char(4) NOT NULL,
    cod_motivo          char(4) NOT NULL,
    logica_debe         char(5) NULL,
    cod_cuenta_debe     char(4) NULL,
    logica_haber        char(5) NULL,
    cod_cuenta_haber    char(4) NULL,
    tipo_movimiento     char(1) NULL,
    PRIMARY KEY (cod_tran_plat, cod_tran_as,
cod_motivo),
    FOREIGN KEY (cod_tran_as, cod_motivo)
                REFERENCES ref_tranas_motivo,
    FOREIGN KEY (cod_tran_plat)
                REFERENCES ref_tran_plat )

CREATE TABLE ref_horario_agencia (
    cod_agencia         char(3) NOT NULL,
    cod_dia             char(1) NOT NULL,
    hora_inicio         char(8) NULL,
    hora_fin            char(8) NULL,
    normal              char(1) NULL,
    PRIMARY KEY (cod_agencia, cod_dia),
    FOREIGN KEY (cod_agencia)
                REFERENCES ref_agencia,
    FOREIGN KEY (cod_dia)
                REFERENCES ref_dia_semana  )

CREATE TABLE dat_total_veh_legal_moneda_his (
    fecha_proceso       char(8) NOT NULL,
    cod_agencia         char(3) NOT NULL,
    cod_cajero          char(4) NOT NULL,
    cod_moneda          char(3) NOT NULL,
    cod_veh_legal       char(5) NOT NULL,
    cod_tran_as         char(4) NOT NULL,
    cod_motivo          char(4) NOT NULL,
    cod_tipo_caja       char(1) NOT NULL,
    efectivo            money NULL,
    documentos          money NULL,
    num_tran            int NULL,
    cod_veh_legal_banco  char(5) NULL,
    PRIMARY KEY (fecha_proceso, cod_agencia,
cod_cajero,
        cod_moneda, cod_veh_legal, cod_tran_as,
cod_motivo,
        cod_tipo_caja)
```

APÉNDICE 4

Como parte de la fase de generación del esquema de base de datos del data warehouse, se realiza el análisis de las tablas y campos a ser desechados del esquema, por ello aquí se presenta el detalle de los mismos y las justificaciones respectivas.

Formato

Tabla		
Columna	Operación	Justificación

Log_cabecera_his		
Flag_suplantacion	Eliminar	No es necesaria porque no se tiene un análisis por transacciones realizadas en estado de suplantación.
Nodo	Eliminar	El nombre del servidor que genera la transacción no es relevante para el análisis.
Bandera_contingencia	Eliminar	El estado de la aplicación de una contingencia no es tomado en cuenta para el análisis
Bandera_facturable	Eliminar	Si es facturable no es relevante para el análisis propuesto, ya que no se tiene un modulo de facturación
Estado_autorización	Eliminar	Las autorizaciones de pagos no están contemplados en el análisis
Reintentos_autorización	Eliminar	El numero de reintentos para la consulta de autorización de pagos no es relevante para el análisis
Bandera_resuelta	Eliminar	No es relevante
Agencia_banco	Eliminar	No es relevante porque el análisis se realiza por agencia propia
Cajero_banco	Eliminar	No es relevante porque el análisis se realiza por cajero propio
Cod_usuario	Eliminar	Porque ya consta el código de cajero como identificador

Log_detalle_his		
Campo1	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Campo2	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Campo3	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Campo4	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Campo5	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Campo6	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Campo7	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Campo8	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Estado_firmas	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis

Bandera_asociada	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Cuenta	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Referencia	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis
Bandera_viaja	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis

Log_tran_nofi_his		
Cod_veh_legal_carga	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Bandera_facturable	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Numero_factura_banco	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Numero_factura_institución	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Bandera_remesa	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Bandera_contingencia	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Campo1	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Campo2	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Campo3	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Campo4	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Campo5	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Campo6	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Campo7	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Campo8	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Error_host	Eliminar	Porque es un campo no utilizado
Comunica_banco	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Comunica_institución	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Referencia	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Agencia_banco	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Cajero_banco	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Secuencial_banred	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Viaje_trama	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.

Log_detalle_trannofi_his		
Cod_veh_legal_carga	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Valor_campo1	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Valor_campo2	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Valor_campo3	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Valor_campo4	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Valor_campo5	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Valor_campo6	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Valor_campo7	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Valor_campo8	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Serial_detalle_trannofi	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Secuencial_banred	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.

Log_trama_financiera_his		
Sec_trama	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Trama	Eliminar	Por ser la trama armada que se envía no es necesario para el análisis de la información.

Log_trama_nofinanciera_his		
Sec_trama	Eliminar	Porque esta información es útil para el sistema transaccional, mas no para el análisis de la información.
Trama	Eliminar	Por ser la trama armada que se envía no es necesario para el análisis de la información.

Log_cabecera_general_his		
Valor_campo	Modificar la definición a varchar(103)	Esta superdimensionado el valor de este campo
Entrada_salida	Eliminar	Podría servir para otros análisis, pero no para esta fase.

Log_transacción_consulta_his		
Bandera_facturable	Eliminar	No es necesario para el análisis de la información.
Numero_factura	Eliminar	No es necesario para el análisis de la información.
Cajero_banco	Eliminar	No es necesario porque el análisis se realiza por cajero propio
Secuencial_banred	Eliminar	No aporta para el análisis de la información en el data warehouse
Estado_comunica	Eliminar	"A" o "R" que significa
Trama_consulta	Eliminar	Para que servirá.
Cuenta	Eliminar	No es necesario para el análisis de la información.
Bandera_contingencia	Eliminar	No es necesario para el análisis de la información.
Modo_transacción	Eliminar	No es necesario para el análisis de la información. Siempre es "N".

Log_exceso_monto_cajero_his		
Nombre_cajero	Eliminar	Se podría quitar pero a lo mejor sirve para consultas más rápidas.

Ref_tran_plat		
Des_tran_plat	Reducir el campo a varchar(24)	Sobredimensionado el campo
Nemónica	Eliminar	Porque no es necesario para el análisis de la información.
Tipo_estructura	Eliminar	Porque no es necesario para el análisis de la información.
Bandera_financiera	Eliminar	Porque no es necesario para el análisis de la información. Talvez si "S" o "N"
Cobro_pago	Eliminar	Porque no es necesario para el análisis de la información. Podria servir "C" o "N"
Cod_tipo_shift	Eliminar	Porque no es necesario para el análisis de la información.
Tecla_funcion	Eliminar	Porque no es necesario para el análisis de la información.
Secuencial_despliegue	Eliminar	Porque no es necesario para el análisis de la información.
Cod_tipo_asiento	Eliminar	Porque no es necesario para el análisis de la información.
Debito_credito	Eliminar	Es un campo vacío
Alcance_transacción	Eliminar	Es un campo constante para todos los registros

		"B" y no necesario.
Cod_relacionado	Eliminar	Porque no es necesario para el análisis de la información.
Tipo_movimiento_contable	Eliminar	Porque no es necesario para el análisis de la información, porque no es requisito un análisis contable que aun no esta implementado en el sistema transaccional.
Cod_tipo_cuenta	Eliminar	Porque no es necesario para el análisis de la información.

Ref_veh_legal		
Nemonico	Eliminar	Porque no es necesario para el análisis de la información.
Aba	Eliminar	Porque no es necesario para el análisis de la información.
Dirección	Eliminar	Porque no es necesario para el análisis de la información.
Coordinador	Eliminar	Porque no es necesario para el análisis de la información.
Cargo_cordinador	Eliminar	Porque no es necesario para el análisis de la información.
Telefonos	Eliminar	Porque no es necesario para el análisis de la información.
And_or_validación	Eliminar	Porque no es necesario para el análisis de la información.
Longitud_cuenta	Eliminar	Porque no es necesario para el análisis de la información.
Cheques_gerencia	Eliminar	Porque no es necesario para el análisis de la información.
Cheques_certificado	Eliminar	Porque no es necesario para el análisis de la información.
Formato_libretas	Eliminar	Porque no es neccsario para el análisis de la información.
Validación_impresión	Eliminar	Porque no es necesario para el análisis de la información.
Comision_certificación	Eliminar	Porque no es necesario para el análisis de la información.
Minimo_transacciones	Eliminar	Porque no es necesario para el análisis de la información.
Costo_promedio	Eliminar	Porque no es necesario para el análisis de la información.
Cuenta_certificación	Eliminar	Porque no es necesario para el análisis de la información.
Directorio_banred	Eliminar	Porque no es necesario para el análisis de la información.
Algoritmo_cupos	Eliminar	Porque no es necesario para el análisis de la información.
Master_key	Eliminar	Porque no es necesario para el análisis de la información.
Tabla_decim	Eliminar	Porque no es necesario para el análisis de la información.
Cod_periodo_facturación	Eliminar	Porque no es necesario para el análisis de la información.

Ref_subtransaccion		
Cod_tipo_shift	Eliminar	Porque no es necesario para el análisis de la información.
Tecla_funcion	Eliminar	Porque no es necesario para el análisis de la información.
Des_corta_subtransaccion	Eliminar	Porque no es necesario para el análisis de la información.

Ref_tranplat_subtran		
Debito_credito	Eliminar	Porque no es necesario para el análisis de la información.

Ref_ciudad		
Des_ciudad	Recortar a varchar(10) o (15)	Porque no es necesario un campo tan grande.
Hora_inicio_diferido	Eliminar	Porque existe otro campo que identifica la transacción en diferido.

Ref_agencia		
Des_agencia	Recortar a varchar(24)	Para ahorrar almacenamiento.
Bandera_ficticia	Eliminar	Porque no es necesario para el análisis de la información.
Bandera_cierre	Eliminar	Porque no es necesario para el análisis de la información.
Fecha_hoy	Eliminar	Porque no es necesario para el análisis de la información.
Fecha_siguiente_laborable	Eliminar	Porque no es necesario para el análisis de la información.
Fecha_anterior_laborable	Eliminar	Porque no es necesario para el análisis de la información.
Dirección	Eliminar	Porque no es necesario para el análisis de la información.
Telefono	Eliminar	Porque no es necesario para el análisis de la información.
Cupo_cajero_boveda	Eliminar	Porque no es necesario para el análisis de la información.
Cupo_cajero	Eliminar	Porque no es necesario para el análisis de la información.
Firmas_habilitadas	Eliminar	Porque no es necesario para el análisis de la información.
Nombre_servidor	Eliminar	Porque no es necesario para el análisis de la información.
Cajero_ficticio	Eliminar	Porque no es necesario para el análisis de la información.
Suscriptor	Eliminar	Porque no es necesario para el análisis de la información.
Nombre_servidor_back	Eliminar	Porque no es necesario para el análisis de la información.
Suscriptor_back	Eliminar	Porque no es necesario para el análisis de la información.
Nombre_servidor_firmas	Eliminar	Porque no es necesario para el análisis de la información.

Ref_producto		
Nemonico_producto	Eliminar	Porque no es necesario para el análisis de la información.

Ref_tipo_caja		
Orden	Eliminar	Porque no es necesario para el análisis de la información.

Ref_moneda		
Producto_cierre	Eliminar	Porque no es necesario para el análisis de la información.
Numero_decimales	Eliminar	Porque no es necesario para el análisis de la información.

Ref_tran_as400		
Numero_campos	Eliminar	Porque no es necesario para el análisis de la información.
Nmo_trama_campo1	Eliminar	Porque no es necesario para el análisis de la información.
Nmo_trama_campo2	Eliminar	Porque no es necesario para el análisis de la información.
Nmo_trama_campo3	Eliminar	Porque no es necesario para el análisis de la información.
Nmo_trama_campo4	Eliminar	Porque no es necesario para el análisis de la información.
Nmo_trama_campo5	Eliminar	Porque no es necesario para el análisis de la información.
Nmo_trama_campo6	Eliminar	Porque no es necesario para el análisis de la información.
Nmo_trama_campo7	Eliminar	Porque no es necesario para el análisis de la información.
Nmo_trama_campo8	Eliminar	Porque no es necesario para el análisis de la información.
Motivo_usuario	Eliminar	Porque no es necesario para el análisis de la información.

Ref_tranplat_subtran_prod		
Cod_tipo_caja	Eliminar	Porque no es necesario para el análisis de la información.
Cod_tipo_transaccion	Eliminar	Porque no es necesario para el análisis de la información.
Nro_transacciones_as	Eliminar	Porque no es necesario para el análisis de la información.
Secuencial_default	Eliminar	Porque no es necesario para el análisis de la información.
Cod_alterno	Eliminar	Porque no es necesario para el análisis de la información.

Ref_tipo_movimiento_fs		
Descripción_fs	Recortar a varchar(20)	Porque esta sobredimensionado.
Tipo_registro	Eliminar	Porque no es necesario para el análisis de la información.
Afectación	Eliminar	Porque no es necesario para el análisis de la información.
Cod_contable	Eliminar	Porque no es necesario para el análisis de la

		información.
Cod_contable_na	Eliminar	Porque no es necesario para el análisis de la información.

Dat_movimiento_fs		
Usuario_cajero	Eliminar	Ya existe otro campo con la misma información.
Fecha_transacción	Eliminar	Ya existe otro campo con la misma información.
Referencia	Eliminar	Ya existe otro campo con la misma información.
Serial_relacionado	Eliminar	Ya existe otro campo con la misma información.
Cod_a_contabilidad	Eliminar	Ya existe otro campo con la misma información.
Ultimo_registro	Eliminar	Ya existe otro campo con la misma información.
Observaciones	Reducir a varchar(182) o varchar(190)	Para reducir almacenamiento.