Implementando OLAP Temporal para la Web

Director: **Dr. Alejandro Vaisman** e-mail: avaisman@dc.uba.ar

Integrantes: **Adrián Izquierdo** L.U.: 540/90 e-mail: aizquier@dc.uba.ar
Ángel M. Ktenas L.U.:1650/86 e-mail: aktenas@dc.uba.ar

2007

Índice

Lista de Figuras

RESUMEN

El procesamiento analítico en línea también conocido como OLAP (*On Line Analytical Processing*), ha recibido mucha atención de la comunidad de Base de Datos en los últimos años. En los modelos de datos para OLAP, es aceptado que las estructuras (típicamente dimensiones y tablas de hechos) están sujetas a frecuentes cambios. Por ejemplo, en una aplicación para un data warehouse de ventas, una dimensión representando las sucursales de una tienda, puede cambiar cuando nuevas sucursales inauguren o cierren, o sean reasignadas de una región a otra. Adicionalmente, la estructura que representa sucursales puede cambiar, por ejemplo, si un nivel del agrupamiento es eliminado, o uno nuevo es introducido.

Nuestro trabajo se basa en el modelo y lenguajes temporales introducido por Mendelzon y Vaisman, denominado *TOLAP* (*Temporal OLAP*). El lenguaje soporta, por ejemplo, consultas como "Listar el monto de ventas por tipo de producto, usando la categorización que cada ítem tenía en el momento de ser vendido". La creciente necesidad de permitir el acceso vía Web a las aplicaciones de soporte a la decisión, ha dado origen al concepto de Web Warehousing. Nuestro enfoque para Web warehousing es diferente al tradicional, ya que nos proponemos disponer la información OLAP en la Web, haciendo uso de la maquinaria existente para desarrollar aplicaciones Web, particularmente XML, SOAP y servicios Web.

La primera contribución de la presente tesis consiste en una implementación de TOLAP basada en un modelo de 3 capas y una arquitectura Web que le da amplia importancia al protocolo de comunicación SOAP dejando solamente a la base de datos la función de ser un repositorio de datos y almacenamiento persistente de las estructuras del modelo *TOLAP*. Esta es una importante evolución del prototipo *TOLAP* – pre-existente. Nuestra segunda contribución es la propuesta del lenguaje *TOLAP-QL* (*TOLAP* Query Language) con un estilo SQL. El lenguaje TOLAP actual se basa en reglas estilo Datalog, un lenguaje menos amigable y más especializado. La tercera contribución de la tesis es el desarrollo e implementación de un optimizador de consultas para *TOLAP-QL*, que permite rescribir las consultas *TOLAP-QL* on-the-fly teniendo en cuenta toda la información de los metadatos *TOLAP* y estadísticas de consultas pasadas. Este optimizador incluye un agente que puede decidir crear *Vistas Materializadas* que pueden a través de la reescritura, ser utilizadas como parte de la consulta, reemplazando a las tablas de hechos (fact tables), obteniéndose una notable mejora en la performance. La característica novedosa de este optimizador es que las *Vistas Materializadas* que crea no deben ser actualizadas, ya que corresponden a estados pasados de la Base de Datos. Finalmente, presentamos un caso de estudio real, sobre el cual, además realizamos experimentos que dan idea de la performance de nuestra implementación

ABSTRACT

OLAP (*On Line Analytical Processing*), has received a lot of attention from Database community during the last ten years. It is widely accepted that structures in OLAP information systems (typically dimensions and facts tables) are subject to frequent changes.

For example, in an application for sales warehouse, a dimension representing the divisions in a shop, can change when new divisions inaugurate or close, or are re-assigned from one region to other one. Additionally, the structure that represents divisions can change, for instance, if an aggregated level is eliminated or if a new one is introduced.

Our work is based on the model and temporal languages introduced by Mendelzon and Vaisman, so called TOLAP (Temporal OLAP). The language supports, for example, queries like "Total of sales for type of product, using the categorization that each item had at sale time ". The increasing need to allow decision support applications to be available across the Web, led to the concept of Web Warehousing. Our approach for Web warehousing is different from the traditional one however, since we propose to arrange the information OLAP in the Web, making use of the existing machinery to develop Web applications, particularly XML, SOAP and Web services.

The first contribution of the present dissertation consists of an implementation of TOLAP based on a three-tier model over a Web architecture where the application resides in a dedicated server, and the database remains as an information repository. This is an important evolution of the pre-existent TOLAP prototype. Our second contribution is the proposal of the TOLAP-QL query language (TOLAP Query Language), a SQL-like language. The previous TOLAP language was based on rules following a Datalog's style, a less friendly and more specialized language. The third contribution of this thesis is the development and implementation of a TOLAP-QL query optimizer, which allows query rewriting on-the-fly, making use of the TOLAP metadata and database statistics. This optimizer includes an agent that can decide whether or not to create Materialized views that can be used as part of the translated query after applying the corresponding query rewriting replacing (mainly) the fact tables by the materialized views, in order to obtain an important performance gain. The new optimizer's characteristics are based on Materialized views and those views are not allowed to be updated since they correspond to the previous (i.e., closed) states of the OLAP database. Finally, we present a real-world case study which helps us making some experiments to measure the performance gain.

CAPÍTULO 1.

INTRODUCCIÓN

El procesamiento analítico en línea también conocido como OLAP (*On Line Analitical Procesing*), ha recibido mucha atención de la comunidad de Base de Datos en los últimos años. En consecuencia, se han propuesto varios modelos para aplicaciones OLAP [CT98,Kim96,Le98]. En estos modelos, los datos están organizados en *dimensiones* y *tablas de hechos* [Kim96]. Es un hecho reconocido que dichas estructuras están sujetas a frecuentes cambios [MV03,Vai01]. Por ejemplo, en una aplicación para un data warehouse[1] de ventas, un usuario puede querer analizar las ventas a lo largo de diferentes dimensiones, como regiones o sucursales. La información de hechos (ventas) debe ser agregada sobre categorías organizadas en dimensiones jerarquizadas. En esta forma, una *ciudad* agrega sobre una *provincia*, una *provincia* sobre una *región*, etc. Por ejemplo, una dimensión representando las sucursales de una tienda de ventas, llamémosla Sucursales, puede cambiar cuando nuevas sucursales inauguren o cierren, o sean reasignadas de una región a otra. Adicionalmente, la estructura de la jerarquía de Sucursales por sí misma puede cambiar cuando un nivel del agrupamiento es eliminado, o uno nuevo es introducido.

Para tratar el problema de representación, almacenaje y consulta de la información temporal en un data warehouse, Mendelzon y Vaisman [MV00] introdujeron un modelo de datos temporal multidimensional así como un lenguaje de consulta temporal para soportarlo, denominado *TOLAP* (*Temporal OLAP*). Este lenguaje soporta consultas del tipo "Listar el monto de ventas por tipo de producto, usando la categorización que cada ítem tenía en el momento de ser vendido", o de tipo booleano como "Estaban los productos categorizados por marcas hace 2 años?". En [MV03] los autores presentan una implementación de este sistema. La creciente necesidad de permitir el acceso vía Web a las aplicaciones existentes ha dado origen al concepto de Web Warehousing. Las aplicaciones de Web Warehousing están principalmente focalizadas en recoger datos de fuentes XML[2], considerando a la Web como un repositorio XML. Otros enfoques utilizan herramientas de data warehousing para analizar el comportamiento de los clientes cuando navegan el sitio Web de la compañía. Nuestro enfoque es

[1] Colección de datos orientadas a un dominio, integrado, no volátil y variable en el tiempo que ayuda a la toma de decisiones de la empresa u organización.

[2] XML (eXtensible Markup Language), es un metalenguaje extensible de etiquetas desarrollado por el W3C -World Wide Web Consortium- (www.w3c.org).

diferente: nosotros queremos disponer la información OLAP en la Web, haciendo uso de la maquinaria existente para desarrollar aplicaciones Web, particularmente XML, SOAP[1] y servicios Web[2].

Un prototipo de *TOLAP* fue realizado con anterioridad a esta nueva implementación. Esta aplicación es muy dependiente de la herramienta de implementación, y está basada en Tecnología de Bases de Datos Oracle y depende de un fat client Java (una applet) realizando conexiones directas a la base de datos para realizar tanto la construcción del modelo *TOLAP* como la ejecución de consultas. Por lo tanto, nuestra primera contribución consiste en una implementación basada en un modelo de 3 capas y una arquitectura Web que le da amplia importancia al protocolo de comunicación SOAP [Mit03], dejando solamente a la base de datos como repositorio de datos y almacenamiento persistente de estructuras del modelo *TOLAP*.

Nuestra segunda contribución es el lenguaje *TOLAP-QL* (*TOLAP* Query Lenguage), con un estilo SQL. En la implementación anterior, las consultas se escribían en el mismo formato que la especificación formal de las reglas *TOLAP* (estilo Datalog[3]), un lenguaje menos amigable. Esta nueva sintaxis resulta mucho más intuitiva para los usuarios habituados a trabajar con SQL. Dado que nuestro trabajo fue implementado en una arquitectura Web abierta, teniendo en cuenta estándares de mercado, como SOAP y Web Services, *TOLAP* puede no sólo ser ejecutado a través de sus herramientas, sino también a través de cualquier cliente habilitado para operar con SOAP, tanto para la creación del modelo de datos *TOLAP* como para la ejecución de consultas *TOLAP-QL*.

Nuestra tercera contribución es el desarrollo e implementación de un optimizador de consultas *TOLAP-QL*, que permite re-escribir las consultas *TOLAP-QL* on-the-fly teniendo en cuenta toda la información de los metadatos *TOLAP* y estadísticas de consultas pasadas. Este optimizador incluye un agente que puede decidir crear *Vistas Materializadas* que pueden a través de la reescritura ser utilizadas como parte de la consulta, reemplazando a las tablas de hecho, obteniéndose una notable mejora en la performance para bases de datos del orden de los 10 GB en adelante. La característica novedosa de este optimizador es que las *Vistas Materializadas* que crea no deben ser actualizadas, ya que corresponden a estados pasados de la Base de Datos.

Finalmente, mostraremos como nuestro modelo e implementación se aplican a un caso real, sobre el cual, además realizamos experimentos que dan idea de la performance de nuestra implementación.

---

[1] SOAP (*Simple Object Access Protocol*) es un protocolo estándar (creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C) que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML (www.w3c.org/TR/SOAP).

[2] Un servicio web es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.

[3] Lenguaje de consultas y reglas para bases de conocimiento, que sintácticamente es un subconjunto del lenguaje Prolog.

En resumen, la nueva versión de *TOLAP* propuesta en este trabajo, incluye:

- Arquitectura Abierta, predominando el acceso de los servicios *TOLAP* a través de la web.
- Mejora en el acceso y administración de Metadata *TOLAP*.
- Soporte para atributos temporales en las jerarquías de las dimensiones.
- Nuevo sistema integrado de tipos de datos para *TOLAP*.
- Nuevas características de Administración *TOLAP*.
- Utilitarios para carga del modelo de datos de negocio.
- Definición e implementación de *TOLAP-QL*, un nuevo lenguaje para especificación de consultas *TOLAP*.
- Optimización de consultas *TOLAP* mediante utilización de Vistas Materializadas (VM) temporales.
- Independencia del proveedor del RDBMS[1].

Cabe aclarar que resultados preliminares de esta Tesis fueron publicados en [VIK106] y [VIK206].

Este documento se organiza de la siguiente forma: en el Capítulo 2 se describe el marco teórico en el cual se basa esta tesis. El Capítulo 3 introduce el modelo de datos de *TOLAP*. El capítulo 4 presenta el lenguaje de consultas de *TOLAP-QL* mientras que en el Capítulo 5 se hace una introducción de la Arquitectura de Software de *Web TOLAP*. En el Capítulo 6 se explica el procesamiento de una consulta *TOLAP-QL* y en el Capítulo 7 se discute el Optimizador de consultas. El Capítulo 8 presenta un caso de estudio *TOLAP* sobre un modelo de la Central de Deudores del Banco Central de la República Argentina. En el Capítulo 9 se muestran y discuten los resultados de experimentaciones realizadas sobre dicho caso de estudio. Finalmente se presentan los apartados de conclusiones, trabajos futuros, anexos y bibliografía.

[1] Sistema Administrador de Base de Datos Relacionales. RDBMS viene del acrónimo en inglés Relational Data Base Management System.

Capítulo 2.

MARCO TEORICO

Diversas propuestas en la literatura OLAP abordan el problema de la temporalidad de los datos. El primero en referenciar el problema del manejo de dimensiones cambiantes ("Slowly changing dimensions") fue Kimball [Kim96], quién sugirió algunas soluciones parciales. Bliujute *et al* [BSSJ98] presentaron una alternativa al esquema estrella. Pedersen *et al* [Ped99] presentaron un modelo multidimensional para el manejo de datos complejos, donde el aspecto temporal es referenciado en conjunción con otros problemas de modelado de datos. Eder *et al* [EKM02], y Bebel *et al* [BEKMW04] también presentaron modelos temporales para OLAP. Mendelzon y Vaisman [MV00, MV03] introdujeron TOLAP, un modelo multidimensional temporal que soporta modificaciones de esquema e instancias de un Data Warehouse.

La importancia creciente de las aplicaciones Web introdujo la necesidad de hacer que las aplicaciones OLAP estén disponibles a través de aquella. Una línea de trabajo en ese sentido ha sido la llamada Web Warehousing. Estas aplicaciones consideran básicamente a la Web como un enorme repositorio de datos en formato XML. Marian et al [MACM01] estudiaron el problema de detectar, manejar y notificar de cambios en los Web data warehouses conteniendo datos XML, en el contexto del proyecto Xyleme [Xylm], un proyecto que apuntó a la construcción de un almacén dinámico de datos en la Web (Dynamic World Wide Web data warehouse). Xyleme periódicamente refresca sus datos y computa los cambios usando un algoritmo de diferencias (diff). Vrdoljak et al [VBR03] utilizaron XML schema [TSMBM06] para diseñar almacenes de datos en una forma semiautomática.

Una segunda línea de trabajo propone el uso de técnicas, modelos y lenguajes orientadas a la Web para poder implementar aplicaciones OLAP y de data warehousing sobre la Web. En este sentido, Bihn *et al* [BTM01] propusieron MetaCube-X, un protocolo que permite acceder a una federación de cubos de datos. MetaCube-X provee una metadata XML para compartir cubos de datos en un ambiente distribuido. Niemi *et al* también utilizan XML para el intercambio de cubos OLAP [NNNT03]. Finalmente, Hümmer *et al* [HBH03] utilizan plantillas XML para almacenar, consultas y cambiar datos en el warehouse. Estas plantillas describen el esquema del cubo de datos y la información de las dimensiones.

Por el lado del software propietario, XML for Analysis [XMLA] utiliza SOAP y XML para acceder a sistemas remotos. Nuestra propuesta sigue un criterio similar, usando XML, SOAP y servicios Web para intercambiar datos. Sin embargo, nosotros le damos a este conjunto de herramientas colaborativas un papel más importante, permitiéndoles optimizar la navegación y modificación del esquema e instancias

del warehouse, poniendo el foco en el problema de OLAP temporal. Nuestra propuesta no está atada a ningún proveedor de base de datos: la base de datos pueden intercambiarse, simplemente modificando un archivo XML de configuración.

2.1 Definiciones básicas

El modelo multidimensional de datos con el que trabajaremos en esta Tesis, se basa en el denominado *modelo estrella* [Kim96], conformado básicamente por tablas de *dimensiones y* tablas de *hechos*.

Las tablas de *hechos* contienen las transacciones de negocios de una empresa. Están compuestas por dos tipos básicos de columnas, las medidas (measures), que típicamente contienen hechos y las columnas que contienen referencias foráneas a tablas de dimensiones. Ejemplos típicos de estas son las tablas de: *Ventas (ClienteId, TiendaId, ArticuloId, Fecha, Cantidad, precio)*, *Compras (ProveedorId, ArticuloId, Fecha, Cantidad, precio)*, etc.

La *medida* es un valor sobre el que se pueden aplicar funciones de agregación. Por ejemplo, si un comercio vendió una serie de productos específicos, la cantidad y el precio de cada uno de los ítems vendidos podrían ser sumados o promediados para obtener el total de artículos vendidos y el precio total o promedio de los artículos.

Las tablas de *dimension*, describen las entidades de negocio de una empresa, la cual usualmente representa información categorizada jerárquicamente tal como el tiempo, departamentos de una empresa, ubicaciones geográficas, productos. También se las conoce como tablas de referencia o maestras.

Las *jerarquías* son estructuras lógicas que utilizan niveles ordenados como un medio de organización de la información. Una jerarquía puede ser utilizada para definir la agregación de la información. Por ejemplo, en una dimensión *tiempo,* una jerarquía podría agregar información desde el nivel *mensual* al nivel *cuatrimestral* y de este al nivel *anual*.

Dentro de una jerarquía, cada *nivel* se relaciona con los *niveles superiores e inferiores* a él. Los valores de datos de *niveles inferiores* se ven agregados a *niveles superiores*. Una *dimensión* puede estar compuesta por más de una jerarquía. Por ejemplo, en la *dimensión producto*, podría haber dos jerarquías, una para las categorías de *productos* y otra para los *proveedores* de los productos.

Debido a la naturaleza propia de la información que describen las *dimensiones*, las jerarquías que la componen usualmente no cambian a lo largo del tiempo. Sin embargo, cuando dichas jerarquías sufren variaciones, originan cambios en la estructura de la tabla de dimensión que la contiene. A las *dimensiones* cuyas jerarquías sufren cambios las denominamos *temporales*.

Al *nivel inferior* de la jerarquía que compone una *dimensión* lo denominamos *ínfimo*.

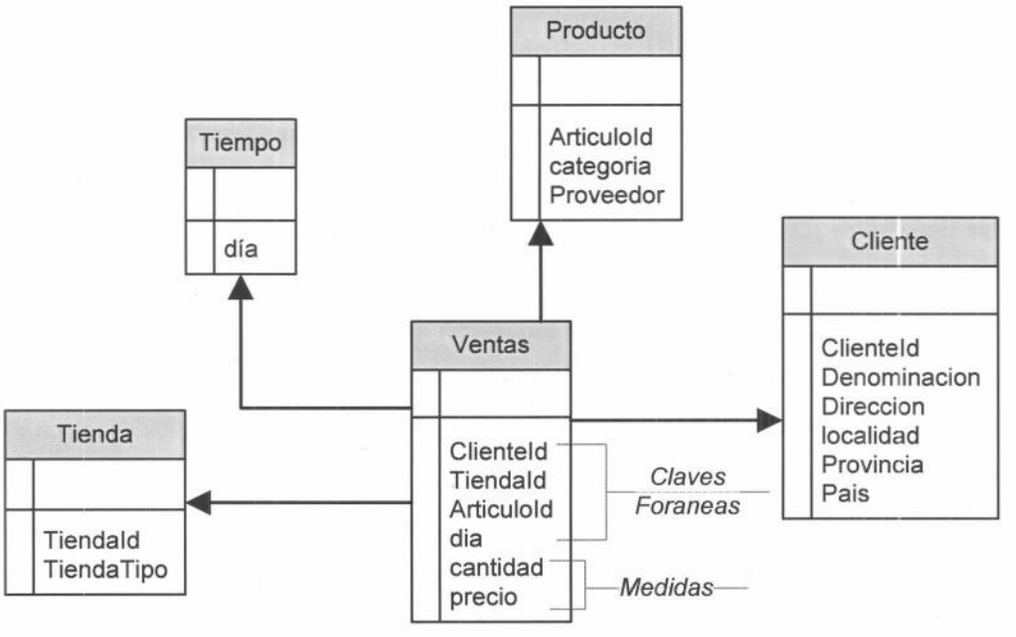


Figura 2.1. Modelo de Ventas

La Figura 2.1 ilustra un modelo típico de un Warehouse de Ventas con una *tabla de hechos* denominada *Ventas* y las correspondientes *dimensiones Producto*, *Cliente*, *Tienda*, *Tiempo*.

OLAP provee un conjunto de operaciones que le permiten al usuario navegar a través de los datos. Las más típicas son *Roll-Up* y *Drill-Down*. Las *roll-ups* agregan datos a una granularidad mayor mientras que las *drill-downs* permiten una menor. Por ejemplo, en el caso planteado en la Figura 2.1 tenemos

cuatro dimensiones, *Producto*, *Cliente, Tienda* y *Tiempo* y una tabla de hechos *Ventas*. En este caso *cantidad* y *precio* son las medidas de la tabla de hechos.

En este ejemplo, un usuario típico podría querer ver la información agregada por *Producto* (operación *Roll-Up*) y luego refinar su análisis descubriendo las ventas mensuales de cada *producto* (operación de *Drill-Down*).

2.2 OLAP Temporal

En trabajos previos Mendelzon y Vaisman [MV00], introdujeron un modelo multidimensional temporal, en donde se agrega una estampa de tiempo a los elementos de las dimensiones, tanto a nivel de esquema como de instancias (o ambos) para mantener la trazabilidad de las modificaciones que ocurren durante el ciclo de vida de la dimensión. Este modelo fue denominado TOLAP.

El modelo soporta lo que en la terminología de base de datos temporales se denomina como "tiempo valido" [Sno95], esto es, el estampado de tiempo (en adelante lo denominaremos *timestamp*) que representa el momento en el cual el hecho grabado en la Base de Datos (BD) se convirtió en válido. El momento en que este fue físicamente grabado en la BD representa el tiempo de transacción. Los conceptos presentados aquí podrían ser fácilmente extendidos para manejar también el tiempo de transacción. Consideraremos el tiempo como discreto, esto es, un punto en la línea-temporal, llamado un "punto temporal", se corresponderá con un entero. Presentaremos la idea básica de este modelo, esencialmente por medio de ejemplos.

En el modelo adoptado, las *dimensiones* y (probablemente) algunas *tablas de hechos*, pueden ser modificadas a través de un conjunto de operadores de modificación, definidos en [MV00, MV03]. La implementación que presentamos en esta tesis también soporta estos operadores, los cuales pueden ser aplicados, cuando afectan al esquema de una *dimensión*, sin acceder a la base de datos. De hecho, en esta tesis proponemos una arquitectura específicamente orientada a la eficiente implementación de estos operadores sobre la Web. En la sección 2.3 se presenta un resumen de este conjunto de operadores de modificación de dimensiones.

2.3 Operadores TOLAP

TOLAP implementa una serie de operadores que permiten definir funciones de *rollup* entre los distintos niveles de una *dimensión*. A modo de ejemplo, la Figura 2.2 muestra la organización jerárquica de *niveles* que contiene la *dimensión Geografía*. Esta jerarquía fue generada a partir de la aplicación sucesiva de operadores *TOLAP* tal como se detalla en la sección 2.3.1. En parte izquierda de la figura, se muestra el esquema de la dimensión. La parte derecha, muestra una posible instancia del esquema.

Los operadores se dividen en dos clases: *operadores estructurales* (que modifican el esquema de una *dimensión*) y *operadores sobre instancias* (que modifican la instancia de una *dimensión*).

2.3.1 Operadores Estructurales

El operador *Generalize*, crea un nivel X por sobre el nivel Y (generaliza el nivel Y). A nivel instancia, es necesario definir una función *rollup* asociada al arco XY del esquema.

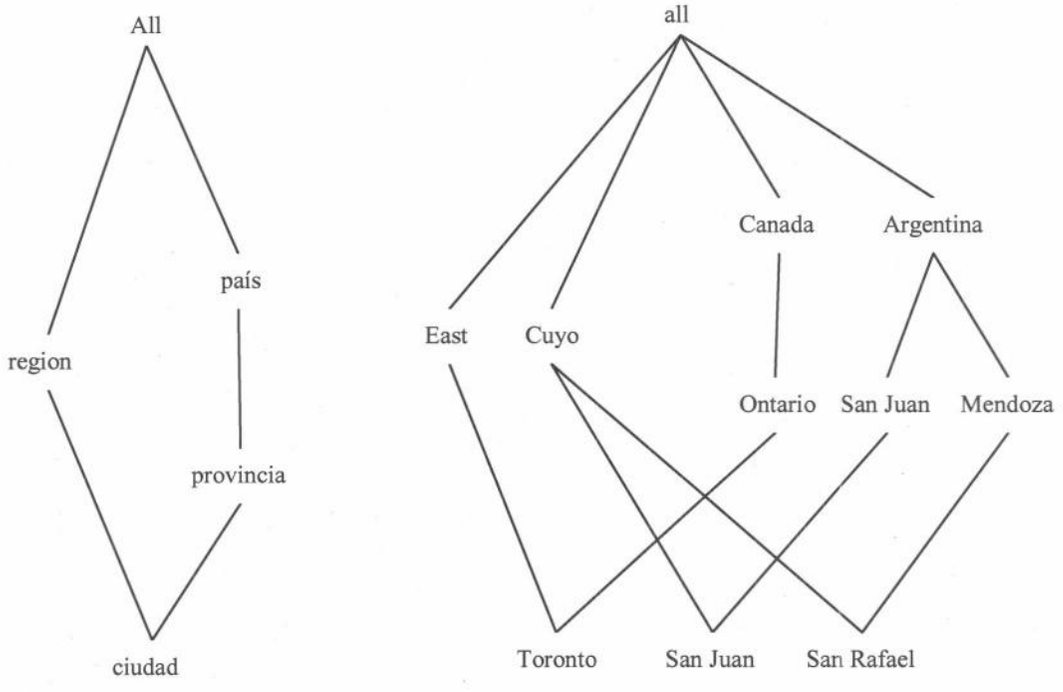


Figura 2.2. Dimensión Geografía: esquema (izquierda) e instancias (derecha).

En el momento de la creación de la tabla de *dimensión*, se especifica el nivel inferior, en este caso el nivel *ciudad*. Luego, por aplicación sucesiva del operador *Generalize*, se agregan los niveles superiores, por ej.: *Generalize ciudad* a *provincia*, luego *Generalize* de *provincia* a *país*, finalmente Generalize de *ciudad* a *región*. Cabe destacar la existencia en todas las tablas de dimensiones del nivel *All*.

La Figura 2.2 muestra, a la izquierda el esquema estructural de la tabla de *dimensión* obtenido por la aplicación sucesiva del operador *Generalize*. A la derecha se muestra la instancia correspondiente.

Cada nivel de una dimensión, esta asociado a una instancia. Habitualmente a los componentes de esta instancia se los denomina *miembros*, es decir, valores correspondientes al dominio de datos de un nivel. Por ejemplo, en el gráfico anterior, *Toronto*, *San Juan* y *San Rafael* son *miembros* del nivel *ciudad*.

El operador *Specialize* ejecuta una especialización de un nivel, en particular, el nivel inferior de una dimensión. Esto es, refina un nivel de agregación.

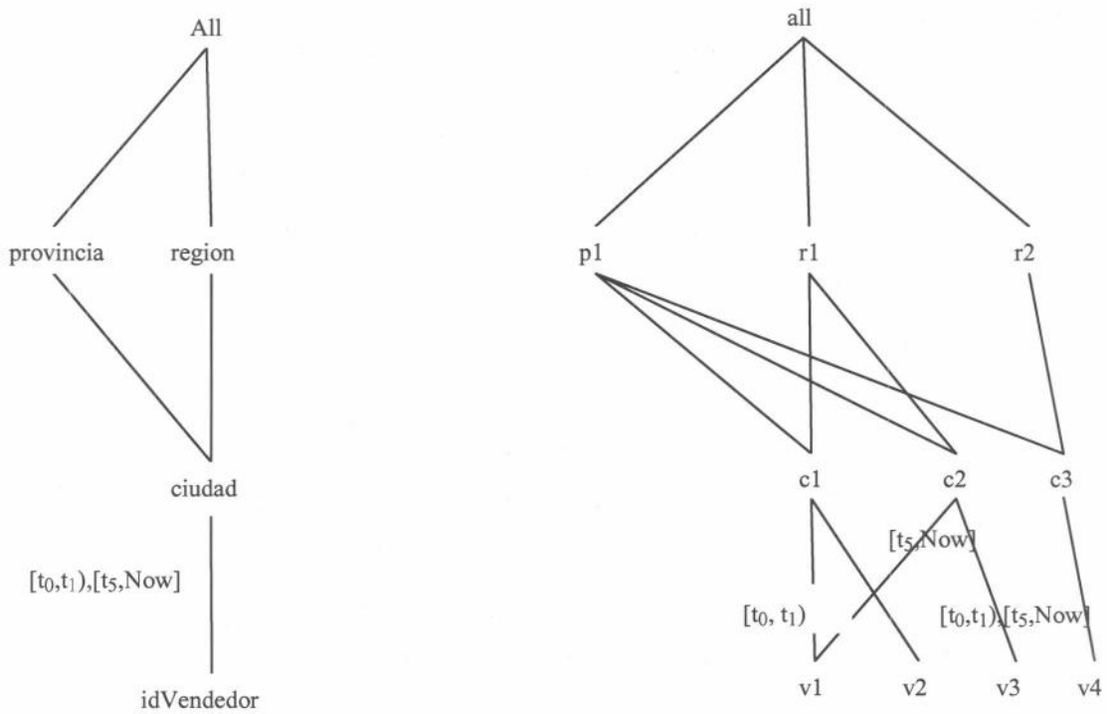


Figura 2.3. Nivel idVendedor: DeleteLevel en t1 y Specialize en t5

El operador DeleteLevel elimina del esquema un nivel y los arcos que lo accesan, solo sí dicha eliminación mantiene la existencia de un único ínfimo en el esquema (All tampoco puede borrarse) .

Nótese que en la Figura 2.2, los arcos no tienen rótulos. Esto indica que el esquema e instancia son validos a lo largo de todo el *lifespan* (intervalo de existencia) de la *dimensión*. Contrariamente, en la Figura 2.3, vemos que los arcos están rotulados con intervalos. Así, el nivel inferior en un instante *t0* es el nivel *idVendedor*, en un instante posterior *t1* un DeleteLevel se aplica sobre el mismo nivel, por lo cual es eliminado. La decisión se revierte en *t5* cuando se vuelve a aplicar un Specialize de *ciudad* con el nivel *idVendedor*. Nótese que las rollups entre instancias cambiaron, dado que *v1* fue movido a la city *c2* en t5. El resto de las rollups, *v2, v3* y *v4* mantienen su relación con las ciudades previas a la eliminación del nivel en *t1*.

En la Figura 2.2 se utilizo una notación para los intervalos, la cual reemplazaremos en el resto de los ejemplos por una más compacta donde: $[t_0, t_{j-1}] = t^*$ y $[t_j, Now] = tj$.

El operador *Relate* define una función de rollup entre dos niveles independientes para un instante *t*. Dos niveles son independientes si no existe un camino entre ellos en la jerarquía de niveles. La inversa de este operador es el *Unrelate*.

2.3.2 Operadores sobre Instancias

Dos operadores básicos se definen sobre instancias: *AddInstance* y *DeleteInstance*. *AddInstance*, permite en un instante *t* insertar una instancia en un nivel. Dicha instancia no debe pertenecer al conjunto de instancias vigente al instante *t* que se quiere agregar. Debe especificarse cada rollup de instancia sobre los niveles superiores. Por ejemplo, si se agrega una instancia al nivel *ciudad*, es necesario especificar la funciones de rollup de instancias hacia los niveles *región* y *provincia*.

El operador *DeleteInstance* permite eliminar una instancia del conjunto vigente a un instante *t*. El operador está definido sólo si la instancia que se desea eliminar no tiene rollups con instancias de niveles inferiores en la jerarquía de la dimensión.

La Figura 2.4 (a) muestra el esquema de la *dimensión Producto* que se utiliza para este ejemplo. En la Figura 2.4 (b) se muestra la *instancia* a un momento dado, mientras que en la Figura 2.4 (c) vemos como resulta la instancia en un instante posterior en que se aplica el operador *DeleteInstance*.

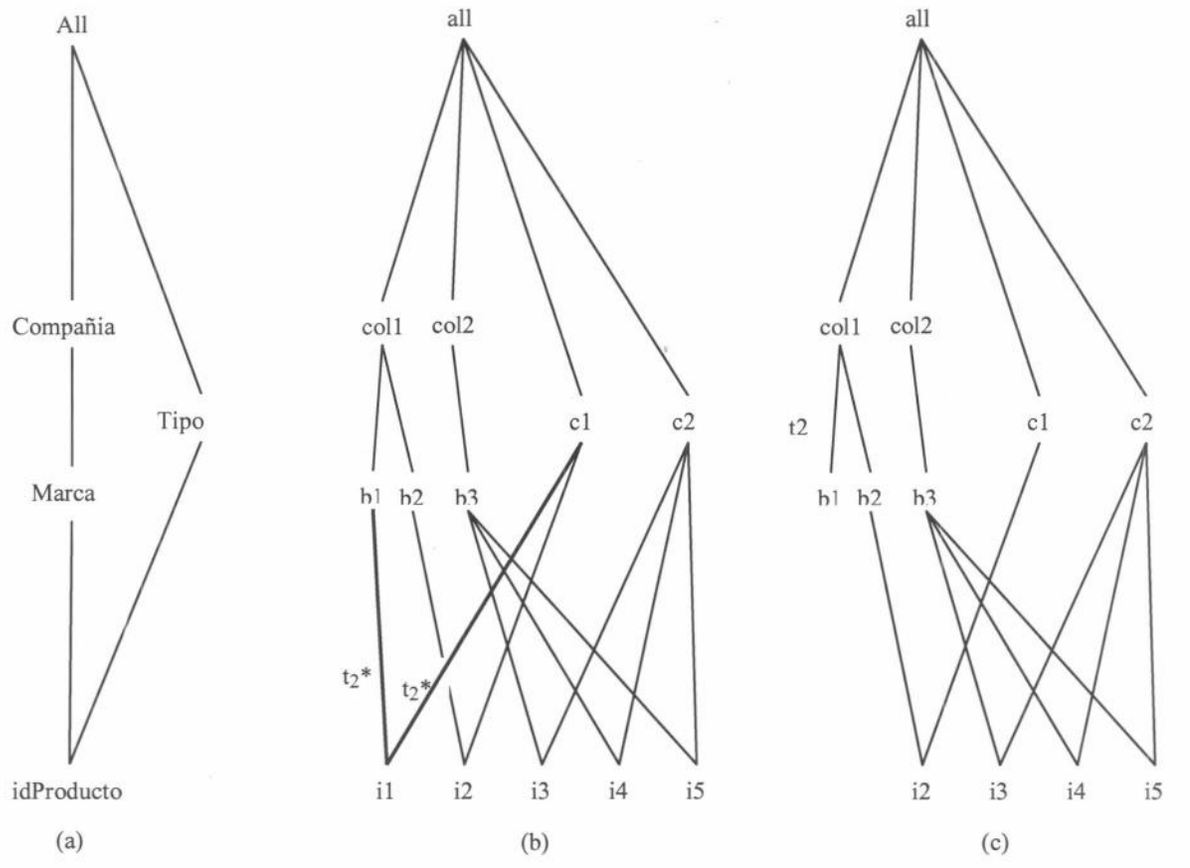


Figura 2.4 Operador DeleteInstance de i1

En la Figura 2.4 se muestra el *DeleteInstance* de *i1*, del nivel *idProducto*, en el instante *t2*, *b1* del nivel *Marca* queda sin rollups con instancias del nivel *productId*, mientras que *c1* mantiene su rollup con la instancia *i2*.

Existen operadores adicionales que se definen a partir de los dos anteriores. El operador *Reclassify* permite reclasificar rollups de instancias en un instante dado. En la Figura 2.6, la instancia *c3* del nivel *ciudad* a partir de *t5* se cambia la rollup de *región* de *r3* a *r2*, es decir, la rollup de *c3* a *r3* deja de ser válida para hacerse válida la relación con *r2*.

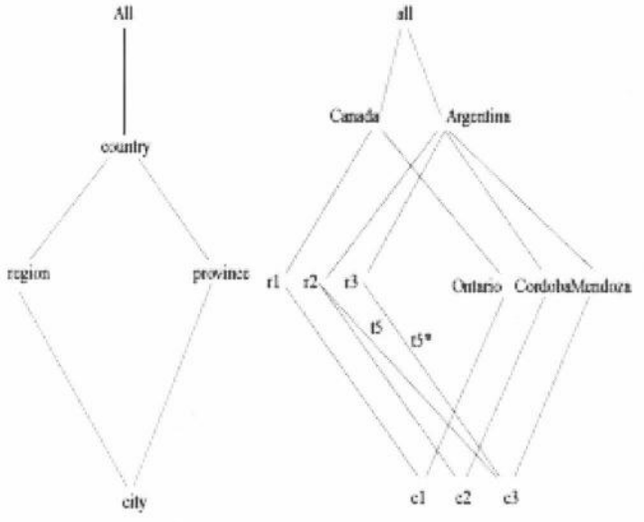


Figura 2.5. Operador Reclassify c3 a r2

El operador *Split* permite realizar una división de un *miembro* de nivel en dos o más nuevos *miembros*, asignando las rollups correspondientes a las nuevas instancias a ser creadas.

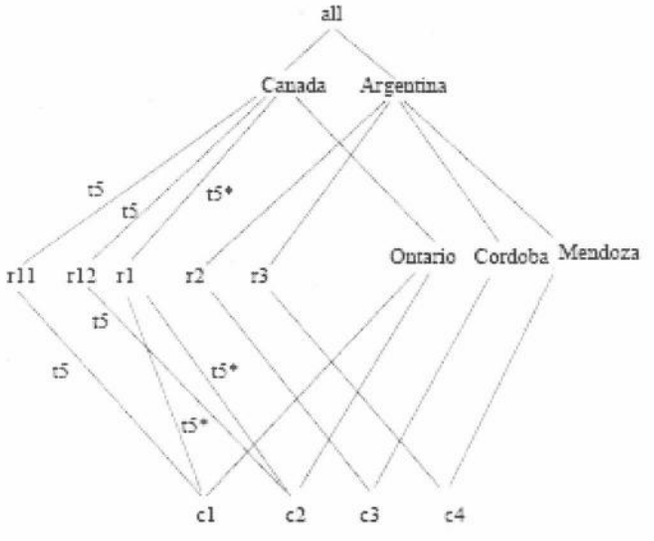


Figura 2.6 Operador Split r1 en r11y r12

En la Figura 2.7, se muestra un *Split* de la instancia *r1* en *r11* y *r12* en el instante *t5*, eliminándose las rollups que tenía *r1* en ese instante.

La inversa del *Split* es el operador *Merge*, que realiza la unión de dos o más instancias de un nivel. En la siguiente figura se muestra un merge de *r2* y *r3* realizado sobre la nueva región *r23* en el instante *t5*. Todas las regiones que tenían rollups con *r2* y *r3* ahora están relacionadas con *r23*, y a su vez con *Argentina*, el *country* al que tenían rollup *r2* y *r3*.

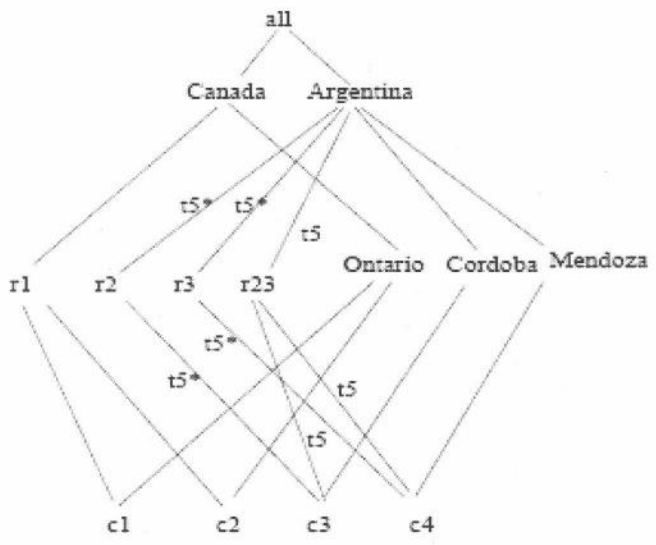


Figura 2.7 Operador Merge r23 a partir de r2 y r3

El operador *Update* se utiliza para agregar una nueva instancia a un nivel de una dimensión a un instante *t*, haciendo que la nueva instancia herede las mismas rollups que poseía la instancia anterior que se especifica en la invocación del operador, por ej., se podría hacer un *Update* sobre el nivel *región r1* por *r4*, que va a mantener las rollups con *c1, c2* del nivel *ciudad* y *Canadá* del nivel *country.*

Los operadores *TOLAP* se estudian en detalle en [Vai01].

2.4 Resumen

En este capítulo se vieron diferentes enfoques para el estudio de OLAP Temporal introducidos por distintos autores y su estado de evolución. Luego se hizo una breve reseña al trabajo realizado sobre TOLAP. Posteriormente, se introdujeron conceptos básicos de OLAP Temporal y por último mostramos los operadores TOLAP y su funcionamiento.

Capítulo 3.

MODELO DE DATOS TOLAP

El modelo de datos *TOLAP* está constituido por tres tipos de objetos:

- *Dimensiones* y *Tablas de Hechos*
- Metadata *TOLAP*
- *Vistas Materializadas*: utilizadas por *TOLAP* para la optimización de consultas.

3.1 Dimensiones y Tablas de Hechos Temporales

La definición de *dimensión* y *tablas de hechos* fue abordada en el Capitulo 2, dentro de la sección 2.1 (Definiciones básicas).

Las tablas de hechos sufren actualizaciones en su estructura, ante la aplicación de determinados operadores de actualización. Cuando una de las dimensiones pertenecientes a una *tabla de hechos* cambia su ínfimo, por ej., por la aplicación de un operador *Specialize*, *TOLAP* automáticamente realiza el cierre de la versión actual de la *tabla de hechos* y la creación una nueva, con el ínfimo actualizado. Esta situación se origina en el hecho de que el operador *Specialize* representa una desagregación de la información que esta capturando el sistema OLAP en la *tabla de hechos*, por lo cual esta debe cambiar en forma acorde. Para ilustrar este hecho, veamos la Figura 2.1, en donde la tabla de hechos a un instante t es Ventas y supongamos que a partir de t+1 necesitamos capturar la información de ventas desagregada por hora para lo cual aplicamos un *Specialize* del nivel *día* al nivel *hora*, en la *dimensión Tiempo*. A partir de t+1 la estructura de Ventas debe cambiar para poder almacenar los datos con dicha desagregación y la anterior estructura queda congelada.

3.2 Metadatos TOLAP

La arquitectura de 3 capas utilizada para la actual implementación de TOLAP consta de: un servidor de Base de Datos, un servidor de aplicaciones y el cliente SOAP.

Debido a que usualmente, el cliente necesita utilizar información de los esquemas de las dimensiones con las que opera y que como la comunicación se realiza a través de servicios web provistos por servidor de aplicaciones, cada vez que se realiza una solicitud, se hace necesario acceder al servidor de base de datos para recuperar información, que en muchos casos es de tipo estructural (es decir, en general su tamaño no es excesivamente grande).

A los efectos de optimizar este circuito, se opto por mantener una copia de los Metadatos de TOLAP fuera del ambiente del servidor de base de datos, por lo que los metadatos de *TOLAP* se encuentran divididos en dos partes, a saber:

- Metadatos SQL, que mantienen los metadatos de *TOLAP* en el motor de base de datos.
- Metadatos XML, que se encuentra almacenada en el servidor de aplicaciones y contiene la información estructural del modelo.

La ventaja de mantener ambos modelos reside en que los metadatos XML son utilizados por el servidor de aplicaciones para brindar los servicios correspondientes al cliente *TOLAP*, en lo referente al modelo estructural, evitando tener que enviar un requerimiento al motor de base de datos por cada solicitud.

Los metadatos son mantenidos por medio de documentos XML ubicados en el servidor de aplicaciones. La primera vez que los metadatos de un objeto *TOLAP* determinado se utilizan, por ejemplo, una dimensión, son cacheados en memoria y luego accedidos a través de las facilidades brindadas por DOM[1]. Este modelo de caching mejora la performance de la operatoria sobre la metadata, ya que son accesos a memoria.

Cabe destacar que la metadata *TOLAP* especificada en XML también reside en la base de datos, en un modelo propio, con la finalidad de lograr la completitud del modelo relacional (ya que tener las *tablas de hechos* sin las *dimensiones* y sus datos no permitiría resolver consultas TOLAP complejas dentro del motor de base de datos como se verá más adelante). Se definió un esquema de coordinación entre los cambios realizados a la metadata XML y la metadata SQL (ver Coordinación SQL/XML).

---

1 DOM abreviatura de Document Object Model (en español 'Modelo de Objetos de Documento'), es una forma de representar los elementos de un documento estructurado (tal como una página web HTML o un documento XML) como objetos que tienen sus propios métodos y propiedades. El responsable del DOM es el World Wide Web Consortium (W3C)W3C La especificación DOM se encuentra disponible en: http://www.w3c.org/DOM/

A modo de ejemplo, se muestra el siguiente documento XML contenido en el archivo `metaFactTables.xml`, que contiene la metadata de las *tablas de hechos*. En este caso, se presenta la tabla Préstamos que posee tres versiones: versión 1, versión 2, creadas a partir del cambio de ínfimo en la dimensión *Geografía*, y la versión 3. Esta última es la versión actual, lo que puede explicarse debido a que no tiene un atributo *instant To* definido.

```
<?xml version="1.0" encoding="UTF-8"?>
<metaFactTables xmlns="http://www.example.com/MetaFactTables"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.example.com/MetaFactTables  metaFactTables.xsd">
    <metaFactTable name="Prestamos" type="A" granularity="s" measure="amount">
        <versions>
            <version number="1" instantFrom="2003-01-01T00:00:01"
                                instantTo="2003-12-31T23:59:59">
                <levels>
                    <level dimension="Asistencias" name="Asistencia">
                    </level>
                    <level dimension="Deudores" name="CUIT">
                    </level>
                    <level dimension="EntidadesFinancieras" name="EntidadId">
                    </level>
                    <level dimension="Geografia" name="Region">
                    </level>
                    <level dimensión="Situaciones" name="SituacionId">
                    </level>
                    <level dimension="Tiempo" name="mes">
                    </level>
                </levels>
            </version>
            <version number="2" instantFrom="2004-01-01T00:00:01"
                                instantTo="2004-12-31T23:59:59">
                <levels>
                    <level dimension="Asistencias" name="Asistencia">
                    </level>
                    <level dimension="Deudores" name="CUIT">
                    </level>
                    <level dimension="EntidadesFinancieras" name="EntidadId">
                    </level>
                    <level dimension="Geografia" name="Idprovincia">
                    </level>
                    <level dimension="Situaciones" name="SituacionId">
                    </level>
                    <level dimension="Tiempo" name="mes">
                    </level>
                </levels>
            </version>
            <version number="3" instantFrom="2005-01-01T00:00:01">
                <levels>
                    <level dimension="Asistencias" name="Asistencia">
                    </level>
                    <level dimension="Deudores" name="CUIT">
                    </level>
                    <level dimension="EntidadesFinancieras" name="EntidadId">
                    </level>
                    <level dimension="Geografia" name="localidadId">
                    </level>
                    <level dimension="Situaciones" name="SituacionId">
                    </level>
                    <level dimension="Tiempo" name="dia">
                    </level>
                </levels>
            </version>
        </versions>
    </metaFactTable>
</metaFactTables>
```

Nótese que para la dimensión *Geografía*, en la versión 1 el atributo correspondiente es *región*, en la versión 2 es *Idprovincia* y en la versión 3 *localidadId*. Esto es debido a dos aplicaciones sucesivas del operador *Specialize* sobre dicha dimensión.

3.3 Tipos de Datos TOLAP

Nuestra implementación soporta la jerarquía de tipos de datos descripta en la Figura 3.9.

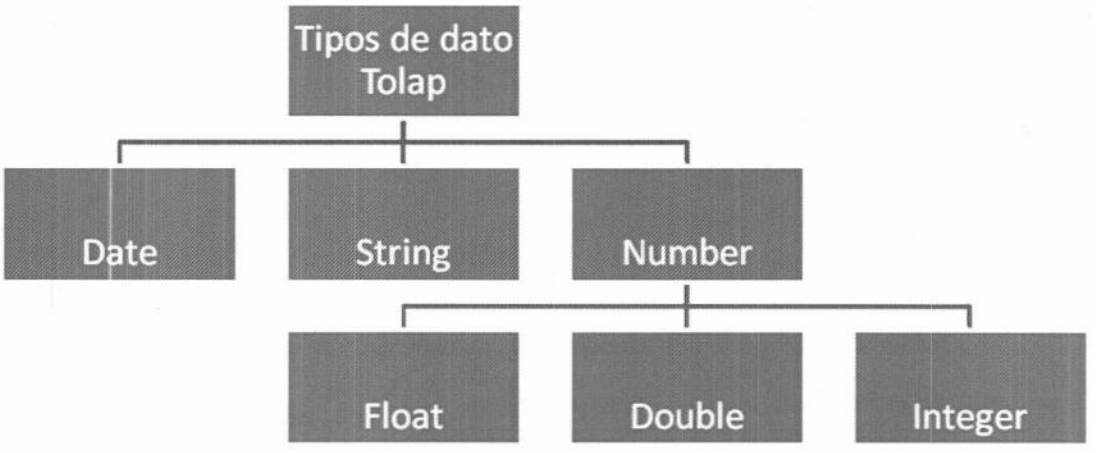


Figura 3.8 Jerarquía de tipos de datos

La implementación de tipos de datos es independiente del RDBMS utilizado, debido a que la correspondencia con el tipo de datos de cada posible motor a utilizar se encuentra parametrizada por medio de entradas en el archivo de configuración de *TOLAP* denominado `parameters.xml`.

Por ejemplo en el caso de utilizar una BD SQL-Server 2000 se introducen las siguientes entradas:

```
<parameter name="TOLAP_TYPE_INTEGER_DB" value="INTEGER"/>
<parameter name="TOLAP_TYPE_FLOAT_DB" value=" NUMERIC(12,2)"/>
<parameter name="TOLAP_TYPE_DOUBLE_DB" value="NUMERIC(20,4)"/>
<parameter name="TOLAP_TYPE_STRING_DB" value="VARCHAR(2000)"/>
<parameter name="TOLAP_TYPE_DATE_DB" value="DATETIME"/>
```

Mientras que en Teradata V2R5 podría utilizarse

```
<parameter name="TOLAP_TYPE_INTEGER_DB" value=" DECIMAL(10,0)"/>
<parameter name="TOLAP_TYPE_FLOAT_DB" value=" DECIMAL(12,2)"/>
<parameter name="TOLAP_TYPE_DOUBLE_DB" value="DECIMAL(20,4)"/>
<parameter name="TOLAP_TYPE_STRING_DB" value="VARCHAR(2000)"/>
<parameter name="TOLAP_TYPE_DATE_DB" value=" TIMESTAMP(22)"/>
```

Nótese que la precisión de cada tipo es definida a través de la interfaz en los casos en que el usuario así lo indique.

3.4 Metadatos SQL

Además de las relaciones representadas por cada *dimensión* en el datawarehouse, existen otras tablas necesarias para almacenar metadatos:

- **TDIMENSION(Dimension_Id,name,description,granularity,modo,tipo,instant_from, instant_to)**
  Contiene la información de los esquemas de dimensión en el sistema.
- **TLEVEL(dimension_Id, level_Id, name, ttype)**
  Indica los niveles que contienen cada Dimensión.
- **TLEVEL_INTERVAL(level_Id, instant_from , instant_to)**
  Para cada nivel, indica su intervalo de validez.
- **TATTRIBUTE(level_Id, attribute_Id,name, ttype):**
  Contiene los atributos temporales para cada nivel en la jerarquía de la dimensión.
- **TATTRIBUTE_INTERVAL(attribute_Id, instant_from , instant_to)**
  Detalla a los intervalos temporales de validez de un atributo.
- **DIMENSION_XXX(instant_from , instant_to, Level_ALL, ...)**
  Contiene las instancias de la dimensión XXX.
- **TROLLUP(level_Id_From, level_Id_To, instant_from , instant_to)**
  Representa la historia de los esquemas de dimensiones
- **TINSTANCE_ATTRIBUTE_XX(attribute_Id, value, instant_from , instant_to, value_at)**
  Contiene los valores de los atributos de instancias a lo largo del tiempo, donde la primer 'X' identifica el tipo de datos de la instancia a la cual pertenece el atributo y la segunda 'X' el tipo de datos del atributo. Por ejemplo, en nuestro modelo de créditos, que presentaremos en detalle más adelante, el valor del atributo de instancias "Denominación" del Nivel "CUIT" en la dimensión

3.6 Validación de Documentos XML

Para asegurar en todo momento la integridad del Sistema se utilizó XML Schema Documents Validation[1] (cuya especificación fue definida por la W3C[2]), definiendo para cada uno de los documentos XML que representan los metadatos en *TOLAP* su correspondiente esquema de validación.

En el Anexo 1, se muestran los distintos XML schemas (archivos xsd) que se utilizan para validar la formación tanto sintáctica como semántica de los documentos XML.

La siguiente tabla indica el nombre del Documento XSD utilizado para cada tipo de documento XML:

XML Schema (XSD)	Aplica al Documento XML
dimSchemas.xsd	Valida dimSchemas.xml, que contiene el catálogo de todas las Dimensión, uno para todo *TOLAP*.
dimension.xsd	Valida los documentos XML que representan metadata de una Dimensión, un XML por dimensión.
MetaFactTables.xsd	Valida metaFactTables.xml, que contiene el catálogo de todas la metadata de las tablas de hechos y sus distintas versiones, uno para todo *TOLAP*.
funcion.xsd	Valida los documentos XML utilizados para especificar las rollups entre instancias, necesario para aplicar los operadores, uno por Operador.

Cada documento xsd contiene distintos tipos complejos, en algunos casos anidados, para representar la estructura XML correspondiente, cada uno con su orden correspondiente en el documento XML, y desde el punto de vista semántico, algunas reglas de restricciones sobre dominio de valores para elementos o atributos en particular.

[1] Define qué elementos puede contener un documento XML, cómo están organizados y qué atributos y de qué tipo pueden tener sus elementos. XML Schema es una especificación mantenida por el W3C y Se encuentra disponible en la siguiente URL: http://www.w3.org/XML/Schema

[2] Abreviatura de World Wide Web Consortium (W3C)

Para la implementación de XML Schema fue utilizado el package Oracle XML Schemas[1]. Al momento de abrir el documento XML correspondiente, *TOLAP* automáticamente realiza el pedido de validación con el documento xsd relacionado.

A modo de ejemplo, consideremos uno de los documentos más complejos definidos para *TOLAP*, que es el que valida la estructura de los archivos de las tablas de dimensiones, denominado dimension.xsd. El archivo dimension.xsd está compuesto de varias partes, que detallamos a continuación, para luego producir un documento XML que resulte exitosamente validado por el XML schema.

1. Encabezado conforme a la especificación XSD (XML Schema Definition), indicando el espacio de nombres (namespace)

```
<schema targetNamespace="http://www.example.com/Dimension"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:d="http://www.example.com/Dimension"
        elementFormDefault="qualified">
```

2. El elemento `metadata` que contiene el tipo complejo `Dimension`.

```
<element name="metadata">
   <complexType>
      <sequence>
         <element name="dimension" type="d:dimensionType" minOccurs="0"/>
      </sequence>
   </complexType>
</element>
```

3. Complex Type `dimensionType`: incluye los tipos complejos *levels* y *rollups*, en ese orden, incluidos en el elemento sequence, y con como mínimo un nivel y una función de rollup (en la dimensión inicial del nivel mínimo al nivel All). Como atributos XML del elemento dimensión, se definen name, description, granularity, modo, tipo, instantFrom e instantTo (vigencia de la dimensión). Se pueden observar las distintas restricciones para el dominio de valores para granularity, modo y tipo.

[1] Más información sobre este package puede obtenerse en: http://www.oracle.com/technology/tech/xml/index.html

```
<complexType name="dimensionType">
   <sequence>
      <element name="levels" type="d:levelsType" minOccurs="1"/>
      <element name="rollup" type="d:rollupType" minOccurs="1"/>
   </sequence>
   <attribute name="name" type="string"/>
   <attribute name="description" type="string"/>
   <attribute name="granularity">
      <simpleType>
         <restriction base="string">
            <annotation>
               <documentation>Y Year - M Month - D Date - h Hour - m Minute -
                                                                    s Second
               </documentation>
            </annotation>
            <enumeration value="Y"/>
            <enumeration value="M"/>
            <enumeration value="D"/>
            <enumeration value="h"/>
            <enumeration value="m"/>
            <enumeration value="s"/>
         </restriction>
      </simpleType>
   </attribute>
   <attribute name="modo">
      <simpleType>
         <restriction base="string">
            <enumeration value="I"/>
            <enumeration value="N"/>
         </restriction>
      </simpleType>
   </attribute>
   <attribute name="tipo">
      <simpleType>
         <restriction base="string">
            <enumeration value="T"/>
            <enumeration value="G"/>
         </restriction>
      </simpleType>
   </attribute>
   <attribute name="instantFrom" type="timeInstant"/>
   <attribute name="instantTo" type="timeInstant" use="optional"/>
</complexType>
```

4. Complex type `intervalType`: se utiliza para definir los intervalos de niveles y rollups.

```
<complexType name="intervalType">
   <sequence>
      <element name="interval" minOccurs="0" maxOccurs="unbounded">
         <complexType>
            <attribute name="instantFrom" type="timeInstant"/>
            <attribute name="instantTo" type="timeInstant" use="optional"/>
         </complexType>
      </element>
   </sequence>
</complexType>
```

5. Complex type `attributeIntervalType`: para las intervalos de atributos *TOLAP*.

```
<complexType name="attributeIntervalType">
   <sequence>
      <element name="attributeInterval" minOccurs="1" maxOccurs="unbounded">
         <complexType>
            <attribute name="instantFrom" type="timeInstant"/>
            <attribute name="instantTo" type="timeInstant" use="optional"/>
         </complexType>
      </element>
   </sequence>
</complexType>
```

6. Complex type `attributeType`: para un attribute *TOLAP*, con el intervalo temporal, su nombre y su TOLAP datatype (con constraint).

```
<complexType name="attributeType">
   <sequence>
      <element name="attribute" minOccurs="1" maxOccurs="unbounded">
         <complexType>
            <sequence>
               <element name="attributeIntervals"
                                 type="d:attributeIntervalType"
                                 minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="name" type="string"/>
            <attribute name="ttype">
               <simpleType>
                  <restriction base="string">
                     <annotation>
                        <documentation>I Integer - F Float - D Double - S String -
                                                T Time Instant (datetime)
                        </documentation>
                     </annotation>
                     <enumeration value="I"/>
                     <enumeration value="F"/>
                     <enumeration value="D"/>
                     <enumeration value="S"/>
                     <enumeration value="T"/>
                  </restriction>
               </simpleType>
            </attribute>
         </complexType>
      </element>
   </sequence>
</complexType>
```

"Deudores" se guarda en la tabla TINSTANCE_ATTRIBUTE_NS debido a que se trata de una instancia de tipo Number que contiene un atributo de instancia de tipo String.

Esta decisión de implementación fue tomada con el objeto de optimizar el modelo evitando la necesidad de realizar un Casting de datos durante la ejecución de las consultas TOLAP.

- **TMETA_FACT_TABLE(mft_Id, name, granularity, measure)**
  Contiene la definición de las tablas de hechos básicas del Sistema (independientemente de su vigencia temporal).

- **TMETA_FACT_TABLE_VERSION(mft_Id, mftv_Id, version, instant_from , instant_to)**
  Contiene la definición de las versiones existentes para cada tabla de hechos.

- **TMETA_FACT_TABLE_LEVELS(mft_Id, mftv_Id, level_Id)**
  Contiene los niveles de cada dimensión asociados a cada versión de una tabla de hechos.

La Figura 3.9 muestra el diagrama Entidad-Relación de los Metadatos TOLAP almacenados en el servidor de Base de Datos.

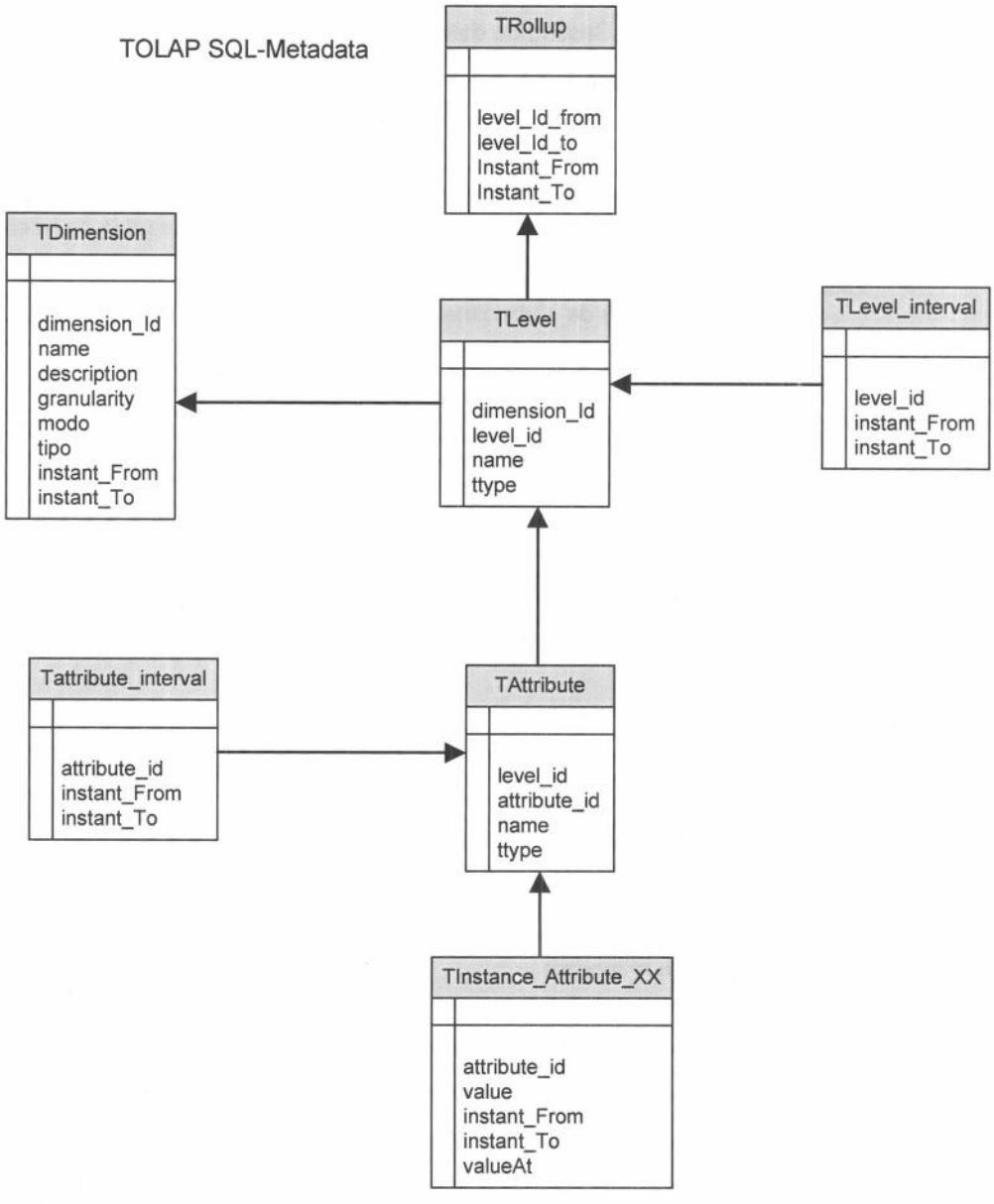
3.5 Metadatos XML

Los metadatos *TOLAP* almacenados como documentos XML consisten en:

- Un documento XML (DimSchemas.xml) conteniendo la definición de las *dimensiones* creadas en el modelo.

- Para cada una de las *dimensiones* en DimSchemas.XML existe un archivo XML (referenciado como: zzz.xml donde zzz es nombre de la dimensión) conteniendo las definiciones inherentes a dicha dimensión (datos básicos, niveles que la componen, atributos de cada nivel, Tipos Tolap involucrados, Rollups definidas, etc).

- Un documento XML (Metafactables.xml) conteniendo los datos básicos de las tablas de hechos definidas así como también los niveles que la integran.



(1): Los valores posibles para "XX" son: DD, DN, DS, ND, NN, NS, SD, SN y SS dependiendo del tipo de datos del campo "value" y "valueAt"

Figura 3.9. Diagrama de Entidad-Relación de los Metadatos SQL de TOLAP

7. Complex type `levelsType`: un nivel puede contener atributos, intervals donde tiene vigencia y su tipo de datos *TOLAP*.

```
<complexType name="levelsType">
   <sequence>
      <element name="level" minOccurs="0" maxOccurs="unbounded">
         <complexType>
            <sequence>
               <element name="attributes" type="d:attributeType" minOccurs="0"
                                                  maxOccurs="unbounded"/>
               <element name="intervals" type="d:intervalType" minOccurs="0"
                                                  maxOccurs="unbounded"/>
            </sequence>
            <attribute name="name" type="string"/>
            <attribute name="ttype">
               <simpleType>
                  <restriction base="string">
                     <annotation>
                        <documentation>I Integer - F Float - D Double - S String -
                                                      T Time Instant (datetime)
                        </documentation>
                     </annotation>
                     <enumeration value="I"/>
                     <enumeration value="F"/>
                     <enumeration value="D"/>
                     <enumeration value="S"/>
                     <enumeration value="T"/>
                  </restriction>
               </simpleType>
            </attribute>
         </complexType>
      </element>
   </sequence>
</complexType>
```

8. Complex type `rollupType`: una rollup contiene un level from, un nivel to y un intervalo de vigencia de esa rollup.

```
<complexType name="rollupType">
   <sequence>
      <element name="levelFrom" minOccurs="0" maxOccurs="unbounded">
         <complexType>
            <sequence>
               <element name="levelTo" minOccurs="0" maxOccurs="unbounded">
                  <complexType>
                     <sequence>
                        <element name="intervals"
                                          type="d:intervalType" minOccurs="0"
                                              maxOccurs="unbounded"/>
                     </sequence>
                     <attribute name="name" type="string"/>
                  </complexType>
               </element>
            </sequence>
            <attribute name="name" type="string"/>
         </complexType>
      </element>
   </sequence>
</complexType>


</schema>
```

El siguiente es un ejemplo de documento XML válido que representa la estructura de la dimensión *Geografía* de la Figura 3.10, a lo largo de su lifespan.:

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata xmlns="http://www.example.com/Dimension"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.example.com/Dimension  dimension.xsd">
    <dimension name="Geografia" description="Geografia" granularity="s" modo="I"
                                tipo="T" instantFrom="2006-02-18T15:06:00">
        <levels>
            <level name="ciudad" ttype="S">
                <intervals>
                    <interval instantFrom="2006-02-18T15:06:00">
                    </interval>
                </intervals>
                <attributes>
                    <attribute name="nombre" ttype="S">
                        <attributeIntervals>
                            <attributeInterval instantFrom="2006-02-18T15:21:00">
                            </attributeInterval>
                        </attributeIntervals>
                    </attribute>
                    <attribute name="poblacion" ttype="D">
                        <attributeIntervals>
                            <attributeInterval instantFrom="2006-02-18T15:29:57">
                            </attributeInterval>
                        </attributeIntervals>
                    </attribute>
                </attributes>
            </level>
            <level name="All" ttype="S">
                <intervals>
                    <interval instantFrom="2006-02-18T15:06:00">
                    </interval>
                </intervals>
            </level>
            <level name="region" ttype="S">
                <intervals>
                    <interval instantFrom="2006-02-18T15:06:09">
                    </interval>
                </intervals>
            </level>
            <level name="pais" ttype="S">
                <intervals>
                    <interval instantFrom="2006-02-18T15:06:18">
                    </interval>
                </intervals>
            </level>
            <level name="provincia" ttype="S">
                <intervals>
                    <interval instantFrom="2006-02-18T15:06:28">
                    </interval>
                </intervals>
            </level>
            <level name="pueblo" ttype="S">
                <intervals>
                    <interval instantFrom="2006-02-18T15:17:07">
                    </interval>
                </intervals>
            </level>
            <level name="manzana" ttype="S">
                <intervals>
                    <interval instantFrom="2006-02-18T15:17:21">
                    </interval>
```

```
        </intervals>
    </level>
</levels>
<rollup>
    <levelFrom name="ciudad">
        <levelTo name="All">
            <intervals>
                <interval instantFrom="2006-02-18T15:06:00"
                          instantTo="2006-02-18T15:06:08">
                </interval>
            </intervals>
        </levelTo>
        <levelTo name="region">
            <intervals>
                <interval instantFrom="2006-02-18T15:06:09">
                </interval>
            </intervals>
        </levelTo>
        <levelTo name="provincia">
            <intervals>
                <interval instantFrom="2006-02-18T15:06:28">
                </interval>
            </intervals>
        </levelTo>
    </levelFrom>
    <levelFrom name="region">
        <levelTo name="All">
            <intervals>
                <interval instantFrom="2006-02-18T15:06:09"
                          instantTo="2006-02-18T15:06:17">
                </interval>
            </intervals>
        </levelTo>
        <levelTo name="pais">
            <intervals>
                <interval instantFrom="2006-02-18T15:06:18">
                </interval>
            </intervals>
        </levelTo>
    </levelFrom>
    <levelFrom name="pais">
        <levelTo name="All">
            <intervals>
                <interval instantFrom="2006-02-18T15:06:18">
                </interval>
            </intervals>
        </levelTo>
    </levelFrom>
    <levelFrom name="provincia">
        <levelTo name="All">
            <intervals>
                <interval instantFrom="2006-02-18T15:06:28"
                          instantTo="2006-02-18T15:06:42">
                </interval>
            </intervals>
        </levelTo>
        <levelTo name="pais">
            <intervals>
                <interval instantFrom="2006-02-18T15:06:43">
                </interval>
            </intervals>
        </levelTo>
    </levelFrom>
    <levelFrom name="pueblo">
        <levelTo name="ciudad">
            <intervals>
                <interval instantFrom="2006-02-18T15:17:07">
                </interval>
```

```
                    </intervals>
                </levelTo>
            </levelFrom>
            <levelFrom name="manzana">
                <levelTo name="pueblo">
                    <intervals>
                        <interval instantFrom="2006-02-18T15:17:21">
                        </interval>
                    </intervals>
                </levelTo>
            </levelFrom>
        </rollup>
    </dimension>
</metadata>
```

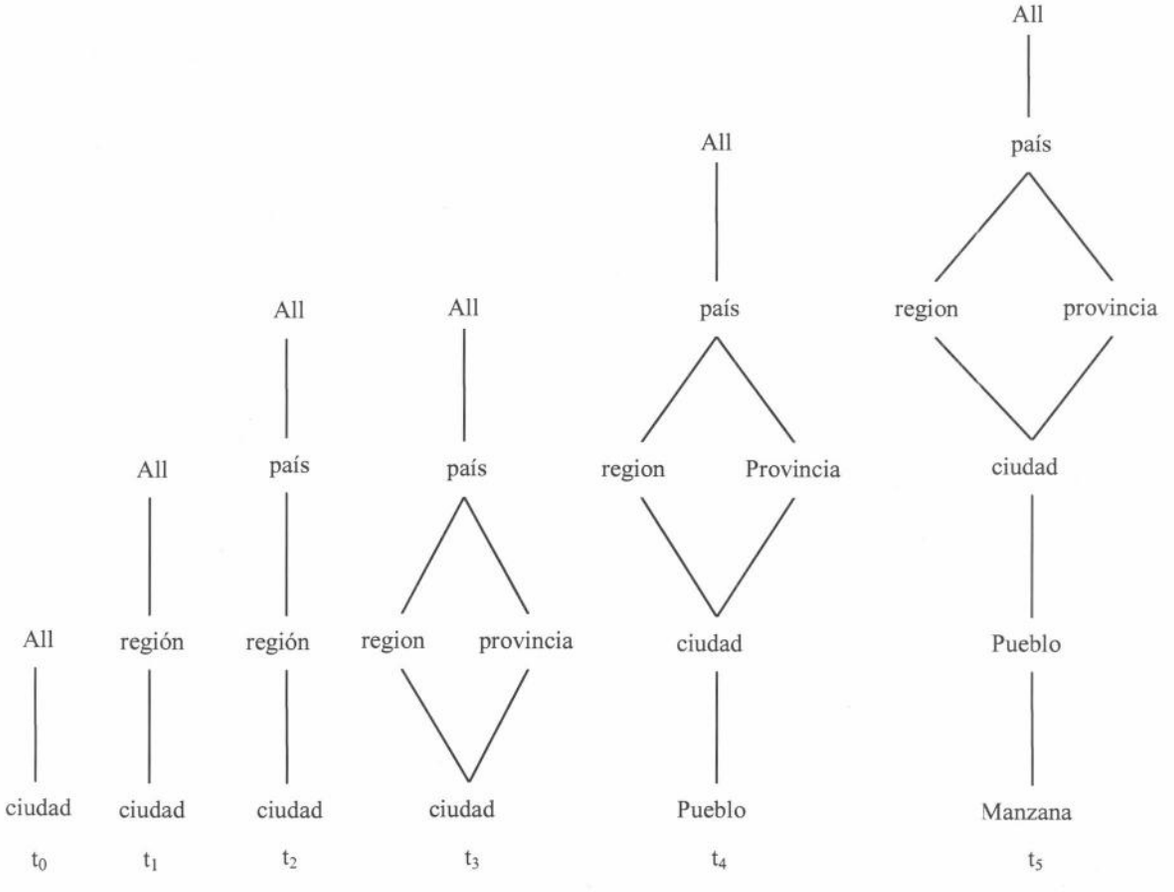


Figura 3.10. Cambios del Esquema de la Dimensión Geografía

3.7 Coordinación SQL/XML

Debido a que tanto los Metadatos XML como los Metadatos SQL contienen la información estructural del modelo *TOLAP*, se hace necesario que, ante la aplicación de un operador *TOLAP* estructural ambos reflejen los mismos cambios estructurales.

Se ofrece la opción a nivel administrador *TOLAP* de que se regeneren los XMLs a partir del modelo de datos relacional cargado en la base de datos. El camino inverso, de XML a la base de datos, debe realizarse en forma manual, es decir, no existe un proceso definido para que desde los XMLs pertenecientes a los metadatos se inserten como registros en las tablas del modelo de base de datos coordinado. Si se pierde la instancia de base de datos, también se perderían la metadata y los datos del modelo *TOLAP* para realizar consultas, por lo que no tiene mucho sentido solamente recuperar la metadata hacia la base de datos, pues siempre resultaría incompleto.

3.7 Resumen.

En este capítulo, se describieron los conceptos de Dimensiones y Tablas de Hechos. Luego se mostró como la información estructural de los mismos es representada en la Metadata TOLAP. Se presentó la jerarquía de tipos de datos soportados y cómo se efectúa la correspondencia con los que maneja cada motor de base de datos.

Posteriormente, se explicó que los metadatos TOLAP se dividen en Metadatos SQL y Metadatos XML y porqué se decidió implementarlos de esa forma. Por último, se mostró como se realiza el manejo de los Metadatos en XML y su coordinación con los que residen en el motor de base de datos.

Capítulo 4.

EL LENGUAJE DE CONSULTAS TOLAP-QL

TOLAP-QL es un lenguaje estilo SQL que permite expresar consultas sobre un modelo de datos *TOLAP*. El lenguaje *TOLAP-QL* tiene una sintaxis que va a resultar familiar a los usuarios normales de SQL. Las consultas expresadas en *TOLAP-QL* pasan por un proceso de parsing y finalmente se las traduce a sentencias SQL estándares.

4.1 Modelo TOLAP ejemplo para las consultas

Para ejemplificar las características y posibilidades de *TOLAP-QL* utilizaremos una serie de consultas sobre dos ejemplos tomados de [Vai01] que describiremos a continuación. Los siguientes diagramas muestra las distintas tablas de hechos y de dimensiones que componen estos dos modelos: el modelo de Ventas y un modelo de Salud.

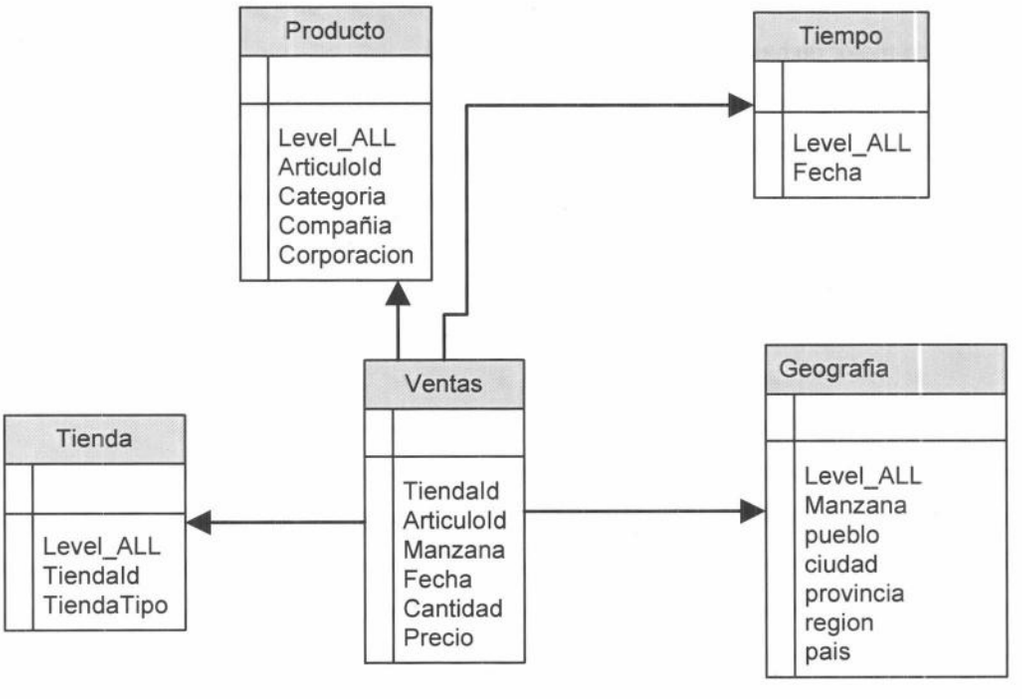


Figura 4.11. Modelo de Ventas

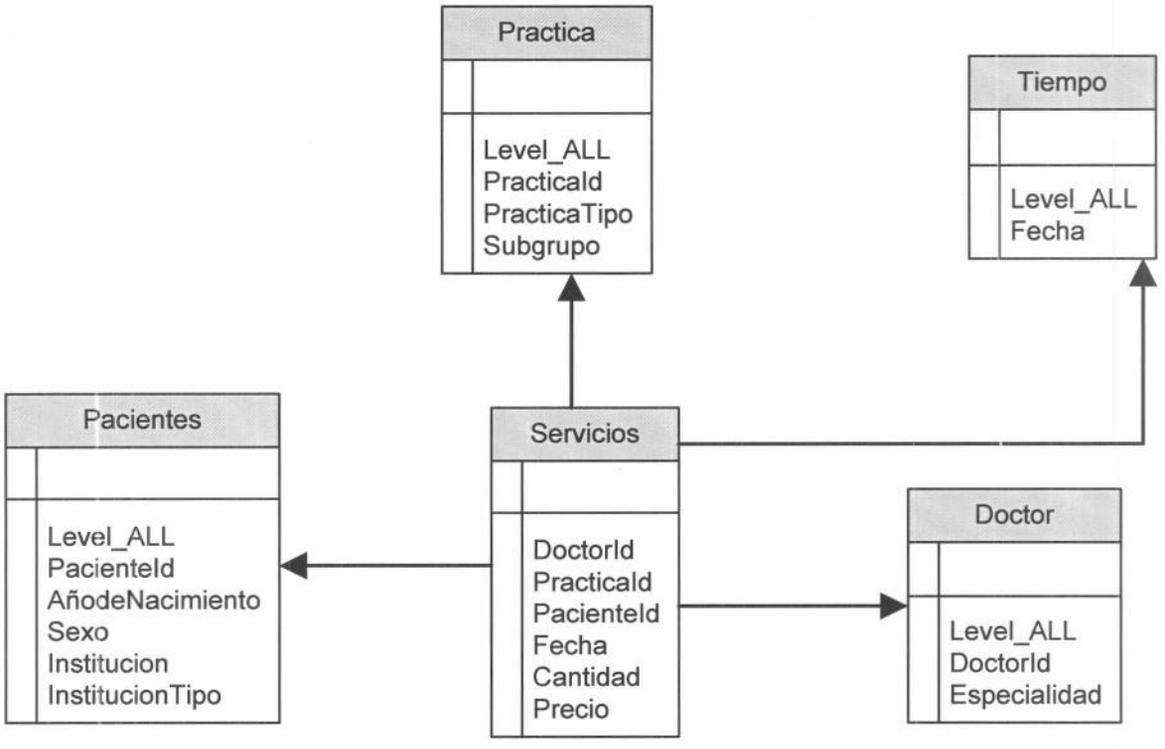


Figura 4.12. Modelo de Salud

4.2 Sintaxis General

La sintaxis general de una consulta *TOLAP-QL* tiene la forma:

```
SELECT [{column1}, {column2}, ..., {columnN}, {Aggregate Function(measure)}]
  FROM Dimension1, [Dimension2, ..., Dimension N], [FactTable]
[WHERE expression1 [AND expression2 ... ]]
```

4.3 Tipos de consultas TOLAP-QL

En general, podemos decir que existen dos tipos de consultas posibles:

- Una Metaconsulta es una consulta realizada sobre una dimensión para poder averiguar aspectos estructurales sobre la misma, por ejemplo, si el rollup entre dos niveles existe al instante t.

  - Obtener información estructural a través del tiempo, por ejemplo:

    Los instantes t en los que la Tienda 's1' pertenece a la Región 'Sur'

```
SELECT t
  FROM Tienda
  WHERE RUP[1]( Tienda.TiendaId:'s1',region:'Sur',t);
```

  - Obtener un valor de verdad a partir de las expresiones que se quieran plantear, por ejemplo:

    Si existe la rollup entre el ínfimo de la dimensión Producto y el nivel categoría al instante 1/1/2006

```
SELECT boolean
  FROM Producto
  WHERE RUP(Producto,categoria,'1/1/2006');
```

- Consultas propiamente dichas, son las que se realizan sobre dimensiones y tablas de hechos para obtener información, por lo general aplicando una función de agregación sobre la medida de la tabla de hechos interviniente, por ejemplo:

  Monto total vendido por categoría de producto para la región 'r1'

```
SELECT P.categoria, SUM(amount)
 FROM Ventas F, Producto P, Geografía G
 WHERE F.Producto = P.bottom
  AND F.Geografia = G.bottom
  AND RUP(G,region:'r1',F.t)
  AND RUP(P,categoria,F.t);
```

[1] Descripta en 4.4

La consulta, expresada de esta forma en TOLAP-QL, es interpretada como la pregunta por las ventas de productos, agrupados por la categoría a la que pertenecían al instante de la venta siendo que la misma se produjo en la región 'r1'.

4.4 El predicado RUP

El predicado *RUP* (Rollup) refleja las funciones de rollup definidas en los esquemas de dimensión. En la sección 4.3, detallamos la forma en que se expresa en TOLAP-QL la consulta: *Monto total vendido por categoría de producto para la región r1.* En dicha consulta, las expresiones *RUP* tienen el siguiente significado: Las dimensiones *Geografía* y *Producto* son componentes de la tabla de hechos *Ventas.* El usuario no necesita conocer cuales son los actuales niveles ínfimos de la tabla de hechos *Ventas*, porque esos niveles pueden cambiar a lo largo del tiempo. Sin embargo, sabemos que las tablas de dimension y de hechos deben siempre estar relacionadas a traves de los niveles ínfimos.

La sintaxis de *RUP* se puede resumir en la siguiente especificación:

```
RUP(Dim.levelInf: [levelInf.value | levelInf.variable],
    Dim.levelSup: [levelSup.value | levelSup.variable],
   time, [instanceAlias])
```

Si `Dim.levelInf` es el nivel ínfimo de la dimensión solo se indica el nombre de la dimensión.

La semántica se indica en el cuadro siguiente:

Expresión	Descripción
`RUP(P,categoria,F.t)`	Devuelve *verdadero* si existe un nivel ínfimo en la dimensión con alias P con rollup al nivel *categoria* en el instante F(t)
`RUP(G,region:'r2','16/01/2001 14:00:00')`	Devuelve *verdadero* si existe un nivel ínfimo en la dimensión con alias G con rollup al nivel *región* para la instancia r2 en el instante fijo.
`RUP(Pa,sexo,NOW)`	Devuelve *verdadero* si existe un nivel ínfimo en la dimensión con alias Pa con rollup al nivel *sexo* al instante actual (uso de NOW)
`RUP(C.ClienteId,VAR X: VAR x,'1/1/98')`	Devuelve *verdadero* si existe rollup desde el nivel *ClienteId* de la dimensión con alias C a un nivel e instancias variables en un instante fijo ('1/1/98')
`RUP(D, CUIT:c1, P.t, c)`	Devuelve *verdadero* si existe un nivel ínfimo en la dimensión con alias D con rollup al nivel *cuit* en el instante F(t) desde la instancia c1 a la instancia c

Es importante tener en cuenta que existe, en las consultas en las que intervienen *dimensiones* y *tablas de hechos*, una relación entre el JOIN efectuado entre ellas y las cláusulas *RUP*.

Veamos la siguiente consulta:

Monto total vendido por categoría de producto para la región 'r1'.

Que en *TOLAP-QL* se expresa:

```
SELECT P.categoria, SUM(monto)
  FROM Ventas F, Producto P, Geografia G
 WHERE F.Producto = P.bottom
   AND F.Geografia = G.bottom
   AND RUP(G,region:'r1',F.t)
   AND RUP(P,categoria,F.t);
```

Si se toma en cuenta que la expresión RUP(P,categoria,F.t) es verdadera si existe una rollup desde el ínfimo del alias de dimensión *P* (en este caso *Producto*) al nivel *categoría* en un instante t, es necesario tener en cuenta al realizar la traducción que la condición de junta entre la tabla de hechos y las dimensiones se realiza a través de los ínfimos de cada dimensión vigentes a ese instante.

Debido a que en nuestra gramática TOLAP-QL, la condición de junta entre la tabla de hechos y las dimensiones debe escribirse explícitamente, siempre que se indica una cláusula RUP en la consulta escrita en TOLAP-QL, las siguientes expresiones se incluirán en la misma:

```
JOIN: F.Producto = P.bottom
RUP : RUP(P,categoria,F.t)
```

La traducción de *TOLAP-QL* a SQL, es la siguiente:

```
JOIN: F.Producto = P.productoId (el ínfimo se traduce al nivel correspondiente)
RUP : F.instant BETWEEN CASE WHEN P.instant_to IS NULL
                         THEN GETDATE() ELSE P.instant_to
```

De no especificarse el JOIN, el sistema realizaría un producto cartesiano, con un efecto colateral no deseado. Si el predicado *RUP* no fuera especificado, no se realizaría la inclusión del instante de la tabla de hechos en el intervalo de la *dimensión*, generando una selección de instancias quizás inválida.

4.5 Bloques

TOLAP-QL brinda la posibilidad de especificar bloques para definir una sentencia con alcance local, es decir, las variables locales a esa expresión son válidas solo en dicho contexto y no fuera de él, pudiendo hacerse referencia a las variables de alcance anidadas.

La implementación de *TOLAP-QL*, maneja alcances anidados dando visibilidad a las variables locales y a todas aquellas definidas en un contexto de orden superior.

Por ejemplo, la siguiente consulta muestra la potencia de las consultas temporales.

Obtener la cantidad de servicios realizados por semana por el doctor Martínez, en los periodos en los cuales el doctor Feinsilver trabajaba en la clínica.

Que en TOLAP-QL se escribe como:

```
SELECT T.semana, SUM(qty)
  FROM Tiempo T, Servicios F, Doctor D
 WHERE F.Tiempo = T.bottom
   AND F.Doctor = D.bottom
   AND RUP(D,doctorId:d,F.t) AND d.nombre='Martinez'
   AND (RUP(D,doctorId:d1,F.t,d) AND d1.nombre='Feinsilver');
```

En este caso, d corresponde al alcance más global del WHERE y la variable d1 solamente es válida en el alcance del bloque que está entre paréntesis.

Como F.t es el mismo para las dos instancias, semánticamente significa que F.t tiene que ser el mismo tanto para la variable d como para la variable d1, es decir, los eventos van a coincidir en su timestamp pero dichas variables no aparecen ligadas entre sí.

Las variables d y d1 podrían incluso tener el mismo nombre, ya que si nos referimos a d en el WHERE va a ser la variable de definida en la primera *RUP*, y si cambiamos el nombre d1 por d, y estando dentro del bloque, d va referirse a la de el mismo y no a la variable d más global que su alcance.

El traductor va a realizar automáticamente una subconsulta correlacionada agregando una sentencia SQL EXISTS mostrando que el valor de F.t va a ser el mismo para la instancia dentro del bloque.

4.6 Modificador NOT

El modificador NOT se aplica a un bloque cuando se quiere indicar que no se cumplen las condiciones asociadas al mismo. Por lo general se aplica a predicados *RUP* o combinación de expresiones indicadas entre paréntesis.

Veamos un ejemplo

Obtener la cantidad de prácticas, discriminadas por sexo, que se han realizado a la fecha los pacientes que aún no han realizado una práctica 'i1'.

Que en *TOLAP-QL* se escribe:

```
SELECT Pa.sexo, SUM(S.qty)
 FROM Servicios S, Paciente Pa, Practica P
WHERE S.Practica=P.bottom
  AND S.Paciente=Pa.bottom
  AND NOT RUP(P,practicaId:'i1',NOW)
  AND RUP (Pa,sexo,NOW);
```

En esta consulta, **`NOT RUP(P,practicaId:'i1',NOW)`** indica que no existe en el instante NOW, una función de rollup desde el ínfimo actual de la dimensión P hacia la instancia 'i1' del nivel practicaId.

También puede utilizarse para negar bloques, por ej., en la consulta anterior de bloques se puede negar el bloque haciendo que no se cumplan las condiciones sobre el predicado *RUP*.

Obtener la semana y la cantidad de servicios realizados por el doctor Martínez mientras el doctor Feinsilver ***no estaba trabajando (en el período de aquél).***

```
SELECT T.semana, SUM(qty)
  FROM Tiempo T, Servicios F, Doctor D
 WHERE F.Tiempo = T.bottom
   AND F.Doctor = D.bottom
   AND RUP(D,doctorId:d,F.t) AND d.nombre='Martinez'
AND NOT (RUP(D,doctorId:d1,F.t,d) AND d1.nombre='Feinsilver');
```

4.7 Redundancia de RUPs

En el caso de las consultas donde intervienen dimensiones y la tabla de hechos, es necesario especificar las *RUPs* sobre los niveles que se van a publicar como columnas en la sentencia SELECT.

Por ejemplo, en el siguiente caso entre las consultas:

a)

```
SELECT P.ArticuloId, S.region, S.pais, SUM(precio)
  FROM Ventas F, Producto P, Tienda S
 WHERE F.Producto=P.bottom
   AND F.Tienda=S.bottom
   AND RUP(S,region,F.t)
   AND RUP(S,tiendaId,F.t)
   AND RUP(P,categoria,F.t);
```

b)

```
SELECT P. ArticuloId, S.region, S.pais, SUM(precio)
  FROM Ventas F, Producto P, Tienda S
 WHERE F.Producto=P.bottom
   AND F.Tienda=S.bottom
   AND RUP(S,region,F.t)
   AND RUP(P,categoria,F.t);
```

Las consultas a) y b) son equivalentes, van a arrojar el mismo resultado, es redundante poner como en a) dos rollups para la misma dimensión S.

4.8 Cláusula SELECT sin especificar Columnas

Supongamos la siguiente expresión TOLAP (ver [Vai01]) que representa la consulta: *Para cada institución, listar su Tipo.*

```
X,x,Y,y ← Paciente:X:x[t]-> institucionTipo:y , Y='institucionTipo';
```

Donde,

X representa el nivel inferior de la dimensión Paciente en cada instante.

x son las instancias correspondientes al nivel X.

y son las instancias de institucionTipo relacionadas.

t está libre para todo la vigencia de la dimensión Paciente

Un ejemplo de lo que devolvería esta consulta sería el siguiente:

institucion	Amtrak	institucionTipo	Education
institucion	MGM	institucionTipo	Show Business

En TOLAP-QL esta expresión se escribe:

```
SELECT
  FROM Paciente P
  WHERE RUP(P, var Y, t) AND Y = 'institucionTipo';
```

La cual representa todas las rollups existentes en la dimensión paciente, para el nivel `institucionTipo` en los instantes en que este es válido.

En este caso se está realizando una consulta sobre los metadatos del Sistema (ya que no existe en la clausula FROM una declaración a la tabla de hechos), motivo por el cual el sistema interpreta que lo que se quiere conocer es como estaba compuesta la estructura de los metadatos a un instante t. Por lo tanto, a las consultas para las cuales no se les define nada en el select, se devuelve como resultado la función de rollup que satisface la condición del where.

4.9 Expresiones sobre atributos

Las expresiones sobre atributos se pueden especificar en la cláusula WHERE de la consulta, y dado que los atributos son temporales en la definición de sus valores, se puede utilizar un instante t como parámetro. Si no se especifica el valor, se asume por defecto F.t.

Total de ventas por artículo y región para las tiendas que tengan más de 90 empleados

```
SELECT P.articuloId, S.region, SUM(precio*cantidad)
  FROM Ventas F, Producto P, Tienda S
 WHERE F.Producto = P.bottom
   AND F.Tienda = S.bottom
   AND RUP(S,tiendaId:st,F.t)
   AND RUP(P,categoria,F.t)
   AND st.nbrEmp(F.t) >= 90;
```

En este ejemplo, el predicado `RUP(S,tiendaId:st,F.t)` indica que existe una función de rollup desde el bottom de la *dimensión Tienda* hacia la instancia 'st' del nivel tiendaId en el instante F.t y además se restringe dicha instancia a que tenga un valor igual o superior a 90 en el atributo nbrEmp (cantidad de empleados), donde se usa *st* como variable sobre el nivel *tiendaId*.

Relaciones complejas entre instancias y atributos

Se pueden realizar consultas más complejas teniendo en cuenta alias de instancias y valores de atributos, por ejemplo:

Obtener la cantidad de servicios semanales realizados por el doctor Martínez mientras el doctor Feinsilver no estaba trabajando.

```
SELECT T.semana, SUM(cantidad)
  FROM Tiempo T, Servicios F, Doctor D
 WHERE F.Tiempo = T.bottom
   AND F.Doctor = D.bottom
   AND RUP(D,doctorId:d,F.t) AND d.nombre='Martinez'
   AND NOT (RUP(D,doctorId:d1,F.t,d) AND d1.nombre='Feinsilver');
```

Nótese, que en esta consulta el átomo negado no esta ligado a la tabla de hechos. También nótese que el usuario define el tiempo por el cual desea realizar la agregación (semanas), la cual aplica a un nivel que no esta ligado a la tabla de hechos *Servicios.* Además los atributos de los nombres de los médicos tampoco se encuentran ligados a *Servicios.*

En el Anexo 4, se puede ver la traducción a lenguaje de SQL de esta consulta, en donde se aprecia el poder de expresividad que tiene TOLAP-QL.

4.10 Programas TOLAP-QL

Los programas *TOLAP-QL* están conformados por un conjunto de sentencias SELECT, pudiendo almacenar los resultados intermedios a través de la cláusula STORE.

Especificación de la cláusula STORE

La cláusula STORE permite asignar un nombre a un resultado intermedio que va a ser reutilizado en otra consulta dentro del mismo programa.

Sintaxis:

```
{QUERY}
STORE AS nombre1;
[
[{QUERY2}
STORE AS nombre2;]
...
[{QUERYN}
STORE AS nombreN;]
]
{QUERY};
```

Veamos la siguiente consulta:

Para los periodos en los que realizaron más de 100 atenciones, listar la cantidad de prácticas realizadas por tipo de grupo.

En *TOLAP-QL, esta consulta* se escribe utilizando la cláusula STORE de la siguiente forma:

```
SELECT Pa.RangodeAño AS Año, SUM(F.cantidad) AS TotalQ1
  FROM Servicios F, Paciente Pa
 WHERE F.Paciente = Pa.bottom
   AND RUP(Pa, RangodeAño,F.t)
STORE AS Q1;

SELECT P.grp, SUM(F.qty)
  FROM Servicios F, Practica P, Paciente Pa,  Q1
 WHERE F.Practica = P.bottom
   AND F.Paciente = Pa.bottom
   AND RUP(P,grp,F.t)
   AND RUP(Pa:p, RangodeAño,F.t)
   AND Q1.Año = p.RangodeAño
   AND Q1.TotalQ1 >100;
```

Para que las columnas de una consulta con `STORE` puedan ser referenciadas por otra consulta, se deben publicar como alias por medio de la cláusula `AS`.

En la consulta anterior, que define el `STORE` Q1, se publica a la columna `Pa.RangodeAño` como `Año` y a `SUM(F.cantidad)` como `TotalQ1`.

En el `FROM` de la consulta se puede referenciar al nombre de cada `STORE` definido previamente en un bloque `SELECT`. Luego, se utilizan como una relación más. En el ejemplo, el resultado de la primera consulta se almacena con el nombre `Q1`, y se utiliza en la segunda consulta, mediante la inclusión de `Q1` en la cláusula `FROM`.

4.11 Resumen

En este capítulo presentamos a TOLAP-QL, el nuevo lenguaje de consultas para TOLAP que brinda el mismo poder expresivo que su antecesor, con el agregado de poseer una sintaxis más parecida a SQL y con una semántica más natural para el usuario. Explicamos los distintos elementos de este lenguaje y dimos ejemplos de cómo utilizar a los mismos en resolución de consultas.

Capítulo 5.

ARQUITECTURA DE SOFTWARE

Nuestra implementación permite al usuario crear y administrar un modelo *TOLAP* y realizar consultas sobre éste a través de la Web. La administración del modelo es llevada a cabo creando dimensiones a través de la aplicación de los operadores *TOLAP* presentados en el Capitulo 2 y la definición de los distintos atributos sobre las tablas de dimensión, junto a la creación de las tablas de hechos, las cuales van componer el modelo de datos.

El usuario tiene también la posibilidad de definir vistas materializadas, las cuales pueden llegar a ser utilizadas por el optimizador de consultas en la resolución de una consulta *TOLAP-QL*. El optimizador también puede solicitar la creación de vistas materializadas a los efectos de la resolución más eficiente de las consultas formuladas.

5.1 Diagrama de Arquitectura

El diagrama de la Figura 5.13 presenta la Arquitectura *TOLAP* propuesta, basada en una arquitectura Web, utilizando como protocolo principal SOAP (Web Services) y un Servlet Container[1], por ejemplo Tomcat Server[2], el cual implementa los módulos de TOLAP.

Se desarrollaron dos módulos para la interacción con el usuario: *TOLAP Viewer* y *TOLAP Query*. El primero es una Applet (modelo de aplicación portable desarrollado por la empresa SUN) que se comunica con el server exclusivamente a través de mensajería SOAP. La interfaz visual permite administrar la Metadata *TOLAP*: tablas de *dimensiones*, *tablas de hechos y vistas materializadas*, así como también administrar instancias y atributos. El modulo denominado *TOLAP Query* brinda la posibilidad de realizar consultas *TOLAP-QL* y visualizar los resultados.

Cabe destacar la interoperabilidad de Web Services en cualquier plataforma, motivo por el cual, si el usuario lo deseara, podría desarrollar una aplicación teniendo en cuenta los métodos Web expuestos por el servicio Web de *TOLAP*.

---

[1] Tipo de componente de código JAVA que se ejecuta en un servidor de proceso y que cumple con la especificación de brindad por Sun. La misma se conoce por JSR-154 y puede ser consultada en la siguiente URL: http://java.sun.com/products/servlet/download.html

[2] Implementación open source de un servidor de aplicaciones que cumple con la norma JSR-154 y que permite también ser utilizado como servidor WEB. Para más información ver http://tomcat.apache.org

Por medio de la utilización de JDBC[1] y archivos de configuración personalizados, para cada RDBMS se logró independizar el modelo *TOLAP* del motor utilizado para la implantación.

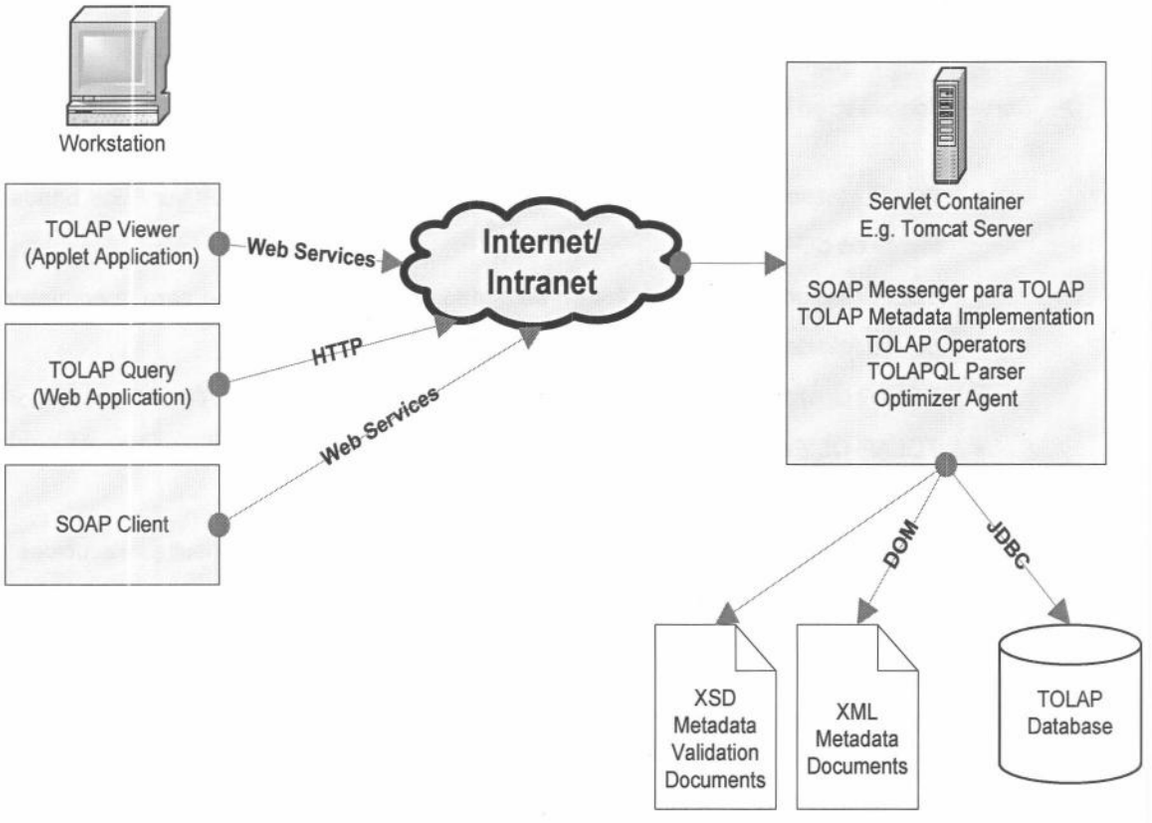


Figura 5.13. Arquitectura de Software

5.2 Componentes de la arquitectura

A continuación se detallan los distintos componentes de la arquitectura, indicado en la Figura 5.13:

- *TOLAP Viewer*: consiste en un applet que brinda una interfaz gráfica para la administración del modelo de datos *TOLAP*.

- *TOLAP Query*: es una aplicación Web que permite al usuario formular consultas *TOLAP-QL* y visualizar sus resultados.

[1] JDBC es el acrónimo de Java DataBase Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede por medio de sentencias SQL.

- *SOAP Client*: dada la interoperabilidad de SOAP entre las distintas plataformas, el usuario puede optar por realizar sus propias implementaciones de *TOLAP Viewer* y *TOLAP Query*, suponiendo que exista una implementación del cliente SOAP para la plataforma en donde correrán las aplicaciones del usuario.

- *Servlet Container*: en este servidor se ejecutan los siguientes componentes:

    - SOAP Messenger para *TOLAP*: se implementa cada uno de los servicios brindados a través de SOAP.
    - *TOLAP* Metadata: implementación de metadata *TOLAP* necesaria para la administración del modelo de datos.
    - *TOLAP Operators*: módulo encargado de ejecutar los operadores *TOLAP* solicitados.
    - *TOLAP-QL Parser*: realiza la evaluación y traducción de una consulta *TOLAP-QL* a SQL estándar.
    - *Optimizer Agent*: encargado de realizar la optimización de las consultas traducidas por el *TOLAP-QL* Parser.

5.2.1 Independencia del RDBMS

Para lograr este objetivo se utilizó un documento XML de configuración definido específicamente para *TOLAP* llamado `parameters.xml`. Se definieron como parámetros del sistema las entradas para la definición del tipo de datos nativo de cada motor utilizado en la representación de los tipos *TOLAP*, así como también las definiciones de funciones especificas para cada motor (por ejemplo: funciones de fecha del sistema, conversiones de tipos de datos, etc.).

A continuación se transcribe una línea del archivo `parameters.xml`, que permite especificar cómo se obtiene la fecha actual en un RDBMS dado.

Por ejemplo en el Caso de SQL-Server 2000, la obtención de la fecha actual para *TOLAP-QL* (brindada por la función *GET_SQL_SYSDATE*) se especifica mediante la utilización de la función *getdate()* de dicho motor, de la siguiente forma :

SQL-Server 2000

```
<parameter name="GET_SQL_SYSDATE" value="SELECT GETDATE() as ActualDate"/>
```

Mientras que para *Oracle* resulta:

Oracle

```
<parameter name="GET_SQL_SYSDATE" value="SELECT sysdate as ActualDate from DUAL"/>
```

Especificación utilizada en pruebas con el motor de base de datos *Teradata*

Teradata

```
<parameter name="GET_SQL_SYSDATE" value="SELECT current_timestamp as ActualDate"/>
```

5.3 Implementación de servicios Web

TOLAP implementa toda su mensajería a través de servicios Web entre un cliente SOAP y el servidor de aplicaciones.

Se definió un servicio denominado *urn:Tolap* en el cual están publicados todos los métodos accesibles a través de SOAP. Para utilizar los servicios, la aplicación debe realizar una conexión SOAP al Endpoint HTTP que implementa los servicios.

Los servicios SOAP se registran a partir de un archivo xml que en el caso de *TOLAP* se denomina `Tolap.xml`, el que posee la definición de los métodos publicados en urn:Tolap y los serializadores/deserializadores[1] de los objetos utilizados en la parametrización de los métodos y la conformación de resultados.

Para *TOLAP* se eligió implementar los servicios de SOAP en el servidor Apache. Los servicios se registran por medio del documento XML haciendo un hit sobre la siguiente servlet:

http://localhost:8080/soap/servlet/rpcrouter

Especificando los servicios en el `Tolap.xml` con el contenido definido en el Anexo 2.

La siguiente sección explica en detalle el funcionamiento de esta arquitectura.

---

1 la serialización (o marshalling en inglés) consiste en un proceso de codificación de un objeto en un medio de almacenamiento (como puede ser un archivo, o un buffer de memoria) con el fin de transmitirlo a través de una conexión en red como una serie de bytes o en un formato humanamente más legible como XML. La serie de bytes o el formato pueden ser usados para crear un nuevo objeto que es idéntico en todo al original, incluido su estado interno (por tanto, el nuevo objeto es un clon del original). La serialización es un mecanismo ampliamente usado para transportar objetos a través de una red, para hacer persistente un objeto en un archivo o base de datos, o para distribuir objetos idénticos a varias aplicaciones o localizaciones. En nuestra implementación SOAP, existen serializadores y deserializadores. Los primeros se encargan de la serialización de un objeto y/o mensaje, residente en el ambiente del emisor, mientras que los segundos se encargan de la reconstrucción en el ambiente del receptor.

5.4 Capas de servicios de Web TOLAP

La estructura de la implementación de SOAP se puede dividir en las siguientes capas:

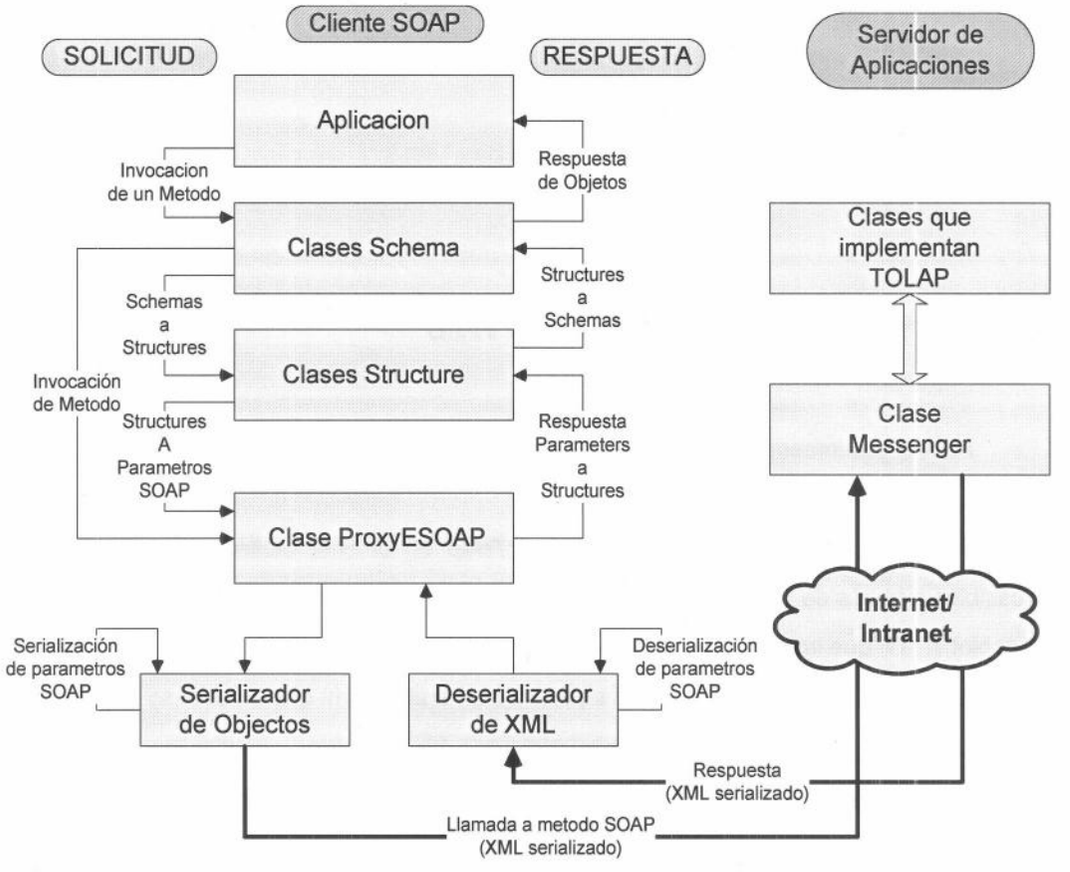


Figura 5.14. Capas de Software de la Arquitectura TOLAP

En este esquema se observan las distintas capas de Objetos Java utilizados en la Arquitectura SOAP. Estas capas tienen la siguiente funcionalidad:

- *Aplicación*: Brinda la interfaz al usuario. Ejemplos de la misma son exhibidos por las Figuras 5.16, 5.17 y 5.18.
- *Clases Schema*: Exponen a la interfaz de usuario las llamadas a objetos *TOLAP*, realizando la abstracción de la arquitectura Web subyacente. Por ejemplo, permite solicitar los niveles existentes para una dimensión a un instante dado.

- *Clases Structure*: Para poder ser enviados objetos a través de SOAP, los mismos deben poder ser serializados/deserializados. Debido a que la implementación SOAP utilizada solo sabe como serializar/deserializar solo ciertos tipos de datos Java y no todos los objetos, es necesario realizar una abstracción que dado un objeto pueda pasarlo del cliente al servidor y viceversa. Estas clases encapsulan dicho proceso realizando una abstracción del mismo y por lo tanto resultan ser las encargadas de conocer como mapear las variables de instancia de los objetos Java al tipo correcto para realizar el binding con XML/SOAP.

- *Clase ProxyESOAP*: Los objetos de esta clase saben como utilizar SOAP para referenciar los servicios brindados por *WEB TOLAP*

- *Serializadores/Deserializadores*: Se encargan de codificar/decodificar los objetos que poseen las estructuras complejas definidas por las clases del Package[1] Structures posibilitando su transmisión desde y hacia el servidor, por parte del cliente SOAP.

- *Clase Messenger*: Son las que se encargan de invocar la aplicación de los Operadores *TOLAP* y también resuelven las demás solicitudes realizadas por los clientes SOAP.

A modo de ejemplo, se detalla la secuencia de ejecución necesaria para obtener los niveles de la dimensión Geografía en el instante '2006-06-30T05:00:11'.

1. La aplicación cliente crea un objeto de la clase `tolap.schemas.Dimension` y realiza la siguiente invocación al método `getLevels` (el cual devuelve los niveles de dicha dimensión en un t dado):

```
String granularity = 'Segundos';
tolap.schemas.Dimension d = new tolap.schemas.Dimension('Geografia');
TimeInstant t = new TimeInstant('2006-06-30T05:00:11', granularity);
Vector Levels = d.getLevels(t);
```

2. La anterior llamada al método `getLevels` desencadena una llamada a un objeto de la clase ProxyESOAP, el cual interactúa con el correspondiente objeto del package Structure para realizar la llamada al Servicio `getLevels` de *WEB TOLAP* a través de SOAP

[1] Conjunto de Clases agrupadas de acuerdo a una funcionalidad especifica.

```
public Vector getLevels(TimeInstant t) throws TolapException {
    try {
       Vector vecLS = ESOAPConnection.proxy.getLevels(this.getName(), t);
        Vector vecL = StructureToSchema.toLevels(vecLS);
         return vecL;
         }
    catch (Exception e) {
         .
         .
         }
      }
```

3. El siguiente método `getLevels` es él correspondiente a la línea resaltada en el punto anterior y en el cual se puede ver como interacciona con el *package Structure* antes de realizar la llamada a SOAP para acceder al servicio de *WEB TOLAP*.

```
public synchronized Vector getLevels(String dimension, TimeInstant t)
                                throws TolapException {
   try {
     // Preparación de la llamada SOAP al servicio getLevels

      Envelope e =  new Envelope();
      Method m = e.setMethod("getLevels", C_SOAP_SERVICE_NAME);

     // seteo de los parametros del servicio a llamar

      m.addString("dimension", dimension);


     // se parametriza el t a su correspondiente TImeInstantStructure,
     //  ya que no se trata de un String, sino de un objeto de
     //           tipo TimeInstant, desconocido para SOAP.

      Parameter tis = t.toParameter();
      m.addAnyType("t", tis);

     // llamada a SOAP para que acceda al servicio

      Envelope r = tESOAP.call(e, null);

     // Resultado obtenido del servicio por la llamada SOAP

      Parameter p = r.getMethod().getParameter(0).getAnyType();
      if (p != null) {
         Vector v = ParameterToStructure.levels(p);

     // Devuelvo los niveles existentes al t solicitado
         return v;
      }
      Else

     // No existen niveles a dicho t
         return null;
   }
   catch (SoapException se) {
      throw buildTolapException(se.getMessage());
   }
}
```

4. La implementación de SOAP implícitamente realiza la serialización a XML de los objetos Java, teniendo en cuenta los serializers registrados. Este proceso es realizado a través del método `marshall()` de cada serializer (se define uno para cada objeto que se transmite).
5. El Servidor de Aplicaciones recibe el mensaje y la clase `tolap.messenger.Messenger` implementa los métodos publicados. En nuestro ejemplo, `getLevels()` es `tolap.messenger.Messenger.getLevels()`. Se resuelve el método y se prepara para devolver el resultado, realizando la serialización correspondiente y enviándola por el canal de transmisión.
6. El objeto de tipo ProxyESOAP recibe la respuesta al mensaje, realizándose implícitamente la deserialización a través del proceso correspondiente.
7. El resultado es convertido de SOAP Parameter a structure, luego a schema y luego es recibido por el método de aplicación que había solicitado esta operación.

Cada clase de la jerarquía `tolap.structures` implementa un método `toParameter()`, necesario para mapear las variables de instancia de Java al tipo correcto para el binding XML/SOAP. Por ejemplo, en la clase `tolap.structure.TimeInstantStructure`, utilizada en el método anterior, se definen dos variables de instancia:

```
public Date date;
public String granularity;
```

Por simplicidad, no se definieron getters/setters[1] para los objetos de tipo Structure, por lo cual todas las variables de instancia están definidas como públicas. El constructor con la totalidad de las variables de instancia como parámetro es el único definido.

A modo de ejemplo, se transcribe el fragmento de código del método `toParameter()` de la clase `tolap.structure.TimeInstantStructure`, donde se nota como se realiza el correspondiente bind de objetos Java a las clases SOAP encargadas de pasar el valor al servicio *WEB TOLAP*.

1 En programación orientada a Objetos, la exposición de los atributos de los mismos se suele hacer por medio de dos tipos de métodos. Los métodos que setean (setters) los valores de los atributos y los que leen (getters) los valores que poseen los atributos.

```
public Parameter toParameter() {
   Parameter p = new Parameter("p");
   // Registración de Serializer/Deserializer
   p.addAttribute("xmlns:ns2", "urn:xml-soap-timeinstantstructure");
   p.addAttribute("xsi:type", "ns2:timeinstantstructure");
   // date
   Parameter pDate = new Parameter("date");
   pDate.addAttribute("xsi:type", "xsd:dateTime");
   try {
      pDate.setDate(this.date);
   }
   catch (Exception e) {
   }
   p.addAnyType("date", pDate);
   // granularity
   Parameter pGranularity = new Parameter("granularity");
   pGranularity.addAttribute("xsi:type", "xsd:string");
   pGranularity.setValueAsString(this.getGranularity());
   p.addAnyType("granularity", pGranularity);
   return p;
}
```

Se puede observar que se registra el serializer/deserializer para esta clase, definido en el urn:Tolap registrado en el application server, que consta en el archivo de Web Services nombrado `Tolap.xml` y del cual se transcribe la parte pertinente.

```
<isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              xmlns:x="urn:xml-soap-timeinstantstructure"
              qname="x:timeinstantstructure"
            javaType="tolap.structures.TimeInstantStructure"
            java2XMLClassName="tolap.soap.TimeInstantStructureSerializer"
            xml2JavaClassName="tolap.soap.TimeInstantStructureSerializer"/>
```

En este caso se convierte la variable de tipo `java.util.Date` llamada date a `xsd:dateTime` de SOAP, y la variable granularity de tipo `String` a `xsd:string`.

Existe una tabla de equivalencias entre los tipos Java y los tipos XML.

Las colecciones de tipo `java.util.Vector` son implementadas por defecto y los tipos complejos son mapeados registrando el serializer/deserializer en el orden y la jerarquía correcta del tipo complejo (en el caso de TimeInstantStructure el serializar/deserializer esta definido en la clase `tolap.soap.TimeInstantStructureSerializer`).

5.5 Documentos XML para especificación de Rollups

Para especificar las funciones de rollup entre las distintas instancias se utilizan documentos XML con estructura a ser validados por medio del archivo `funcion.xsd` XML schema. Dichas instancias pueden ser las iníciales de un nivel inferior al momento de la creación de una Dimensión, o los rollups entre las instancias componentes de los distintos operadores que lo requieran.

La estructura del documento `funcion.xsd` puede verse en el Anexo 1.

A modo de ejemplo se transcribe el archivo XML aplicado en el modelo descripto en la Figura 2.2 al realizar el *Generalize* de *ciudad* a *provincia*, en donde se pueden apreciar las rollups aplicadas a las instancias *Toronto, San Juan* y *San Rafael.*

```
<?xml version="1.0" encoding="UTF-8"?>
<funcion xmlns="http://www.example.com/Funcion"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.example.com/Funcion  funcion.xsd">
   <instance levelFrom="ciudad"
             valueFrom="Toronto" levelTo="provincia" valueTo="Ontario"/>
   <instance levelFrom=" ciudad"
             valueFrom="San Juan" levelTo="provincia" valueTo=" San Juan"/>
   <instance levelFrom=" ciudad"
             valueFrom="San Rafael" levelTo="provincia" valueTo="Mendoza"/>
</funcion>
```

5.6 Aplicación de Operadores TOLAP

Los distintos operadores de actualización de dimensión *TOLAP,* pueden aplicarse también a través de Servicios Web. El *TOLAP* Messenger publica un método por cada uno de los operadores temporales.

Ciertos operadores, como por ejemplo *Generalize*, requieren especificar los rollups de las instancias a las que van a aplicarse cuando se ejecute el operador. Para cada operador que lo requiera, se puede definir un documento XML como el descrito en 5.5 a ser validado por el XML schema llamado funcion.xsd. Una vez que el documento XML, que respeta el formato es construido, se puede enviar como parámetro del método SOAP correspondiente. Este archivo es comprimido por medio de las librerías Java ZIP y es enviado en el mensaje SOAP como parámetro de tipo binary. Para armar el zip, se obtiene un idfile a través del método SOAP `getIdFile()`. De esta manera se centraliza en el servidor la generación de números de identificación (idfile) para garantizar su unicidad. El método es invocado antes de realizar el llamado al método SOAP que se pidió originalmente por medio de la aplicación. Con ese idfile, un número entero, se procede a renombrar el archivo XML (indicado por el usuario) como `<idfile>.xml` y ese archivo se comprime en un archivo de nombre `<idfile>.zip`, por ejemplo,

`884.zip` conteniendo como único archivo a `884.xml`. Si resultare algún error en la aplicación del operador, se procederá a renombrar el archivo `<idfile>.xml` como el nombre original. El archivo .zip es siempre eliminado.

En el caso que ocurra alguna excepción en el servidor al aplicar el operador TOLAP, esta excepción es arrojada (ya serializada) automáticamente por el Apache SOAP y recibida por el cliente SOAP para su tratamiento, en este caso construyendo con el mensaje recibido una `tolap.client.Exception`, el tipo de excepción que entiende la aplicación.

La aplicación *TOLAP* Browser basada en Applets[1] está implementada utilizando ESOAP[2], un cliente liviano SOAP que por su biblioteca pequeña en cantidad de bytes resulta adecuado para Applets.

El diagrama de secuencia de la Figura 5.15, muestra la interacción de las distintas clases de la arquitectura *Web TOLAP* ante la aplicación de un *Generalize* (este diagrama resulta extendible a la invocación de cualquier otro Operador *TOLAP*), en el mismo se puede ver como la invocación del método SOAP tGeneralize (a través de la clase `tolap.schemas.Dimension`.) inicia la secuencia de aplicación.

Nótese que no hace falta realizar la llamada a este método desde el Applet que hemos desarrollado, bastaría con tener un cliente SOAP que invoque directamente el urn:Tolap, y método tGeneralize() con la adecuada parametrización.

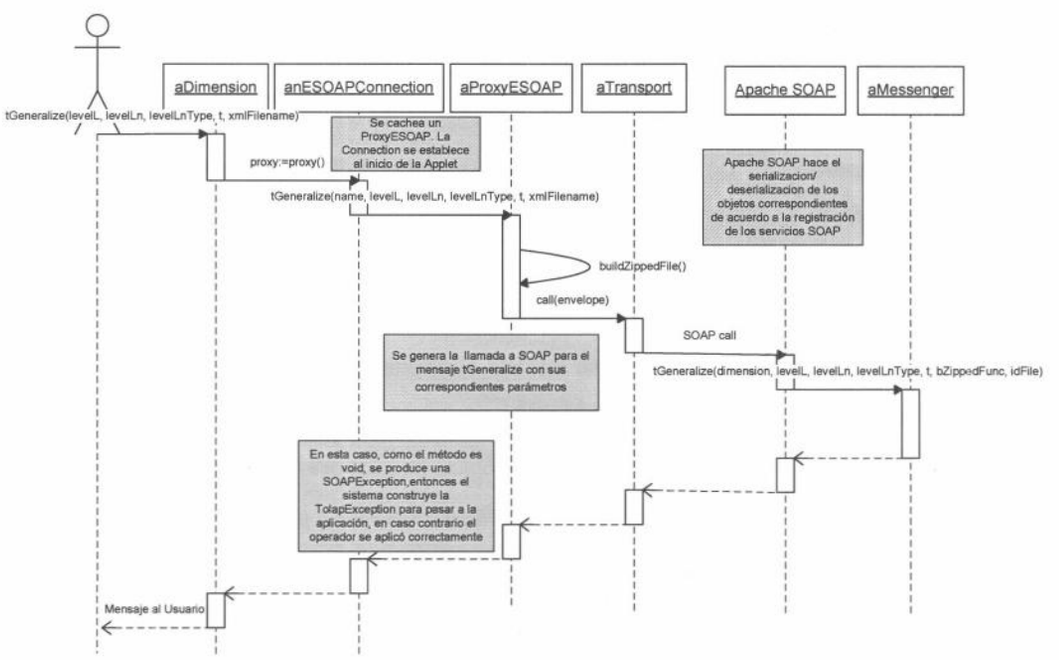


Figura 5.15. Secuencia de interacción del tGeneralize

[1] Para más información se puede referir al siguiente link: http://esoap.ultimodule.com

[2] Implementación open source de SOAP, para más información consultar la siguiente URL: http://www.embedding.net/eSOAP.

5.7 TOLAP Viewer

TOLAP Viewer es una aplicación de Administración de *TOLAP* para la edición de *dimensiones*, *tablas de hechos*, *Atributos*, *Operadores*, *Vistas Materializadas*, etc. Está implementado utilizando Applets, por lo cual es necesario instalar el plug-in de Applets de Sun Microsystems de la plataforma sobre la cual se lo quiere ejecutar. Igualmente, ante el hit inicial sobre la URL del *TOLAP Viewer*, se hace un chequeo sobre la JVM[1], y si no es la correcta, se redirecciona al link Internet de Sun para comenzar el download del plug-in requerido.

A continuación se exhiben algunas pantallas del aplicativo con ejemplos de las distintas operaciones que se pueden realizar a través del *TOLAP Viewer*.

Edición de dimensiones

Esta pantalla constituye la pantalla inicial del aplicativo. La misma se encuentra organizada en tres secciones a saber:

- *La ventana izquierda*

  Exhibe la jerarquía de la dimensión actualmente seleccionada, en este caso la de *Product*, en la misma se puede visualizar a simple vista que *itemId* es el nivel inferior actual.
- *La ventana derecha*

  Exhibe una estructura de árbol conteniendo las instancias correspondientes a cada nivel de la dimensión. Para facilitar la identificación de las mismas, se las ha dotado del mismo color del nivel al que pertenecen.
- *El extremo superior*

  Contiene el menú de selección de navegación, el cual nos permitirá cambiar la dimensión que se está consultando, el correspondiente snapshot (a través de este podremos ver todos los cambios temporales que ha sufrido la dimensión que se está consultando), el instance snapshot (que nos permite ver como se han ido aplicando las distintas rollups a las instancias), buscar una instancia, crear nuevas dimensiones o acceder a las ventanas de manejo de tablas de hechos y vistas materializadas.

[1] Maquina Virtual Java

Sobre ambas ventanas existen menús contextuales que se despliegan al presionar el botón derecho del mouse sobre un objeto de la interfaz (Por ejemplo, si se presiona sobre un nivel de la ventana izquierda, se desplegara un menú que le permitirá aplicar los operadores disponibles para ese nivel, mientras que si lo hace sobre una instancia de la ventana derecha solo se desplegaran los inherentes a las instancias). La Figura 5.16 muestra la ventana en cuestión.

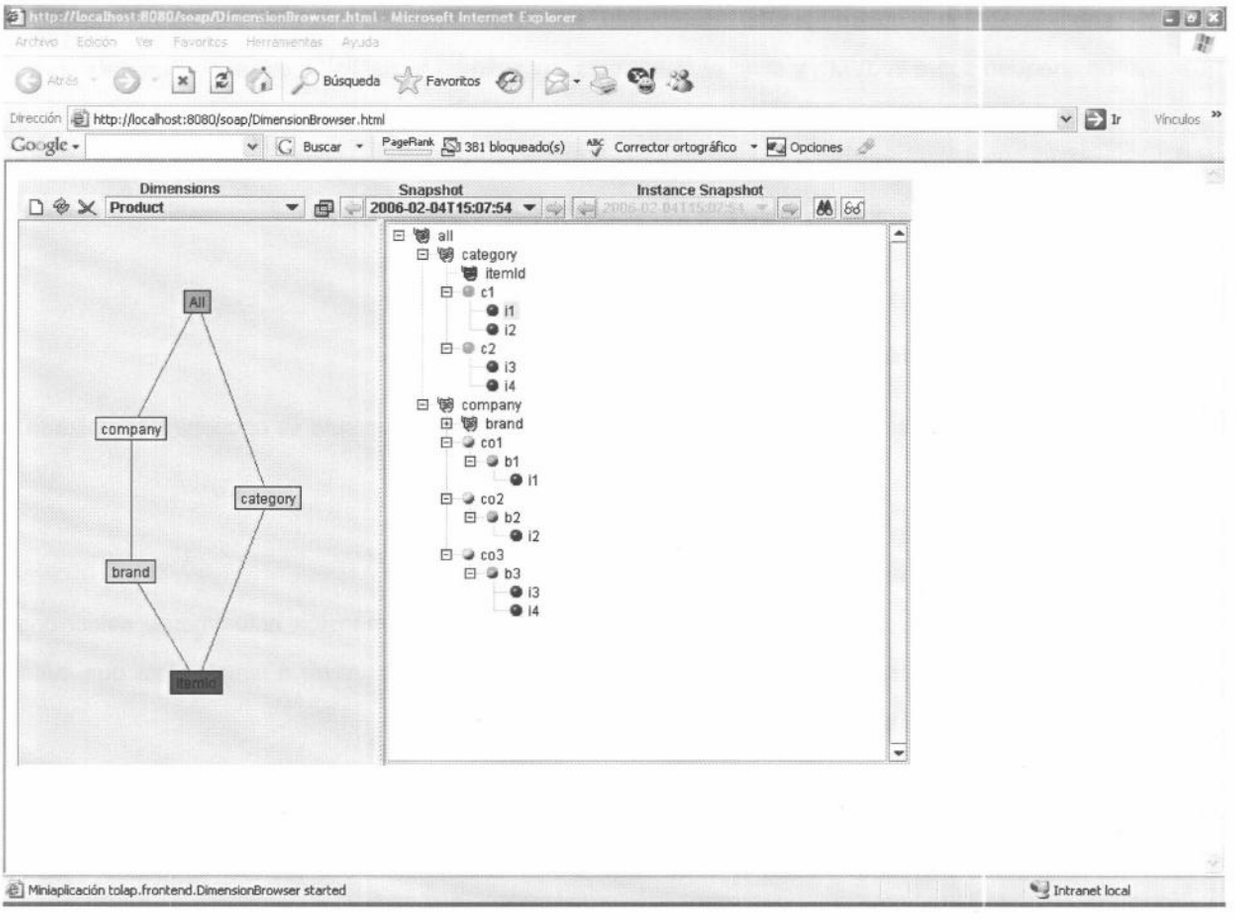


Figura 5.16. Edición de dimensiones

En la Figura 5.16, en la ventana izquierda se ve que el nivel infimo (*itemId*) de la dimensión *Product*, hace rollup hacia los niveles *Brand* y *Category.* En la ventana derecha, se ve que existen cuatro miembros para el nivel *itemId*: i1, i2, i3 e i4. El elemento i1, por ejemplo, hace rollup al elemento b1 en el nivel *brand*.

Edición de tablas de hechos

Esta ventana permite ver las tablas de hechos disponibles, posibilitando tanto la modificación de las existentes como la creación de nuevas.

La Figura 5.17 muestra la interfaz en cuestión, en donde en la solapa denominada *Meta Fact Tables* se visualizan dos tablas de hechos: *Sales* y *Services*, nótese que *Sales* está dividida a su vez en tres versiones las cuales corresponden a los cambios del nivel inferior acontecidos en la dimensión *Geography* y los cuales son descriptos a continuación.

En el momento de creación de Sales (2006-02-18T15:07:03), los niveles inferiores de las dimensiones que la componen eran *city* para *Geography*, *itemId* para *Product* y *storeId* para *Store* y como medida se tomo el campo *amount*. En el instante 2006-02-18T15:17:07 se aplica un Specialize a *Geography* de *city* a *town*, lo cual produce el cierre de la versión 1 de *Sales* al 2006-02-18T15:17:06 y la creación de una versión 2 al instante en que se aplico el *Specialize*. Posteriormente se aplica un nuevo *Specialize* pero ahora de *town* a *square* en el instante 2006-02-18T15:17:21 determinando la creación de la versión 3 de *Sales* a dicho instante y el cierre de la versión 2 al instante previo 2006-02-18T15:17:20 (en nuestro ejemplo, el instante previo es un segundo antes debido a que estamos trabajando con granularidad de segundos, pero en un ejemplo en el cual la granularidad fuese mensual el instante previo seria el mes anterior).

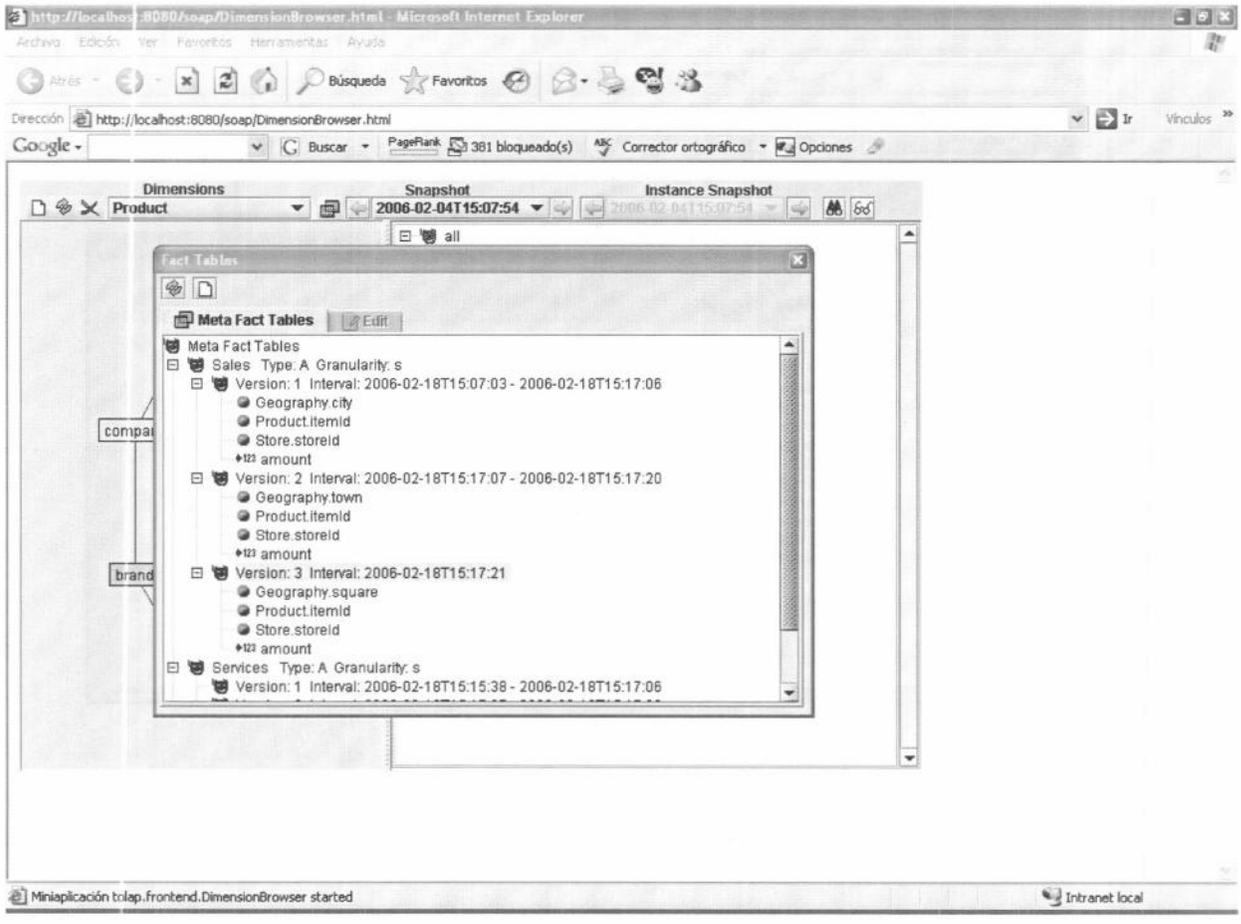


Figura 5.17. Edición de Tablas de Hechos

Edición de Vistas Materializadas

Esta interfaz tiene como finalidad permitir al Usuario gestionar las vistas materializadas disponibles para la resolución de las consultas *TOLAP-QL*. En la Figura 5.18 se muestra la vista materializada *VMTest(región, brand, amount)*. En la pantalla puede verse información estadística de la misma como ser:

- Datos referentes a la *Tabla de hechos* que esta agregando, en este caso *Sales*
- Niveles que la componen, *región* y *brand* en nuestro ejemplo.
- Si ya se la ha materializado
- La cantidad de veces que se la ha referenciado
- La cantidad de registros que posee.

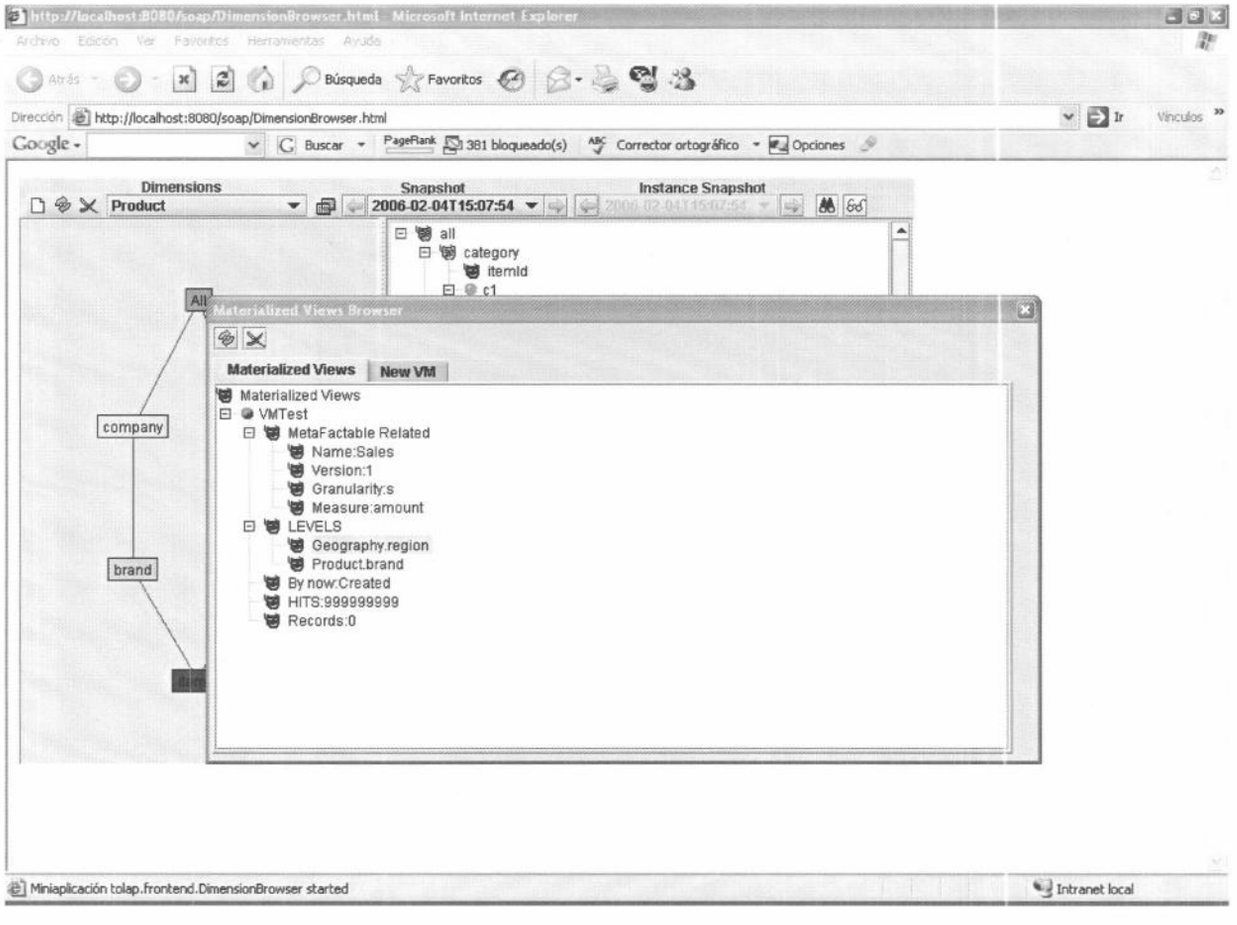


Figura 5.18. Edición de vistas materializadas

5.8 Representación del resultado mediante documentos XML o HTML

Se dispone de la opción de obtener el resultado de una consulta *TOLAP-QL* con formato XML o HTML.

Una consulta *TOLAP-QL* puede ejecutarse a partir de la interfaz Web provista o por medio de un llamado al método SOAP que recibe como parámetro el texto de la consulta y devuelve según la opción elegida, un archivo XML, el cual deberá ser tratado por el cliente o un archivo HTML con la imagen de la consulta previamente transformada a través de XSLT. La plantilla xsl puede ser modificada de acuerdo a las necesidades.

Total de ventas por marca para la región r1 hasta 2006/02/01

```
SELECT P.brand,SUM(amount)
  FROM Sales F, Product P, Geography G
 WHERE F.Product = P.bottom
   AND F.Geography=G.bottom
   AND RUP(G,region:'r1',F.t)
   AND RUP(P,brand,F.t)
   AND F.t < '2006/02/01';
```

Ejemplo de salida XML

Este documento XML lo genera *TOLAP* en el servidor para ser entregado al cliente que lo solicite, ya sea el browser o un programa SOAP:

```
<?xml version="1.0" encoding="ISO-8859-1"?><resulset>   <fila>
<columna>b1</columna>      <columna>700.00</columna>   </fila>   <fila>
<columna>b2</columna>      <columna>400.00</columna>
   </fila>        <fila>          <columna>b3</columna>
<columna>111640.00</columna>        </fila>     </resulset>
```

Con una simple plantilla XSL, que transforma el XML a HTML, como la mostrada a continuación:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                                    version="1.0">
   <xsl:template match="/">
      <table width="128" border="0" cellspacing="0" cellpadding="0">
         <xsl:for-each select="resulset/fila">
            <tr>
               <xsl:for-each select="./columna">
                  <td><xsl:value-of select="."/></td>
               </xsl:for-each>
            </tr>
         </xsl:for-each>
      </table>
   </xsl:template>
</xsl:stylesheet>
```

El documento XML, expresado más arriba, utilizando la plantilla XSL para la transformación, resulta en el siguiente código HTML:

```
<table cellpadding="0" cellspacing="0" border="0" width="128">
<tr>
<td>c1</td>
<td>700.00</td>
</tr>
<tr>
<td>c2</td>
<td>400.00</td>
</tr>
<tr>
<td>c3</td>
<td>111640.00</td>
</tr>
</table>
```

La transformación se realiza servidor, por lo cual el cliente obtiene el resultado directamente en HTML, no hace falta utilizar plantillas en el cliente.

5.9 Resumen

Hemos visto que la arquitectura de software de esta implementación es de tres capas, compuestas por un servidor de aplicaciones, encargado de brindar los servicios Web, un motor de base de datos relacional para soportar el warehouse y un cliente SOAP. Mostramos la independencia que tiene la implementación de uso del motor de base de datos y como se consumían los servicios Web que permiten entre otras cosas invocar a los operadores TOLAP. También mostramos la interfaz grafica del cliente SOAP y sus facilidades. Por último se mostró como el sistema devuelve el resultado de las consultas efectuadas.

Capítulo 6.

PROCESAMIENTO DE CONSULTAS TOLAP-QL

En el capítulo 3 se presentaron la estructura y gramática del lenguaje *TOLAP-QL*. En este capítulo mostraremos como se realiza la traducción de una consulta *TOLAP-QL* a una consulta SQL, que puede ser ejecutada en un motor relacional. En principio, cualquier motor de base de datos que soporte el estándar JDBC 2.0 o superior puede utilizarse.

El proceso de traducción consiste en los siguientes pasos:

- *Análisis Sintáctico*: parsing de la consulta y armado del Árbol Sintáctico.
- *Análisis Semántico*: armado de Tabla de Símbolos, Chequeo de Tipos
- *Generación de SQL*: construcción de la sentencia SQL correspondiente, lista para ser ejecutada.

6.1 Análisis Sintáctico

A partir de la gramática definida en BNF (forma Backus-Naur), incluída en el Anexo 3, se utilizó Java CUP[1], para implementar un Parser LALR(1)[2] de dicha gramática.

En Java CUP, se define un archivo con extensión .cup en el que se vuelca toda la gramática y se escriben las reglas correspondientes. Luego, a partir de ese archivo, se genera el Parser por medio de un utilitario, resultando un conjunto de clases Java que se integraron al código fuente de *TOLAP*.

El árbol sintáctico se genera en tiempo de ejecución por Java CUP dejando una referencia en memoria a un árbol de objetos Java que representa el árbol sintáctico. Dicho árbol es utilizado por la implementación de *TOLAP-QL* para realizar el correspondiente análisis semántico y la posterior traducción a SQL. Cabe destacar que el parser LALR(1) definido representa la sintaxis de *TOLAP-QL*.

[1] Java CUP es la abreviación para Constructor of Useful Parsers for Java. Es un generador de parser para gramáticas de tipo LALR para utilizar con Java. Para más información visite la siguiente URL: http://www.cs.princeton.edu/~appel/modern/java/CUP.
[2] Tipo de gramatica que resulta optima para escribir lenguajes de programación.

Java CUP en tiempo de ejecución evalúa la sintaxis de la consulta *TOLAP-QL* formulada por el usuario, y si existiese algún problema de sintaxis, se devolvería un código de error por parte de Java CUP indicando que no existe una regla *shift* o *reduce* que pueda ser aplicada en el estado del parsing hasta donde se hubiere llegado aplicando las reglas exitosamente.

6.1.1 Representación del Árbol Sintáctico

El árbol sintáctico resultante del parsing de Java CUP es un conjunto de instancias de objetos relacionadas en la jerarquía de clases definida por la clase `tolap.parser.SintacticTree`.

Tomemos como ejemplo la consulta:

Cantidad total de productos vendidos por categoría para la región 'r1'

La cual hemos estudiado previamente y que escribe en TOLAP-QL como:

```
SELECT P.categoria, SUM(cantidad)
  FROM Ventas F, Producto P, Geografia G
 WHERE F.Producto = P.bottom
   AND F.Geografia = G.bottom
   AND RUP(G,region:'r1',F.t)
   AND RUP(P,categoria,F.t);
```

Dicha consulta genera el siguiente árbol sintáctico, en el cual podemos ver que a la izquierda de cada línea figura el nombre de la clase que implementa el elemento del lenguaje *TOLAP*. Por ejemplo, cada consulta en sí es un programa *TOLAP-QL* (programa que puede contener una o más sentencias) representado por la clase Java `tolap.parser.TolapProgram`. A derecha aparece una keyword que identifica cada línea.

Arbol Sintáctico.

```
tolap.parser.TolapProgram
    tolap.parser.Select SELECT
        tolap.parser.SelectHead
            tolap.parser.Atom ATOM
                tolap.parser.IdentifierHead HEAD_DIM_LEVEL
                    tolap.parser.Id P
                    tolap.parser.Id categoria
            tolap.parser.Aggregate SUM
                tolap.parser.IdentifierLevel ID_LEVEL
                    tolap.parser.Id cantidad
        tolap.parser.SelectBody SELECTBODY
            tolap.parser.List
            tolap.parser.From FROM_ALIAS
                tolap.parser.Id Ventas
                tolap.parser.Id F
            tolap.parser.From FROM_ALIAS
                tolap.parser.Id Producto
                tolap.parser.Id P
            tolap.parser.From FROM_ALIAS
                tolap.parser.Id Geografia
                tolap.parser.Id G
            tolap.parser.Where WHERE
                tolap.parser.And ANDEXP_AND_RUPEXP
                    tolap.parser.And ANDEXP_AND_RUPEXP
                        tolap.parser.And ANDEXP_AND_EQEXP
                            tolap.parser.Condition EQEXP
                                tolap.parser.EqualityExpression REL_EQ_REL
                                    tolap.parser.IdentifierLevel ID_DIM_LEVEL
                                        tolap.parser.Id F
                                        tolap.parser.Id Producto
                                    tolap.parser.IdentifierLevel ID_DIM_LEVEL
                                        tolap.parser.Id P
                                        tolap.parser.Id bottom
                            tolap.parser.EqualityExpression REL_EQ_REL
                                tolap.parser.IdentifierLevel ID_DIM_LEVEL
                                    tolap.parser.Id F
                                    tolap.parser.Id Geografia
                                tolap.parser.IdentifierLevel ID_DIM_LEVEL
                                    tolap.parser.Id G
                                    tolap.parser.Id bottom
                        tolap.parser.RupExpression RUP
                            tolap.parser.RupNoVarIdentifier RUP_ID_DIM
                                tolap.parser.IdentifierDimension DIMENSION
                                    tolap.parser.Id G
                            tolap.parser.RupNoVarIdentifier RUP_ID_VALUE
                                tolap.parser.IdentifierLevel ID_LEVEL
                                    tolap.parser.Id region
                                tolap.parser.ValueString r1
                            tolap.parser.IdentifierLevel ID_DIM_LEVEL
                                tolap.parser.Id F
                                tolap.parser.Id t
                    tolap.parser.RupExpression RUP
                        tolap.parser.RupNoVarIdentifier RUP_ID_DIM
                            tolap.parser.IdentifierDimension DIMENSION
                                tolap.parser.Id P
                        tolap.parser.RupNoVarIdentifier RUP_ID
                            tolap.parser.IdentifierLevel ID_LEVEL
                                tolap.parser.Id categoria
                        tolap.parser.IdentifierLevel ID_DIM_LEVEL
                            tolap.parser.Id F
                            tolap.parser.Id t
```

Veamos el siguiente extracto del árbol sintáctico anterior

```
...
        tolap.parser.SelectHead
            tolap.parser.Atom ATOM
                tolap.parser.IdentifierHead HEAD_DIM_LEVEL
                    tolap.parser.Id P
                    tolap.parser.Id categoria
            tolap.parser.Aggregate SUM
                tolap.parser.IdentifierLevel ID_LEVEL
                    tolap.parser.Id cantidad
...
```

El objeto `tolap.parser.SelectHead` representa las columnas especificadas en la cláusula `SELECT`: `P.categoria` y `SUM(cantidad)`. Nótese la diferencia entre cada átomo: uno es un objeto de tipo `tolap.parser.Atom` con un `tolap.parser.IdentifierHead` como hijo, y éste último con dos nodos hijo de tipo `tolap.parser.Id` que representan `P.categoria`; y el otro nodo hijo de `tolap.parser.SelectHead` es un objeto de tipo `tolap.parser.Aggregate SUM` sobre `cantidad`.

`HEAD_DIM_LEVEL` indica que los elementos que cuelgan de este nodo del árbol (en la impresión se han indentado a derecha) contienen la información de un nivel de una dimensión, mientras que para el caso de **`ID_LEVEL`** los hijos contienen la información un nivel.

A partir de este árbol sintáctico, se detalla a continuación cómo se realiza el análisis semántico basado en síntesis de atributos a partir del árbol de objetos entregado como resultado del parsing de Java CUP.

6.2 Análisis Semántico

6.2.1 Síntesis de Atributos del parser

A partir de la síntesis de atributos sobre el árbol sintáctico se realizan:

- El armado de la Tabla de Símbolos
- El Chequeo de Tipos

6.2.2 Tabla de Símbolos

La tabla de símbolos se construye a partir del Árbol Sintáctico. Posee distintos tipos de Símbolos, que se resumen en los siguientes:

Tipo de Símbolo	Descripción
Palabra Reservada	Son las keywords de TOLAP-QL
Select	Símbolo representando una sentencia SELECT, utilizado para agregar los símbolos de su jerarquía
Tabla	Existen tres tipos de tablas: Dimensión, Tabla de hecho y Store
Alias	Son los alias que se realizan sobre elementos componentes de las consultas
Columna	Son los alias de cada columna que se exporta de una consulta con cláusula STORE
VarLevel	Alias de variable nivel X perteneciente a una cláusula RUP(Dimensión, VAR X : var x, t)
VarInstance	Alias de variable instance x perteneciente a una cláusula RUP(Dimensión, VAR X : var x, t)

El parser soporta scopes (alcances) anidados, de los siguientes tipos:

Tipo de Scope	Descripción
Global	Contiene todas las palabras reservadas de TOLAP QL, por ej.: SELECT, FROM, etc.
Sentencia	Cada sentencia SELECT tiene su scope, los cuales poseen a su vez scopes anidados.
WHERE	Contiene los alias de instancias, VAR levels y VAR instances.
Bloque	Los bloques y los bloques NOT poseen su propio scope y permiten declarar por ej. alias locales de instancias que pertenecen solamente al bloque y no son visibles afuera.

6.2.3 Generación de Símbolos

La Generación de la Tabla de Símbolos involucra la ejecución del método polimórfico `generarSimbolos()` de la jerarquía `SintacticTree`. Éste método se divide básicamente en dos partes, en el siguiente orden:

- Generación de símbolos del FROM
- Generación de símbolos del WHERE

En el caso que la sentencia contenga la cláusula STORE, antes de invocar a la generación de símbolos de la cláusula FROM y WHERE, se debe invocar la generación de símbolos para la cláusula STORE. Los símbolos correspondientes a la cláusula STORE (las columnas con alias en el SELECT),

necesitan ser agregados a la Tabla de Símbolos para que puedan ser referenciados por las consultas siguientes pertenecientes al programa *TOLAP*.

Para la generación de símbolos de la cláusula WHERE se ejecutan los siguientes métodos polimórficos:

- `generarSimbolos()`: agrega los símbolos pertenecientes al scope del WHERE. Ellos son:
  - Alias de variables de instancia realizados en *RUPs*

```
RUP(S,tiendaId:st,F.t)
```

En este caso la variable `st` es agregada al scope del WHERE

  - Scopes anidados: las estructuras de Blocks agregan scopes anidados

```
SELECT T.semana, SUM(cantidad)
  FROM Tiempo T, Servicios F, Doctor D
 WHERE F.Tiempo = T.bottom AND
       F.Doctor = D.bottom AND
       RUP(D,doctorId:d,F.t) AND d.nombre='Martinez' AND
       NOT (RUP(D,doctorId:d1,F.t,d) AND d1.nombre='Feinsilver');
```

d1 no es visible a nivel del WHERE, solamente en el alcance del Block NOT mientras que d es visible dentro de dicho alcance.

- `getVariables()`: utilizado por `generarSimbolos()`, obtiene los niveles o instancias variables con el keyword VAR de las sentencia *RUP*.

Existen dos tipos de formulación de VARs en las sentencias SELECT:

`RUP(C.customerId, VAR X, t)`	Variable el nivel, todas las rollups estructurales desde el indicado en RUP from, en este caso C.customerId.
`RUP(C.customerId, VAR X: VAR x, t)`	Variables los niveles y sus instancias correspondientes.

El método polimórfico `getVariables()` recorre toda la jerarquía del `SintacticTree` y está implementado en la clase `RUPVarIdentifier`.

Se realiza un parsing para determinar la estructura de la dimensión mencionada en la expresión *RUP*. De esta manera se obtienen las VAR correspondientes que son asignadas al `ParserProfile` para la correcta traducción.

Siguiendo con el ejemplo de la consulta anterior, se generó la siguiente tabla de símbolos:

```
Scope Global
    Palabra reservada SELECT
    Palabra reservada FROM
    Palabra reservada WHERE
    Palabra reservada COUNT
    Palabra reservada SUM
    Palabra reservada AVG
    Palabra reservada MAX
    Palabra reservada MIN
    Palabra reservada NOT
    Palabra reservada AND
    Palabra reservada OR
    Palabra reservada AS
    Palabra reservada VAR
    Palabra reservada RUP
    Palabra reservada STORE
    Scope Select
        Alias F Ventas Fact
        Alias P Producto Dimension
        Alias G Geografia Dimension
        Scope Where
```

Nótese que el árbol que representa la tabla de símbolos es diferente del árbol que representa el árbol sintáctico de la consulta. En este caso, se pueden observar los distintos scopes anidados comenzando por el `Scope Global`, que contiene las palabras reservadas del lenguaje *TOLAP-QL*, el `Scope Select` que representa el scope de la consulta y el `Scope Where` que representa los símbolos definidos a nivel del WHERE.

El `Scope Select` contiene los alias definidos en las columnas del SELECT y los definidos para las tablas de dimensión y Tablas de hechos del FROM. En este caso, F como alias de la *tabla de hechos Ventas*, y P como alias de la *dimensión Producto* y G como alias de la *dimensión Geografía* respectivamente.

6.2.4 Aspectos de Implementación

Clase ParserProfile

La necesidad de definir una clase especial para administrar perfiles en el Parser se debe a que por cada SELECT perteneciente a un programa *TOLAP*, existen datos particulares que son obtenidos a lo largo de la sesión de traducción y deben ser almacenados para optimizar el acceso a todos valores vitales para la traducción, como por ejemplo: la tabla de hechos utilizada, flags sobre optimizaciones que pueden ser realizadas sobre la consulta, niveles de la vista materializada a ser utilizados en la reescritura de la consulta, etc.

Se crea un perfil por cada sentencia SELECT, y una colección de objetos `ParserProfile` es mantenida a lo largo de una sesión de traducción de *TOLAP-QL* a SQL.

El creador de cada objeto `ParserProfile` es el método `generarSimbolos()` especializado en la clase `Select`.

Luego, durante la traducción se consultarán los perfiles para obtener información de entorno correspondiente.

Decoración de Nodos con Información

Se diseñó la jerarquía `InfoNode` con el propósito de decorar con información los nodos de la consulta que representan niveles, atributos y variables.

A cada nodo en el árbol sintáctico que representa alguno de los ítems nombrados anteriormente, se agrega un objeto de la jerarquía `InfoNode` de acuerdo a su tipo de nodo:

`InfoNodeAliasInstance`	Alias sobre una instancia de una sentencia RUP
`InfoNodeBoolean`	Un node BOOLEAN como columna del SELECT
`InfoNodeBottom`	Un ínfimo de una Dimensión, luego se reemplazará en tiempo de ejecución por la versión correspondiente
`InfoNodeDimLevel`	Un nivel de una Dimensión
`InfoNodeDimLevelAttr`	Un Atributo de una Dimensión
`InfoNodeFactTableColumn`	Una columna de una tabla de hechos
`InfoNodeNow`	Una nodo de tipo NOW
`InfoNodeStoreColumn`	Un STORE column
`InfoNodeTable`	Una Tabla del FROM de tipo: dimensión, tabla de hechos, Store
`InfoNodeVarInstance`	Una instancia VAR de una cláusula RUP
`InfoNodeVarLevel`	Un nivel VAR de una cláusula RUP

Los objetos `InfoNode` se crean en el método polimórfico `generarSimbolos()`, teniendo en cuenta los lookups realizados a la Tabla de Símbolos.

6.3 Generación de expresiones SQL

El programa *TOLAP-QL* se traduce generalmente a un conjunto de expresiones SQL.

Para explicar el proceso de traducción detallamos el método *generarSQL* definido en la clase `tolap.parser.Select`. Este método se encarga de evaluar si la consulta es un query o metaquery y procede con la traducción.

Un programa *TOLAP* es un conjunto de sentencias. En particular en esta implementación de *TOLAP-QL* son sentencias de tipo SELECT, representadas por objetos de tipo `tolap.parser.Select`. Un programa *TOLAP* entonces es una colección de objetos `tolap.parser.Select`, y si se tiene una sola sentencia SELECT, es un programa *TOLAP* con sólo dicha sentencia, es decir, una colección de un solo elemento.

El siguiente pseudocódigo explica el proceso de traducción para una sentencia SELECT.

Método generarSQL

```
inicio
      si es un metaquery
            para cada variable VAR definida
                  obtener rollups correspondientes
            fin para
            traducir a SQL
      else
            si se especificó una tabla de hechos
                  para cada version de la tabla de hechos
                        si la versión está entre los intervalos de t
                                          definidos por el usuario
                              calcular ínfimos
                              marcar dimensiones no utilizadas del FROM
                              si existe una Vista Materializada aplicable
                                                            a la consulta
                                    eliminar joins de la consulta original
                                                            (si es necesario)
                              fin si
                              traducir a SQL (para esta versión)
                              end if
                  fin para
            si no
                  traducir a SQL
            fin si
      fin si
      agregar GROUP BY y UNIONS
      si es un STORE
            crear VIEW STORE
      fin si
fin
```

Traducción de expresiones RUP

En la traducción a SQL de expresiones *RUP* se toman en cuenta los tres argumentos incluidos en la definición de la misma:

```
RUP(Dim.NivelInf: [NivelInf.value | NivelInf.variable],
    Dim.NivelSup: [NivelSup.value | NivelSup.variable],
    time, [instanceAlias])
```

Si `Dim.NivelInf` es el ínfimo solo se menciona el nombre de la Dimensión.

A continuación se describe el caso general y se citan algunos de los casos y combinaciones más usuales en la utilización y formulación de expresiones *RUP* en consultas *TOLAP-QL*:

Caso general) `RUP(DimensionInf.nivelInf,DimensionSup.nivelSup,F.t)`

La semántica de este caso es: RUP devuelve *verdadero* si existe una función del rollup desde el nivel `nivelInf` de la dimensión `DimensionInf` al nivel `nivelSup` de la dimensión `DimensionSup` en el instante F.t.

La siguiente consulta: *Listar las localidades actuales de cada provincia* que en TOLAP-QL escribimos:

```
SELECT localidadId, provinceId
 FROM Geografia G
 WHERE RUP(G.localidadId, G.provinceId, Now);
```

Se traduce a SQL como:

```
SELECT DISTINCT G.localidadId, G.provinceId
 FROM DIMENSION_Geografia G
 WHERE convert(datetime, '2006-10-07 15:26:31', 121) BETWEEN G.instant_from AND
      CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to END
```

Como se puede apreciar, la traducción solo controla que al momento de ejecución de la consulta exista una rollup del nivel *localidadId* al nivel *provinceId*.

Caso 1) `RUP(Dimension,nivel1,F.t)`

En este caso la expresión RUP devuelve *verdadero* si existe un nivel `nivel1` en el instante F.t.

Aquí, F.t debe estar contenido entre el intervalo del evento de la dimensión representado por los atributos *instantFrom* e *instantTo*.

La traducción a SQL genera para las expresiones *RUP* una cláusula BETWEEN, teniendo en los tres parámetros de la expresión *RUP*. El nombre de columna default para la columna t en las tablas que representan las versiones de las tablas de hechos es *instant*. Debido a que en la implementación se ha

tomado como NULL el valor de *instantTo* para las instancias abiertas (a NOW) al encontrarse estos casos se reemplaza el valor NULL por la función del RDBMS que devuelva el instante actual.

Se puede resumir la traducción en la siguiente cláusula:

```
F.instant BETWEEN D.instant_from
              AND CASE WHEN D.instant_to IS NULL THEN GETDATE()
                                                 ELSE A.instant_to END
```

Se brinda un ejemplo de este tipo de *RUP* y su traducción para la consulta: *Asistencia brindadas por línea de financiación.* (Correspondiente al modelo de datos expuesto en la Figura 8.19)

Consulta TOLAP-QL

```
SELECT A.Asistencia, SUM(amount)
  FROM Prestamos P, Asistencias A
 WHERE P.Asistencias = A.bottom
   AND RUP(A,Asistencia,P.t);
```

Traducción a SQL

```
SELECT col1, SUM(col2) AS col2
  FROM (SELECT Asistencias_Asistencia col1, SUM(P.amount) col2
          FROM Prestamos_2 P
        GROUP BY Asistencias_Asistencia
        UNION
        SELECT A.Asistencia col1, SUM(P.amount) col2
          FROM Prestamos_3 P, DIMENSION_Asistencias A
         WHERE P.Asistencias_Asistencia = A.Asistencia
           AND P.instant BETWEEN A.instant_from AND CASE
                     WHEN A.instant_to IS NULL THEN GETDATE()
                                    ELSE A.instant_to END
         AND A.Asistencia IS NOT NULL
         GROUP BY A.Asistencia)
UNIONS
GROUP BY col1
```

Caso 2) `RUP(Dimension,nivel:'instance','datetime')`

En este caso se realiza la selección de una instancia solamente perteneciente al **`nivel`** *To* de la función de rollup, mientras que el intervalo de tiempo en el que la función rollup es válida se puede especificar como un datetime fijo o con la palabra reservada NOW, que va forzar al traductor a evaluar el datetime actual al momento de la traducción y lo va a plasmar como constante en la consulta SQL.

Es decir, la semántica es: RUP devuelve *verdadero* si en **`datetime`** existe un miembro **`instance`** en el nivel **`nivel`**.

Consulta TOLAP-QL

```
SELECT G.provinceId, SUM(amount)
  FROM Prestamos P, Geografia G
 WHERE P.Geografia = G.bottom AND
       RUP(G,provinceId:'1',NOW);
```

Traducción a SQL

```
SELECT col1, SUM(col2) AS col2 FROM (
SELECT Geografia_provinceId col1, SUM(P.amount) col2
  FROM Prestamos_2 P
 WHERE Geografia_provinceId = '1'
   AND instant = convert(datetime, '2006-10-07 15:07:39', 121)
GROUP BY Geografia_provinceId
UNION
SELECT G.provinceId col1, SUM(P.amount) col2
  FROM Prestamos_3 P, DIMENSION_Geografia G
 WHERE P.Geografia_localidadId = G.localidadId
   AND convert(datetime, '2006-10-07 15:07:39', 121)
       BETWEEN G.instant_from AND CASE WHEN G.instant_to IS NULL
                  THEN GETDATE() ELSE G.instant_to END
   AND G.provinceId = '1'
   AND G.provinceId IS NOT NULL
GROUP BY G.provinceId)
UNIONS GROUP BY col1
```

El primer SELECT que pertenece a la versión 2 de la tabla de hechos, en la que se realiza un join optimization (ver sección Capitulo 7 sección 2) y se compara el t de la tabla de hechos contra la constante NOW reemplazada con el datetime actual (`'2006-10-07 15:07:39'`).

El segundo SELECT que pertenece a la versión 3 de la tabla de hechos, agrega la traducción del *RUP* agregando el NOW en lugar de la constante NOW como argumento del BETWEEN.

En ambos casos se agrega el filtro sobre provinceId.

Caso 3) `RUP(Dimension.nivelId,VAR X: VAR x, 'datetime')`

En este caso, lo que se hace es instanciar X y x y si la instanciación hace que la RUP sea *verdadera* entonces se devuelve X y/o x. O sea, se realiza el enlace de los indicadores de tipo VAR para los niveles y para las instancias en forma genérica, informándose *verdadero* si existen funciones de rollups que cumplan la condición.

En la siguiente consulta: *Para cada localidad listar la provincia a la que pertenece,* que en TOLAP-QL se escribe:

```
SELECT localidadId, X, x
  FROM Geografia G
 WHERE RUP(G.localidadId, VAR X:VAR x ,NOW);
```

El traductor agrega un par (nombre de nivel, valor de instancia) por cada expresión *RUP* de este tipo, representando la X al nombre del nivel y la x el valor de la instancia, siendo el primero constante para todos los registros de dicho nivel. A continuación, se transcribe la traducción a SQL.

Traducción a SQL

```
SELECT DISTINCT G.localidadId, 'provinceId' AS provinceId,
       provinceId AS instance_provinceId
  FROM DIMENSION_Geografia G
 WHERE convert(datetime, '2006-10-07 15:26:31', 121)
       BETWEEN G.instant_from AND CASE WHEN G.instant_to IS NULL
                  THEN GETDATE() ELSE G.instant_to END
```

Extracto del resultado

```
1602 provinceId 1
1603 provinceId 1
1605 provinceId 1
1606 provinceId 1
1607 provinceId 1
1609 provinceId 1
1611 provinceId 1
1612 provinceId 1
1613 provinceId 1
1614 provinceId 1
1615 provinceId 1
```

Traducción de expresiones sobre atributos de un nivel

En *TOLAP-QL* las expresiones sobre atributos de un nivel de una dimensión pueden expresarse de dos maneras: por medio de AttributeExpressions o por medio de expresiones que involucren un alias de una instancia de una *RUP* y sobre ella consultar sobre un atributo en particular.

La traducción resulta en una cláusula del tipo:

```
nivel IN (SELECT value
              FROM tinstance_attribute_XX
              WHERE attribute_id = id
                 AND value_at = value
                 AND rupTimeExpression)
```

Nivel	nombre de columna (nivel) que representa la instancia a la cual está asociado el atributo
id	es calculado por el traductor en base a la metadata TOLAP.
value	argumento en la formulación de la consulta.
rupTimeExpression	derivado de la expresión RUP o t fijo.

Veamos la siguiente consulta sobre el modelo de datos expuesto en la Figura 8.19.

Obtener la asistencia financiera brindada por cada línea de financiación en la provincia de 'Buenos Aires'

Que en *TOLAP-QL* se escribe

```
SELECT A.Asistencia, SUM(amount)
  FROM Prestamos P, Asistencias A, Geografia G
 WHERE P.Asistencias = A.bottom AND
       P.Geografia = G.bottom AND
       RUP(A,Asistencia,P.t) AND
       RUP(G,provinceId:p,P.t) AND
       p.nombre = 'Buenos Aires' AND
       P.amount > 0;
```

Debido a que en el modelo de datos, el nombre de la provincia resulta ser una variable de instancia, debemos escribir la siguiente expresión RUP: `RUP(G,provinceId:p,P.t)`. La semántica de dicha expresión es tal que la expresión es verdadera si existe una rollup desde el nivel *bottom* de la dimensión *Geografía* a la instancia *p* del nivel *provinceId* durante el intervalo de validez del bottom de Geografía en la Tabla de Hechos Prestamos. De esta forma, la instancia *p* queda expuesta para determinar si en su variable *nombre* posee el valor buscado. Esto se efectúa por medio de la condición `p.nombre = 'Buenos Aires'`

A continuación podemos ver la correspondiente traducción a SQL

```
SELECT col1, SUM(col2) AS col2 FROM (
       SELECT Asistencias_Asistencia col1, SUM(P.amount) col2
         FROM Prestamos_2 P
        WHERE Geografia_provinceId IN (
                        SELECT value
                          FROM tinstance_attribute_NS
                         WHERE attribute_id = 54
                           AND value_at = 'Buenos Aires'
                           AND P.instant BETWEEN instant_from AND
                               CASE WHEN instant_to IS NULL THEN GETDATE()
                               ELSE instant_to END)
              AND P.amount > 0
       GROUP BY Asistencias_Asistencia
       UNION
       SELECT A.Asistencia col1, SUM(P.amount) col2
         FROM Prestamos_3 P, DIMENSION_Asistencias A, DIMENSION_Geografia G
        WHERE P.Asistencias_Asistencia = A.Asistencia
          AND P.Geografia_localidadId = G.localidadId
          AND P.instant BETWEEN A.instant_from AND
              CASE WHEN A.instant_to IS NULL THEN GETDATE()
              ELSE A.instant_to END
          AND P.instant BETWEEN G.instant_from AND
              CASE WHEN G.instant_to IS NULL THEN GETDATE()
              ELSE G.instant_to END
          AND G.provinceId IN (
                        SELECT value
                        FROM tinstance_attribute_NS
                        WHERE  attribute_id = 54 AND value_at = 'Buenos Aires'
                               AND P.instant BETWEEN instant_from AND
                               CASE WHEN instant_to IS NULL THEN GETDATE()
                               ELSE instant_to END)
         AND P.amount > 0 AND A.Asistencia IS NOT NULL
       GROUP BY A.Asistencia)
UNIONS GROUP BY col1
```

En esta traducción vemos como la condición sobre un atributo de un nivel, expresada por:

```
RUP(G,provinceId:p,P.t) AND p.nombre = 'Buenos Aires'
```

Es traducida mediante la condición:

```
Geografia_provinceId IN (
                    SELECT value
                      FROM tinstance_attribute_NS
                     WHERE attribute_id = 54
                       AND value_at = 'Buenos Aires'
                       AND P.instant BETWEEN instant_from AND
                           CASE WHEN instant_to IS NULL THEN GETDATE()
                           ELSE instant_to END)
```

Traducción de Bloques

El objeto currentParserProfile mantiene un flag para saber si la expresión *RUP* a ser traducida es parte de un bloque que incluye a dicha cláusula. Si el flag es true, se realiza la traducción del bloque a nivel de la traducción de la expresión *RUP*.

Veamos la siguiente consulta sobre el modelo de datos expuesto en la Figura 8.19.

Indicar el total de asistencia financiera brindada por los bancos que poseen como clientes a 'SA LA NACION' y 'EDITORIAL EL ATLANTICO S A'.

La cual se escribe en *TOLAP-QL*

```
SELECT P.EntidadesFinancieras, SUM(P.amount)
  FROM Prestamos P, Deudores D
 WHERE P.Deudores = D.bottom
   AND RUP(D, CUIT:c, P.t) AND c.nombre = 'SA LA NACION'
   AND (RUP(D, CUIT:c1, P.t, c) AND c1.nombre = 'EDITORIAL EL ATLANTICO S A');
```

Traducción a SQL

```
SELECT col1, SUM(col2) AS col2 FROM (
        SELECT EntidadesFinancieras_EntidadId col1, SUM(P.amount) col2
        FROM Prestamos_2 P
        WHERE Deudores_CUIT IN (SELECT value
                          FROM tinstance_attribute_SS
                         WHERE attribute_id = 44 AND value_at = 'SA LA NACION'
                           AND P.instant BETWEEN instant_from AND
                           CASE WHEN instant_to IS NULL THEN GETDATE()
                                  ELSE instant_to END)
              AND EXISTS (SELECT 1
                        FROM Prestamos_2 P1
                        WHERE P.instant = P1.instant
                              AND Deudores_CUIT IN (SELECT value
                                        FROM tinstance_attribute_SS
                                       WHERE attribute_id = 44
                                        AND value_at =
                                        'EDITORIAL EL ATLANTICO S A'
                                         AND P.instant BETWEEN instant_from AND
                                       CASE WHEN instant_to IS NULL THEN
                                       GETDATE() ELSE instant_to END))
        GROUP BY EntidadesFinancieras_EntidadId
        UNION
...
```

Al efectuar dicha traducción el objeto currentParserProfile detecta que ambas expresiones *RUP* pertenecen al mismo bloque, motivo por el cual, debe contemplar que los eventos declarados por ambas expresiones *RUP* suceden al mismo t. Para ello, utiliza la clausula EXISTS ligando la traducción correspondiente a ambas expresiones *RUP* por medio de una subquery que se correlaciona al mismo t.

De esta forma, se pueden realizar comparaciones sobre el mismo atributo, sin tener el usuario que preocuparse por los intervalos de tiempo.

Sentencia sin campos en la cláusula SELECT

TOLAP-QL brinda la posibilidad de escribir un metaconsulta sobre una tabla de dimensión en la que no sean especificados campos para la cláusula SELECT. Esto permite recuperar todas las funciones de rollup que existen desde un nivel a otro de la jerarquía de una dimensión.

La especificación de este tipo de consultas es la siguiente:

```
SELECT
  FROM Dimension D
 WHERE RUP(D, VAR Y, t) AND Y = "NombreNivel";
```

El `NombreNivel` debe ser especificado con un lazo sobre una variable VAR especificando el nombre del levelTo. De esta manera, el traductor buscará las funciones de rollup que estén relacionadas con el levelTo en el t especificado.

A modo de ejemplo citamos la siguiente consulta:

Listar las provincias que componen cada región

Consulta TOLAP-QL

```
SELECT
  FROM Geografia G
 WHERE RUP(G, VAR Y, t) AND Y = 'Región';
```

Traducción a SQL

```
SELECT 'provinceId',provinceId, 'Region',Region
  FROM DIMENSION_Geografia G
  WHERE G.instant_from BETWEEN G.instant_from AND
        CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to END
        AND provinceId IS NOT NULL
        AND Region IS NOT NULL
```

Generación de vistas temporales para STORE

Las consultas con cláusulas STORE son implementadas creando una vista en la base de datos. Dicha vista es utilizada en la consulta que hace referencia a ese STORE. Una vez que el programa *TOLAP* es ejecutado, la vista se remueve de la base de datos.

Una vista en un RDBMS es solamente una definición de una consulta SQL bajo un nombre de objeto, no existe almacenamiento de datos. La implementación de *TOLAP* genera el nombre de la vista basado en un identificador autonumérico.

A modo de ejemplo, se muestra el siguiente programa *TOLAP* y la correspondiente traducción de *TOLAP-QL* a SQL:

Consulta

```
SELECT Pa.yearRange AS year, SUM(F.qty) AS TotalQ1
  FROM Services F, Patient Pa
 WHERE F.Patient = Pa.bottom
   AND RUP(Pa,yearRange,F.t)
STORE AS Q1;
SELECT P.grp, SUM(F.qty)
  FROM Services F, Procedure P, Patient Pa,  Q1
 WHERE F.Procedure = P.bottom
   AND F.Patient = Pa.bottom
   AND RUP(P,grp,F.t) AND RUP(Pa:p,yearRange,F.t)
   AND Q1.year = p.yearRange AND  Q1.TotalQ1 >100;
```

Tabla de Símbolos

```
Scope Global
    Palabra reservada SELECT
    Palabra reservada FROM
    Palabra reservada WHERE
    Palabra reservada COUNT
    Palabra reservada SUM
    Palabra reservada AVG
    Palabra reservada MAX
    Palabra reservada MIN
    Palabra reservada NOT
    Palabra reservada AND
    Palabra reservada OR
    Palabra reservada AS
    Palabra reservada VAR
    Palabra reservada RUP
    Palabra reservada STORE
    Table Q1 Store                          1
        Scope Store
            Alias F Services Fact
            Alias Pa Patient Dimension
            Scope Where
            Column year Pa  String          2
            Column TotalQ1 Double
    Scope Select
        Alias F Services Fact
        Alias P Procedure Dimensión
        Alias Pa Patient Dimensión
        Table Q1 Store                      3
        Scope Where
            Alias p Pa Dimension            4
```

En la estructura de la tabla de símbolos, se puede observar lo siguiente:

1. Se define Q1 como un símbolo de tipo Tabla STORE.
2. Se exportan las columnas year y TotalQ1 en Q1.
3. Se hace uso de la Tabla Q1.
4. El símbolo alias p sobre Pa, en el scope del WHERE, es un alias representando a la dimensión Patient.

Traducción a SQL

Creación de VIEW temporal

TOLAP realiza en este caso la creación de la vista de nombre STORE_176, siendo 176 un identificador auto numérico.

```
CREATE VIEW STORE_176 AS
SELECT year, SUM(TotalQ1) AS TotalQ1
  FROM (
  SELECT Pa.yearRange AS year, SUM(qty) AS TotalQ1
     FROM Services_1 F, DIMENSION_Patient Pa
     WHERE F.Patient_patientId = Pa.patientId AND instant
       BETWEEN Pa.instant_from AND CASE WHEN Pa.instant_to IS NULL
                  THEN GETDATE() ELSE Pa.instant_to END AND Pa.yearRange IS
            NOT NULL GROUP BY Pa.yearRange
  UNION
  SELECT Pa.yearRange AS year, SUM(qty) AS TotalQ1
       FROM Services_2 F, DIMENSION_Patient Pa
       WHERE F.Patient_patientId = Pa.patientId AND instant
            BETWEEN Pa.instant_from AND CASE WHEN Pa.instant_to IS NULL
            THEN GETDATE() ELSE Pa.instant_to END AND Pa.yearRange IS
                  NOT NULL GROUP BY Pa.yearRange
  UNION
  SELECT Pa.yearRange AS year, SUM(qty) AS TotalQ1
       FROM Services_3 F, DIMENSION_Patient Pa
       WHERE F.Patient_patientId = Pa.patientId AND instant BETWEEN
            Pa.instant_from AND CASE WHEN Pa.instant_to IS NULL THEN
                  GETDATE() ELSE Pa.instant_to END AND Pa.yearRange IS
            NOT NULL GROUP BY Pa.yearRange)
UNIONS GROUP BY year
```

La vista temporal `STORE_176` es luego utilizada en la consulta, reemplazando las apariciones de Q1.

```
SELECT col1, SUM(col2) AS col2
  FROM (
        SELECT P.grp col1, SUM(qty) col2
            FROM Services_1 F, DIMENSION_Procedure P, DIMENSION_Patient
                    Pa, STORE_176
            WHERE F.Procedure_procedureId = P.procedureId AND
              F.Patient_patientId = Pa.patientId AND instant BETWEEN
              P.instant_from AND CASE WHEN P.instant_to IS NULL THEN
              GETDATE() ELSE P.instant_to END AND instant BETWEEN
              Pa.instant_from AND CASE WHEN Pa.instant_to IS NULL THEN
              GETDATE() ELSE Pa.instant_to END AND STORE_176.year =
              Pa.yearRange AND STORE_176.TotalQ1 > 100 AND P.grp IS NOT
                        NULL
            GROUP BY P.grp
        UNION
        SELECT P.grp col1, SUM(qty) col2
            FROM Services_2 F, DIMENSION_Procedure P, DIMENSION_Patient
                    Pa, STORE_176
            WHERE F.Procedure_procedureId = P.procedureId AND
              F.Patient_patientId = Pa.patientId AND instant BETWEEN
              P.instant_from AND CASE WHEN P.instant_to IS NULL THEN
              GETDATE() ELSE P.instant_to END AND instant BETWEEN
              Pa.instant_from AND CASE WHEN Pa.instant_to IS NULL THEN
              GETDATE() ELSE Pa.instant_to END AND STORE_176.year =
              Pa.yearRange AND STORE_176.TotalQ1 > 100 AND P.grp IS NOT
                        NULL
            GROUP BY P.grp
        UNION
        SELECT P.grp col1, SUM(qty) col2
            FROM Services_3 F, DIMENSION_Procedure P, DIMENSION_Patient
                    Pa, STORE_176
            WHERE F.Procedure_procedureId = P.procedureId AND
              F.Patient_patientId = Pa.patientId AND instant BETWEEN
              P.instant_from AND CASE WHEN P.instant_to IS NULL THEN
              GETDATE() ELSE P.instant_to END AND instant BETWEEN
              Pa.instant_from AND CASE WHEN Pa.instant_to IS NULL THEN
              GETDATE() ELSE Pa.instant_to END AND STORE_176.year =
              Pa.yearRange AND STORE_176.TotalQ1 > 100 AND P.grp IS NOT
                    NULL
          GROUP BY P.grp)
  UNIONS GROUP BY col1
```

Resultado

```
g1  1583.00
g2  110.00
```

En el Anexo 3 se exhiben más casos de consultas *TOLAP-QL*, con sus correspondientes tablas de símbolos, traducción y resultado obtenido.

6.4 Resumen

En este capítulo estudiamos en detalle el procesamiento de consultas TOLAP-QL. Entre otros aspectos vimos como se realiza el manejo del árbol sintáctico y posteriormente el análisis semántico. Por último se mostró como se traducen las expresiones TOLAP-QL a SQL.

Capítulo 7.

OPTIMIZACION DE CONSULTAS EN TOLAP-QL

Este capítulo describe la optimización de consultas de *TOLAP* implementado en esta tesis.

Las técnicas que utiliza este optimizador son:

- Descarte de versiones de una tabla de hechos.
- Optimización de Joins.
- Query pruning.
- Optimización por medio de Vistas Materializadas

7.1 Descarte de versiones de una tabla de hechos

En el proceso de traducción de *TOLAP-QL* a SQL no se genera una clausula SELECT para una versión de una Tabla de hechos si alguno de los niveles pertenecientes a la sentencia SELECT no están dentro del intervalo de tiempo definido para la versión de dicha tabla.

Para que una versión sea descartada, deben cumplirse al menos una de las dos condiciones siguientes:

a) Uno o más niveles no están definidos en todo el intervalo de la versión que está siendo traducida.
b) Existen restricciones sobre el instante de la tabla de hechos, tal que estas restricciones se verifican totalmente fuera del intervalo correspondiente a la versión.

Ejemplificamos el caso a) con la siguiente consulta:

Cantidad de préstamos otorgados en la provincia de Buenos Aires.

Que en *TOLAP-QL,* teniendo en cuenta la definición de las distintas versiones de la tabla de hechos *Préstamos* (ver Figuras 8.19, 8.20 y 8.21), se escribe:

```
SELECT G.provinciaId, COUNT(*)
  FROM Prestamos P, Geografia G
 WHERE P.Geografia = G.bottom
   AND RUP(G, provinciaId:prov, P.t)
   AND prov.nombre(P.t) = 'Buenos Aires'
   AND P.amount > 0;
```

En la primera versión de *Préstamos*, el nivel *provinciaId* no está definido en la versión 1 (ver Figura 8.19). Luego, sobre la dimensión *Geografía* se realiza un *Specialize*, cambiando el nivel ínfimo *regionId* por *provinciaId*, lo que hace que se genere una nueva versión de la tabla de hechos (la versión 2). Por lo tanto, la versión 2 tiene como ínfimo a *provinciaId* y se debe incluir en la traducción.

La versión 3 de la tabla de hechos *Préstamos* se crea a partir de un *Specialize* sobre *Geografía*, que cambia el ínfimo de *provinciaId* a *localidadId*. Esta versión 3 (cuyo ínfimo no es *provinciaId*) no se incluye en la traducción, agregando en el clausula FROM la tabla de instancias de la dimensión *Geografía* haciendo un join con la versión 3 por el ínfimo correspondiente a esa versión. En este caso, la condición de junta es: `Geografia.localidadId = Prestamos.localidadId`. Luego en los usos de *provinciaId*, se va a referenciar Geografia.provinciaId, la tabla de instancias correspondiente a la *dimensión*.

Traducción a SQL

```
SELECT col1, SUM(col2) AS col2
  FROM (

/* Versión 2 */

  SELECT Geografia_provinciaId col1, COUNT(*) col2
    FROM Prestamos_2 P
   WHERE Geografia_provinciaId IN
              (SELECT value
                    FROM tinstance_attribute_NS
                    WHERE  attribute_id = 54 AND value_at = 'Buenos Aires' AND
                           P.instant BETWEEN instant_from AND
                                CASE WHEN instant_to IS NULL
                                      THEN GETDATE() ELSE instant_to END)
     AND P.amount > 0
  GROUP BY Geografia_provinciaId

  UNION

/* Versión 3 */

  SELECT G.provinciaId col1, COUNT(*) col2
    FROM Prestamos_3 P, DIMENSION_Geografia G
   WHERE P.Geografia_localidadId = G.localidadId
     AND P.instant BETWEEN G.instant_from AND
         CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to END
     AND G.provinciaId IN
               (SELECT value
                    FROM tinstance_attribute_NS
                    WHERE  attribute_id = 54 AND value_at = 'Buenos Aires' AND
                           P.instant BETWEEN instant_from AND
                                CASE WHEN instant_to IS NULL
                                      THEN GETDATE() ELSE instant_to END)
     AND P.amount > 0
     AND G.provinciaId IS NOT NULL
  GROUP BY G.provinciaId)
UNIONS GROUP BY col1
```

En negrita se aprecia que en la traducción sobre la versión 2 se utiliza directamente la tabla de hechos *Préstamos* y su tabla de instancias correspondiente a la versión, en este caso Prestamos_2. Para la versión 3, se resaltaron el agregado de la tabla DIMENSION_Geografia y el join con la tabla Prestamos_3. Luego en los usos de provinciaId, se utiliza la columna provinciaId de la tabla de instancias de *Geografía*.

Para el caso b), el método `isFactTableVersionInTimeConstraints` es invocado en el ciclo de la traducción para detectar si una versión debe ser parte o no de la traducción, dada las restricciones sobre el instante de la tabla de hechos. Este método toma en cuenta las restricciones temporales que fueron almacenadas durante el análisis sintáctico en el ParserProfile y luego se hace la conjunción de ellos y se comparan contra el intervalo de la versión de la tabla de hechos.

Éste es un caso de *query pruning*, una de las optimizaciones básicas propuestas para *TOLAP-QL*. En la sección 7.3 se dan más detalles sobre este método.

7.2 Optimización de joins

La optimización de joins consiste en la eliminación de todas las tablas de dimensión en la traducción, que pueden ser resueltas a través de los niveles intervinientes (ínfimos de dimensión) en la versión de la tabla de hechos correspondiente a la versión que está siendo traducida.

Si el optimizador detecta que puede aplicarse la optimización de joins, se realizan luego las siguientes operaciones sobre la traducción a SQL resultante:

- Se eliminan las tablas de dimensiones de la clausula FROM, dejándose solamente la versión de la tablas de hechos correspondiente
- Se eliminan los EqualityExpression, las igualdades que representan los joins.
- Se elimina el producto de la traducción de las cláusulas *RUP*.

El siguiente algoritmo detecta si existe la posibilidad de aplicar Join Optimization

Algoritmo hasJoinOptimization

```
inicio
      para cada nivel en select head
            si nivel no es una función aggregate
                  bottom = obtenerBottom(nivel.dimension, version.instantFrom)
                  si (bottom.name != nivel.name)
                        // El nivel que está en el SELECT es un ínfimo
                        // El ínfimo está siempre en la tabla de hechos
                         (supuestamente siempre porque el JOIN con las dimensiones
                              es explícitamente hecho en el FROM)
                        return false;
                  end if
            end if
      fin para
      para cada infoNode in currentParserProfile.usesLevelIds
            if (infoNode.isFactTableNode)
                  // Se busca ínfimo desde instant from de la version de la
                  //                                    table de hechos.
                  bottom = getBottom(infoNode.dimension, version.instantFrom)
                  if (bottom.name != infoNode.columnName)
                        // El nivel que está en el SELECT es un ínfimo
                        // El ínfimo está siempre en la tabla de hechos
                        (supuestamente siempre porque el JOIN con las dimensiones
                         es explícitamente hecho en el FROM)
                        return false;
                  fin si
            fin si
      fin para
fin
```

Veamos el siguiente ejemplo, tomado de una consulta efectuada sobre el modelo de Ventas detallado en la Figura 4.11:

Total de ventas por ciudad

Que en TOLAP-QL expresamos:

```
SELECT  G.ciudad, SUM(amount)
  FROM Ventas V, Geografia G
 WHERE V.Geografia=G.bottom;
```

Traducción a SQL

Para la versión 1 de la tabla de hechos *Ventas*, Ventas_1, se puede eliminar el join, ya que para esa versión ciudad, como ínfimo de *Geografía,* pertenece a *Ventas*. Además, *ciudad* está en la cláusula `SELECT`. Para el resto de las versiones, al cambiar el ínfimo, es necesario realizar el join.

```
SELECT col1, SUM(col2) AS col2
  FROM (
        SELECT Geografia_ciudad col1, SUM(amount) col2
          FROM Ventas_1 F
          GROUP BY Geografia_ciudad
        UNION
        SELECT G.ciudad col1, SUM(amount) col2
          FROM Ventas_2 F, DIMENSION_Geografia G
          WHERE F.Geografia_pueblo = G.pueblo AND G.ciudad IS NOT NULL
          GROUP BY G.ciudad
        UNION
        SELECT G.ciudad col1, SUM(amount) col2
          FROM Ventas_3 F, DIMENSION_Geografia G
          WHERE F.Geografia_manzana = G.manzana AND G.ciudad IS NOT NULL
          GROUP BY G.ciudad)
UNIONS GROUP BY col1
```

7.3 Poda de consultas (Query pruning)

Si una consulta *TOLAP-QL* contiene una condición con una restricción temporal, tal que el ciclo de vida de alguna versión de la tabla de hechos involucrada no interseca el intervalo determinado por dicha restricción, la traducción a SQL correspondiente a dicha subconsulta no es generada, ya que no existen tuplas resultantes de tal subconsulta. A dicha eliminación de una subconsulta la denominamos *query pruning*. Veamos el siguiente ejemplo:

Total de ventas por marca para la región r1 desde 2006/02/01

```
SELECT P.marca,SUM(amount)
  FROM Ventas F, Producto P, Geografia G
 WHERE F.Producto = P.bottom
   AND F.Geografia =G.bottom
   AND RUP(G,region:'r1',F.t)
   AND RUP(P,marca,F.t)
   AND F.t >= '2006/02/01';
```

Traducción a SQL

```
SELECT col1, SUM(col2) AS col2
  FROM (
       SELECT P.marca col1, SUM(amount) col2
        FROM Ventas_2 F, DIMENSION_Producto P, DIMENSION_Geografia G
        WHERE F.Producto_itemId = P.itemId AND F.Geografia_pueblo = G.pueblo
              AND instant BETWEEN G.instant_from AND
              CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to END
              AND G.region = 'r1' AND instant BETWEEN P.instant_from AND
              CASE WHEN P.instant_to IS NULL THEN GETDATE()
                     ELSE P.instant_to END
              AND P.marca IS NOT NULL
        GROUP BY P.marca
      UNION
       SELECT P.marca col1, SUM(amount) col2
        FROM Ventas_3 F, DIMENSION_Producto P, DIMENSION_Geografia G
        WHERE F.Producto_itemId = P.itemId AND
              F.Geografia_manzana = G.manzana
              AND instant BETWEEN G.instant_from AND
              CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to END
              AND G.region = 'r1' AND instant BETWEEN P.instant_from AND
              CASE WHEN P.instant_to IS NULL THEN GETDATE()
                     ELSE P.instant_to END
              AND P.marca IS NOT NULL
        GROUP BY P.marca)
UNIONS GROUP BY col1
```

Se observa que en este caso, la subconsulta sobre la tabla Ventas_1, versión 1 de la tabla de hechos Ventas, es eliminada de la traducción debido a que por la restricción `F.t >= '2006/02/01'` ninguna de las tuplas de dicha tabla de hechos seria afectada.

7.4 Optimización por medio de Vistas Materializadas

Una *vista materializada* es una vista tal que sus tuplas se encuentran almacenadas en la base de datos, mejorando de esta forma la performance de las consultas que la utilizan. Las *vistas materializadas* pueden ser vistas como un cache el cual puede ser directamente accedido sin tener que adentrarse en las relaciones de la base de datos.

La utilización de este modelo de aproximación para la mejora de tiempos de respuesta es fundamental debido a que las consultas comunes al warehouse son de tipo agregadas requiriendo eventualmente un scan completo sobre las filas de las tablas de hechos. Supongamos una aplicación de warehouse sobre un modelo como el descripto en la Figura 4.11, los usuarios del mismo podrían realizar con suma frecuencia consultas del tipo:

El monto total vendido por categoría de producto para la región r1

```
SELECT P.categoria, SUM(cantidad*precio) as MontoVenta
  FROM Ventas V, Producto P, Geografía G
 WHERE V.Producto = P.bottom
   AND V.Geografia = G.bottom
   AND RUP(G,region:'r1',F.t)
   AND RUP(P,categoria,F.t);
```

La siguiente tabla muestra parte del resultado:

P.Categoria	MontoVenta
Calzado	$150.000
Ropa Vestir	$1.320.500
Otros	$9.323.597

Obviamente, obtener el resultado anterior, requiere del RDBMS un esfuerzo muy superior (proporcional al tamaño de *Ventas*, *Producto* y *Geografía*) al de realizar la siguiente consulta:

```
SELECT categoria, MontoVenta
 FROM VM_CR;
```

donde *VM_CR* es la vista materializada definida en base a la **¡Error! No se encuentra el origen de la referencia.**.

En este simple concepto se basa el tipo de optimización que hemos implementado y que trataremos en esta sección.

En el Capitulo 3 se introdujo la noción de *tablas de hechos* y las versiones que las componían. Como resultado de esto, la información de una *tabla de hechos* contenida en una versión anterior a la vigente, es de carácter estático, implicando que las tuplas resultantes de una consulta sobre las mismas son invariantes. Por este motivo, no es necesario implementar un mecanismo de mantenimiento de *vistas materializadas*, siempre que las mismas no tengan referencia a la versión abierta/actual de las *tablas de hechos*, ya que se trata de datos que no van a cambiar. *Debido a lo expuesto, esta implementación solo utiliza vistas materializadas contra las versiones cerradas de las tablas de hechos.*

7.4.1 Modelo de datos para administración de vistas materializadas

El modelo de datos utilizado para almacenar las definiciones de *vistas materializadas* existentes así como también las candidatas se basa en las siguientes entidades:

- **TVM(vm_id, vm_name, mftv_id, hits, creada, last_update, reg_count)**

  Contiene el detalle de las *vistas materializadas* candidatas y las existentes en el sistema. La diferenciación entre ambas se realiza por medio del atributo "creada" (1:existe ; 0:candidata). Para cada vista materializada, el atributo mftv_id nos permite identificar la tabla de hechos asociada así como su correspondiente versión. El atributo "last_update" guarda el instante del último acceso realizado a dicha *vista materializada* y es utilizado con el fin de poder purgar las vistas que llevan más de X tiempo sin utilizarse (donde X es el tiempo máximo de envejecimiento permitido por el sistema y que se encuentra parametrizado para permitir la personalización por parte del Usuario).

- **TVM_DET(vm_id, level_id, is_in_ft)**

  Contiene el detalle de los niveles involucrados en cada *vista materializada*, así como una indicación de si se trata o no de un ínfimo.

7.4.2 Creación de vistas materializadas

Para la creación automática de *vistas materializadas*, se ha implementado un proceso que para toda consulta *TOLAP-QL* ejecutada, almacena en el modelo detallado en 7.4.1 la información de qué niveles han sido involucrados en la misma, así como un contador de la cantidad de veces que se ha utilizado el mismo conjunto de niveles en la resolución de una consulta e indica que se trata de una vista candidata. De esta forma, el sistema va llevando una estadística de uso y en el momento que lo considere adecuado (ver 7.4.4) procederá a efectuar la materialización de dicha vista candidata.

Además de la creación automática de *vistas materializadas*, cuando el usuario así lo considere, puede optar por crearlas manualmente. Para ello, se brinda un asistente de creación de vistas materializadas (ver Figura 5.18), el cual permite seleccionar una tabla de hechos y su correspondiente versión, y dado la estructura de ella, se proponen las combinaciones de los niveles de la tabla de hechos conjuntamente con los todos los niveles de las dimensiones relacionadas (no ínfimos) que existan en el

intervalo correspondiente a aquella versión. Luego el usuario puede elegir una de las combinaciones para ser creada automáticamente.

7.4.3 Thread de control para creación de vistas materializadas automáticamente

Para llevar a cabo la creación automática de las *vistas definidas* y aun no materializadas, se definió una tarea en Tomcat (implementada mediante servlets), parametrizada por un retardo temporal 't'. De esta forma cada 't' milisegundos el servlet invoca al OptimizerAgent para que verifique las *vistas materializadas* definidas que tienen una cantidad de hits superior al parámetro definido por el usuario para realizar la materialización. En caso de detectarse alguna, el OptimizerAgent dispara en background su generación actualizando el estado de dicha *vista materializada* a *creada* una vez finalizada la misma.

En este mismo momento el OptimizerAgent realiza la purga de las vistas materializadas para las cuales el atributo *last_update* ha superado el tiempo máximo de envejecimiento permitido por el sistema (parámetro definido por el usuario), liberando de esta forma disco que no está siendo utilizado.

7.4.4 Creación automática de vistas materializadas a partir de coeficiente basado en frecuencia de uso

TOLAP crea automáticamente las *vistas materializadas* que cumplan con un coeficiente, un valor fijo como parámetro de sistema, contra el cual se compara la cantidad de hits realizados sobre una consulta estadísticamente sobre los identificadores de nivel utilizados en las consultas.

Por ejemplo, se realizan consultas que involucran a los niveles *city* y *storeId* n veces y n >= m, con m parámetro del sistema que expresa la cantidad mínima de consultas que cumplan con la frecuencia de la selección de los mismos identificadores de niveles. En este caso, se procede a crear la *vista materializada* automáticamente a partir del thread de creación.

Los datos de los identificadores de nivel y tablas de hechos utilizados en cada consulta realizada, son almacenados en el mismo modelo de datos descripto anteriormente, indicando que la *vista materializada* es candidata: tabla TVM columna creada: (1:existe ; 0:candidata).

7.4.5 Generación de la Vista Materializada

Una vez detectada una *vista materializada* en condiciones de ser creada, el OptimizerAgent dispara la creación de la tabla que alojará el conjunto resultante de registros comprendidos en la definición de la misma. Cada campo de la nueva tabla tendrá el tipo de datos correspondiente al nivel al cual esta mapeado.

Para el llenado propiamente dicho, se genera una instrucción SQL insert-select la que contendrá todos los niveles elegidos más las funciones de agregación sobre la medida de la tabla de hechos asociada. Para la cláusula FROM, con cada nivel involucrado en la *vista materializada* que no es ínfimo en el 't' de la versión de la tabla de hechos utilizada (por lo cual no se haya en la tabla de hechos), se procede a hacer el correspondiente join con la dimensión a la que pertenece.

Debido a que la creación de la *vista materializada* involucra una scan de la *tabla de hechos*, se aprovecha para crear todos los agregados posibles (aunque en la definición de la *vista materializada* no hayan sido especificados).

7.4.6 Reescritura de consultas utilizando vistas materializadas.

Condiciones que debe reunir una consulta para poder ser reescrita en base a vistas materializadas

Las *vistas materializadas* solo son generadas sobre las versiones de *tablas de hechos* anteriores a la que esta generada hasta NOW (debido a que sino deberíamos estar haciendo updates de esta *vistas materializadas* en la medida en que la tabla de hechos se vea modificada). Por dicho motivo todas las consultas que involucran a NOW o tienen como resultado un conjunto de datos que existen en la tabla de hechos actual no son contempladas.

Por otro parte, la utilización en las consultas *TOLAP* de atributos de instancia implica que se debe entrar a la tabla de hechos buscando por el 't' de atributos de instancias los posibles cambios existentes, perdiendo de esta forma la posibilidad de realizar agregados a un nivel.

Además, como los agregados efectuados sobre las *vistas materializadas* no contemplan el 't', las consultas que las utilicen deben incluir totalmente al intervalo de la versión de la tabla de hechos.

Obtención de vistas materializadas candidatas para hacer la reescritura de una consulta solicitada

Una vez determinado si la consulta puede ser reescrita en base a *vistas materializadas*, se realiza una solicitud al OptimizerAgent para que éste devuelva el conjunto de *vistas materializadas* que cubren completamente los niveles involucrados en la consulta *TOLAP*. Luego, se procede a elegir la de mínima cobertura, que es devuelta al Parser para que efectúe la reescritura.

Finalmente, el OptimizerAgent actualiza las estadísticas de hits efectuados a las *vistas materializadas*.

Proceso de reescritura de la consulta original a partir de la vista materializada elegida por el optimizador

En este proceso el Parser realiza las redefinición de la traducción en base a los identificadores de niveles de la *vista materializada* elegida, realizando *join optimization* luego de realizar la reescritura siempre que fuese posible.

Sea el ejemplo:

Monto total por categoría de producto para la región r1

```
SELECT P.marca, SUM(monto)
  FROM Ventas F, Producto P, Geografia G
 WHERE F.Producto = P.bottom
   AND F.Geografia = G.bottom
   AND RUP(G,region:'r1',F.t)
   AND RUP(P,marca,F.t);
```

Supongamos que previamente existe una *vista materializada* llamada VMTest que se compone de los siguientes niveles, que en la metadata TOLAP están representados internamente por identificadores de niveles generados por la base de datos:

VMTest, sobre la table de hechos Sales versión 1

Product.itemId	(ínfimo en versión 1)
Product.brand	
Geography.city	(ínfimo en versión 1)
Geography.region	

El proceso de reescritura consiste en los siguientes pasos (adaptados al ejemplo dado):

- Chequear que la *vista materializada* debe poder utilizarse: el traductor chequea las condiciones que se enumeraron más arriba.

- El traductor envía al Optimizer Agent una lista de identificadores de niveles para los usos que son registrados en la consulta actual. Los identificadores de niveles corresponden a los que se usan en la consulta tanto en el SELECT como en el WHERE. Con esos usos se conforma una lista a ser pasada al Optimizer Agent.

  En este caso se conforma una lista con los identificadores de niveles correspondientes a cada versión:

  Version 1: { Product.itemId (bottom), Product.brand,
  Geography.city (bottom), Geography.region }
  Version 2: { Product.itemId (bottom), Product.brand,
  Geography.town (bottom), Geography.region }

  La Versión 3 no aplica porque es la actual.

- Si el Optimizer Agent devuelve una *vista materializada* (si existen varias candidatas a ser utilizadas, la mejor) para realizar la reescritura, el traductor reemplaza la traducción de cada level id por la correspondiente a la imagen SQL del uso de la columna de la *vista materializada*, y se eliminan los joins con las tablas de *dimensiones* pertenecientes al FROM original, ya que la cobertura de identificadores de niveles es completa.

La consulta luego de la reescritura, se lee:

```
SELECT col1, SUM(col2) AS col2
  FROM (
        SELECT Product_brand col1, SUM(MEASURE_SUM) col2
               FROM VMTest
               WHERE Geography_region = 'r1' GROUP BY Product_brand
        UNION
        SELECT P.brand col1, SUM(amount) col2
               FROM Sales_2 F, DIMENSION_Product P, DIMENSION_Geography G
               WHERE F.Product_itemId = P.itemId
                     AND F.Geography_town = G.town
                     AND instant BETWEEN G.instant_from AND
                     CASE WHEN G.instant_to IS NULL
                          THEN GETDATE() ELSE G.instant_to END
                     AND G.region = 'r1' AND instant BETWEEN P.instant_from AND
                     CASE WHEN P.instant_to IS NULL
                          THEN GETDATE()ELSE P.instant_to END
                     AND P.brand IS NOT NULL
               GROUP BY P.brand
        UNION
        SELECT P.brand col1, SUM(amount) col2
               FROM Sales_3 F, DIMENSION_Product P, DIMENSION_Geography G
               WHERE F.Product_itemId = P.itemId AND
                     F.Geography_square = G.square AND instant BETWEEN
                     G.instant_from AND CASE WHEN G.instant_to IS NULL
                           THEN GETDATE() ELSE G.instant_to END
                     AND G.region = 'r1' AND instant BETWEEN P.instant_from AND
                     CASE WHEN P.instant_to IS NULL
                          THEN GETDATE() ELSE P.instant_to END
                     AND P.brand IS NOT NULL
               GROUP BY P.brand)
 UNIONS GROUP BY col1
```

En la primera versión se realiza la reescritura, utilizando VMTest. Se eliminan los joins a la dimensión Product y a la dimensión Geography. La instancia 'r1' de región es parte del WHERE. La versión 2 no ofrece reescritura por no existir una vista materializada adecuada y la 3, como es la versión actual, no aplica.

7.5 Resumen

En este capítulo, se detallaron los tipos de optimización estudiados y aplicados en esta implementación, a saber: descarte de versiones de una tabla de hechos, optimización de joins, query prunning y vistas materializadas. Se mostro la aproximación tomada para el manejo de vistas materializadas (solo se crean sobre versiones cerradas de las tablas de hechos) y como las mismas eran utilizadas en la reescritura de consultas.

Capítulo 8.

CASO DE ESTUDIO: CENTRAL DE INFORMACIÓN DE DEUDORES DEL SISTEMA FINANCIERO

El Banco Central de la República Argentina es una entidad autárquica del Estado nacional, cuya misión primaria y fundamental es preservar el valor de la moneda y entre otras de sus funciones ejerce la supervisión de la actividad financiera y cambiaria por intermedio de la Superintendencia de Entidades Financieras y Cambiarias (SEFyC), la que depende directamente del Presidente de la Institución. Es en el ámbito de la SEFyC que con la finalidad de brindar transparencia crediticia al mercado financiero y facilitar el ingreso al crédito por parte del público se creó la Central de Deudores del Sistema Financiero (CDSF).

La CDSF es una base de datos constituida por personas físicas y jurídicas 'bancarizadas' (que han tomado créditos bancarios, que son poseedores de tarjetas de crédito, descubiertos, cartas de crédito y toda operación que implique la utilización de valores que no sea efectivo ni cheques emitidos de cuentas con fondos propios). En ella se presenta la calificación bancaria de dichas personas de acuerdo a los niveles de deuda que poseen y a sus conductas de pago. La CDSF es de difusión pública y posee solamente parte de la última información brindada por las Entidades Financieras (EF) respecto de sus deudores, la cual es remitida por las EF a la SEFyC por medio de un régimen informativo denominado Deudores del Sistema Financiero (RI-DSF).

El warehouse en el cual reside la información del RI-DSF es utilizado (además de nutrir a la CDSF) para el análisis tanto de la cartera crediticia de las EF como así también para el seguimiento del comportamiento crediticio de ciertos individuos o grupos de ellos en el tiempo, la realización de credit scoring sobre grupos de deudores para determinar si la clasificación otorgada por las EF a sus clientes es correcta (debido a que desde el punto de vista de la Supervisión se debe asegurar que la entidad previsiono correctamente el crédito otorgado en función de las posibilidades de repago del deudor) y todo tipo de análisis crediticio tendiente a determinar las condiciones actuales del mercado crediticio a nivel comunal, provincial, regional, nacional y su posible evolución futura.

Desde un punto de vista de modelización, este warehouse sustenta un modelo de datos comparable con cualquier otro en donde existan Entidades brindadoras (EF), Entidades solicitantes (deudores) y operaciones entre ellas (detalles de las deudas/créditos solicitados), que además constituye un sistema del tipo VLDB (en la actualidad posee información de más de 12 millones de deudores a lo

largo de últimos 10 años de historia y algunas de sus estadísticas indican que la información es almacenada en más de 200 Gb en disco con tablas del sistemas con más de 1300 millones de registros).

Uno de los mayores inconvenientes a la hora de realizar consultas de índole geográfico en este modelo, radica en la evolución que ha sufrido el detalle de la información remitida por las EF, la cual paso en el termino de 3 (tres) años, de referenciar la región comercial en la que se había originado el crédito informado para luego indicar en qué provincia y por último en que localidad. Motivo que nos fuerza a realizar una comparación temporal para saber que localidades pertenecían o no, en fecha dada, a una región comercial en particular cada vez que se realiza una consulta geográfica histórica. Nótese que en el interior del país, las localidades han sufrido cierta evolución en el transcurso de los últimos 10 años, pasando en muchos casos, parajes a conformar nuevas localidades.

8.1 Modelo de experimentación

Para nuestro modelo de experimentación se tomo un subconjunto del warehouse correspondiente al caso de estudio que abarca las operaciones registradas en el periodo 2003-01-01 al 2005-12-31 (ya que dentro de este rango temporal se produce las modificaciones estructurales en las dimensiones del modelo (desagregaciones geográficas de la información). Además se desestimo información de atributos relevantes a las garantías y previsiones del crédito, debido a que a los efectos de este estudio no introducían ninguna variante cualitativa, pero determinaban un incremento critico en el volumen y tamaño físico de la información a tratar.

En este modelo, una tupla en la tabla Prestamos (Tabla de hechos) representa un crédito/pago (dependiendo del signo del importe) brindado/recibido por una EF a/de un deudor en una línea especifica de crédito (por ejemplo: tarjeta de crédito, cuenta corriente, etc.) en una ubicación geográfica del país en un día dado y el cual mantiene en virtud de la normativa vigente una calificación, denominada situación.

Para representar los cambios temporales que impactaron al modelo del caso de estudio respecto de la exactitud de la ubicación geográfica registrada se crearon 3 versiones de la Tabla de hechos a diferencia del caso de estudio (mundo real), en el cual la Tabla de hechos es única y ante cada uno de los cambios descriptos solo se procedió a grabar en el campo representativo de la ubicación geográfica el valor correspondiente a ese atributo (Por ejemplo, la deuda de "Montoya" con el Banco Provincia de Bs As, en el 2003/06/30 aparecería informada como perteneciente a la región comercial "Pampeana", mientras que la información del mismo crédito en el año 2004 aparece como informada en la provincia de Buenos Aires y en el año 2005 ya aparece como tomada en la localidad de "Ensenada").

8.2 Ventajas del Modelo Temporal propuesto

En el modelo del caso de estudio, las consultas agregadas de tipo histórico sufren problemas al tratar de captar datos que en principio han quedado mal clasificados debido a los cambios estructurales que ha sufrido el modelo. Por ejemplo, si en el caso de estudio planteáramos la siguiente consulta

```
SELECT G.provinciaId, SUM(monto)
FROM Prestamos P, Geografia G
WHERE P.zona = G.provinciaId and G.provinciaId = 'Buenos Aires'
GROUP BY G.provinciaId
```

Con la finalidad de obtener el total de préstamos brindados para la de 'Buenos Aires', tendríamos como inconveniente que la misma solo resolvería correctamente para la información brindada durante el año 2004, mientras que la consulta perdería la información referente al año 2005 debido a que en el campo zona de la tabla de hechos Prestamos, se integra la ciudad para la información referente al año 2005 (en cuanto al año 2003 no se posee el dato en el warehouse por lo cual resulta imposible su cálculo). Debido a esto el usuario debe realizar consultas especiales como la detallada a continuación, para obtener los resultados correctos.

```
SELECT provinciaId, SUM(monto) FROM (
 SELECT P.provinciaId, SUM(monto)
 FROM Prestamos P
 WHERE P.zona = 'Buenos Aires'  and P.dia >= '2004-01-01' and
       P.dia <= '2004-12-31'
 GROUP BY P.provinciaId
UNION
 SELECT G.provinciaId, SUM(monto)
 FROM Prestamos P, Geografia G
 WHERE P.zona = G.ciudad and G.provinciaId = 'Buenos Aires'
 and P.dia >= '2005-01-01'
 GROUP BY G.provinciaId
 )  U (provinciaId, monto)
GROUP BY U.provinciaId
```

Nuestro modelo temporal ha sido diseñado para contemplar los problemas típicos debido a cambios estructurales en la definición del Warehouse, permitiendo al Usuario realizar sus consultas de una forma más natural y sencilla mientras que se optimiza la ejecución de las mismas.

El modelo propuesto para el caso de estudio, consta de tres versiones de una tabla de hechos originadas debido a los cambios del ínfimo de la Dimensión geografía. En nuestro modelo temporal la

consulta se formularía con una sintaxis SQL-LIKE mucho más sencilla de entender. A continuación se la transcribe como ejemplo:

```
SELECT G.provinciaId, SUM(monto)
FROM Prestamos P, Geografia G
WHERE P.zona = G.bottom
and RUP(G, provinciaId:'Buenos Aires', P.t)
```

En esta consulta, nuestro modelo tiene 3 tipos distintos de optimización aplicables: Query prunning, join optimization y utilización de Vistas materializadas. Query prunning se aplica debido a que en la versión 1 de Prestamos, la dimensión geografía no contenía al nivel provinciald. El join optimization es aplicado al detectar que durante el ciclo de vida de la versión 2 de Prestamos, provinciald es el ínfimo de la Dimensión Geografía. Existe la posibilidad de materializar una Vista por provinciald sobre la versión 2 de Prestamos para utilizarla en la resolución.

8.3 Demografía de los datos

A continuación se detalla el modelo de datos y las distintas versiones que lo integran, así como un grafico estadístico de los datos en cuestión.

Tabla	# de filas	Espacio físico utilizado (datos brutos)
Geografía	22.887	<1 MB
Situaciones	7	<1 MB
Deudores	6.054.776	276.8 MB
Tiempo	3.289	<1 MB
EntidadesFinancieras	700	<1 MB
Asistencias	19	<1 MB
Prestamos_1	45.385.538	2706.8 MB
Prestamos_2	51.614.937	3078.2 MB
Prestamos_3	70.846.431	5528.6 MB

8.4 Evolución del modelo de datos

Mostramos aquí cómo evoluciona el modelo en el tiempo. Esta evolución definirá las versiones de la tabla de hechos, y los distintos esquemas de dimensiones con los que se trabajará e los experimentos.

Versión 1

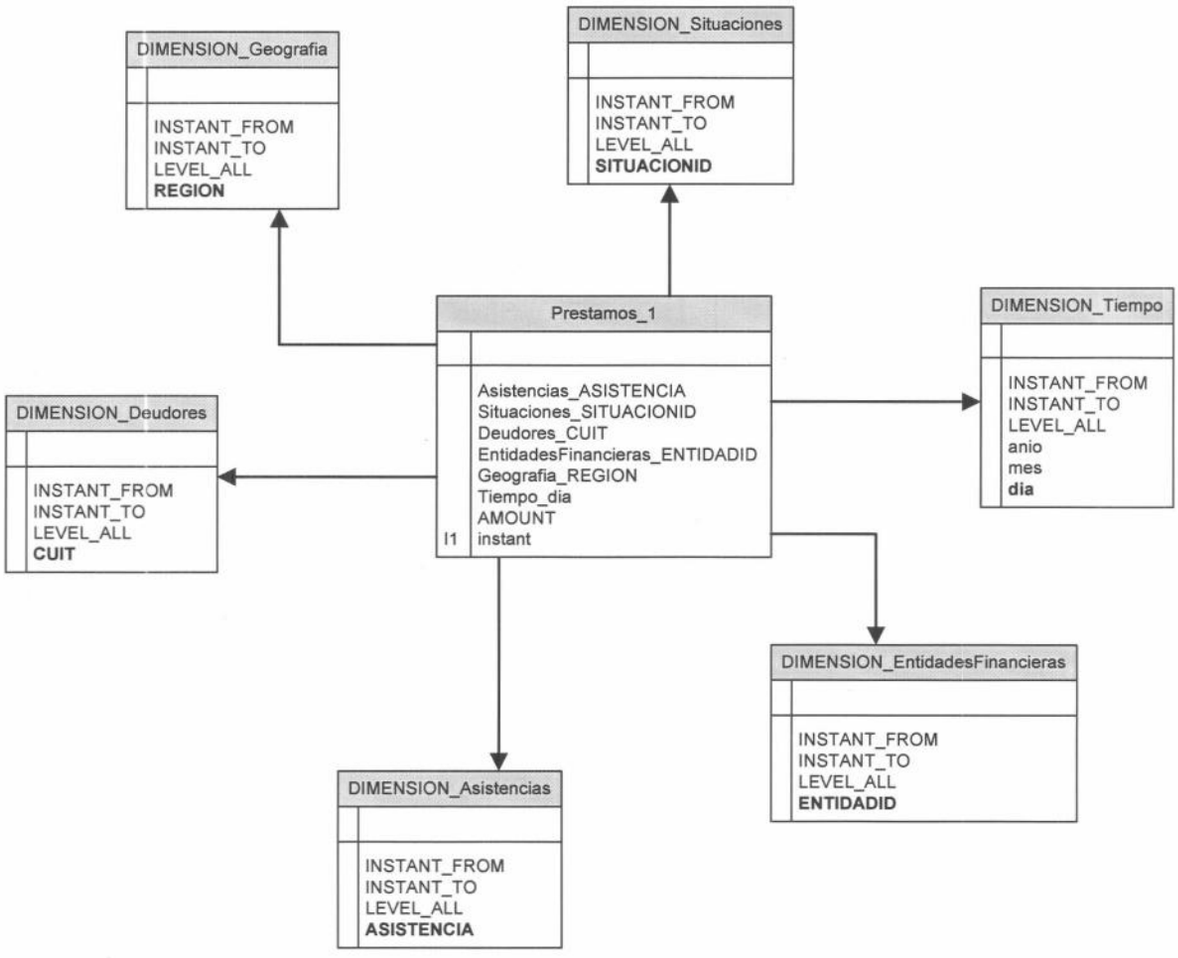


Figura 8.19. Modelo Central de Deudores (versión 1)

Versión 2

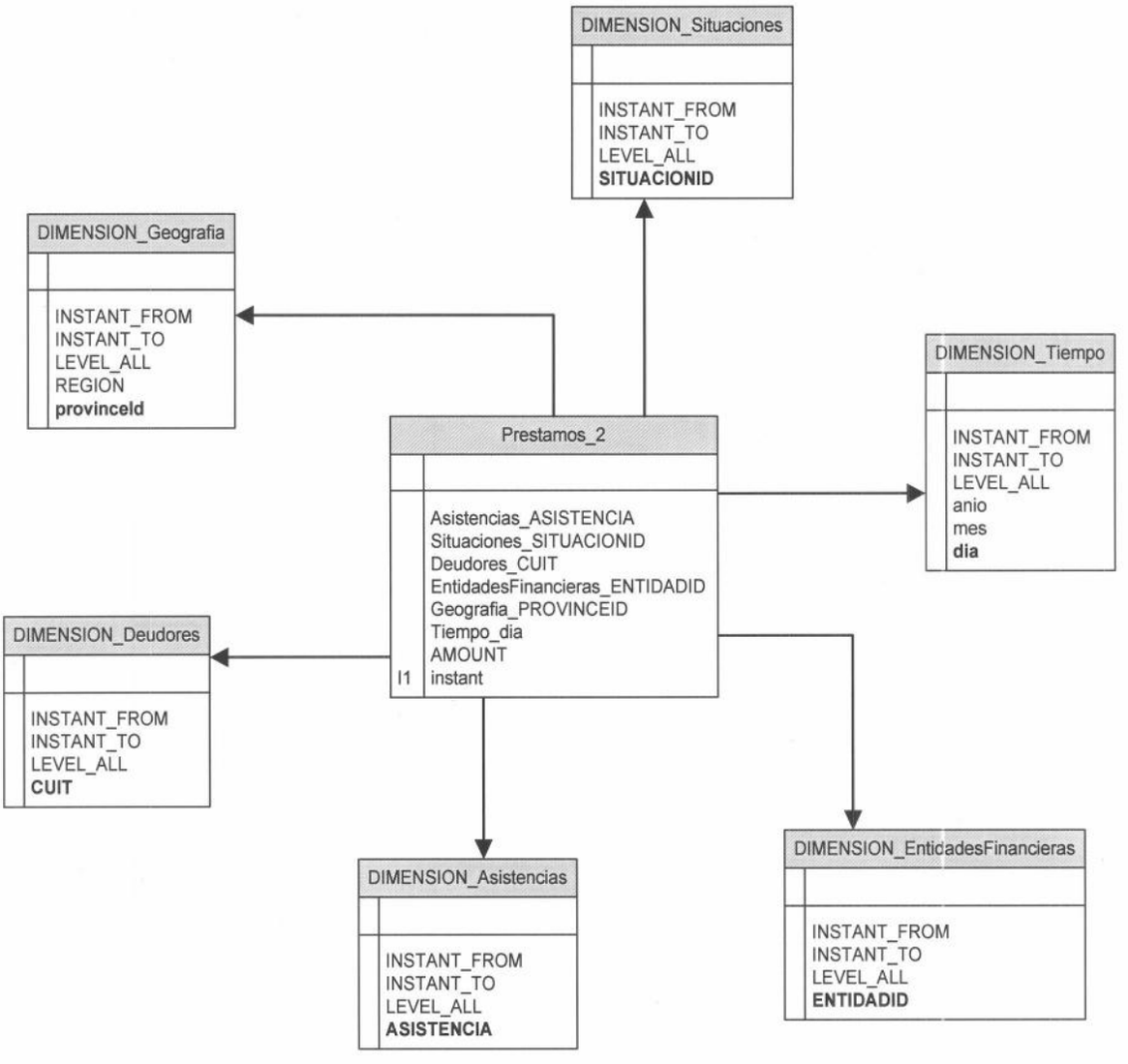


Figura 8.20. Modelo Central de Deudores (versión 2)

Versión 3

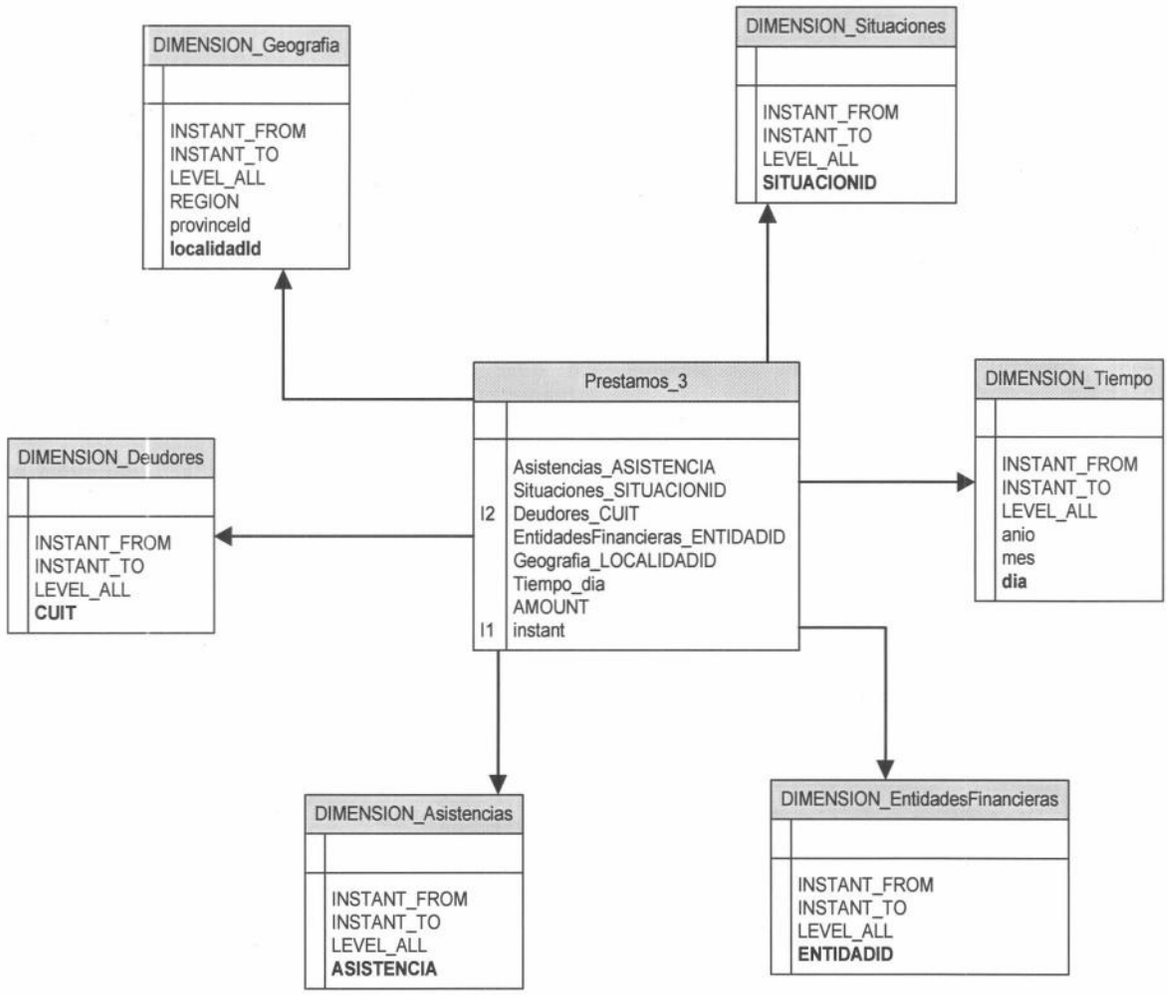


Figura 8.21. Modelo Central de Deudores (versión 3)

8.5 Resumen

En este capítulo presentamos un caso de estudio de la vida real. Contamos la complejidad que el mismo tiene y como se beneficia de la aplicación del modelo TOLAP. Por último se mostro el modelo de datos y los cambios que sufrió en el tiempo.

Capítulo 9.

EXPERIMENTACION

9.1 Objetivo de los experimentos

En el capítulo 5 se detallaron las modificaciones introducidas en esta implementación, a los efectos de mejorar la performance del sistema. En este capítulo, intentaremos corroborar si dichas optimizaciones producen o no una mejora en los tiempos de respuesta, para lo cual procederemos a ejecutar una serie de consultas representativas haciendo uso de la optimización implementada y sin uso de la misma. Asimismo, nuestros experimentos intentarán demostrar que los tiempos de respuesta a consultas típicas son compatibles con las necesidades de los usuarios.

9.2 Entorno de los experimentos

9.2.1 Plataforma de Hardware

Para llevar a cabo la experimentación planteada anteriormente se utilizó una PC con las características detalladas a continuación:

Motherboard : MSI – K8T NEO
Procesador : AMD Athlon 64 3000+ (2.0 Ghz)
Memoria RAM : 1 GB DDR-SDRAM (400 MHZ)
HD: WD250KS (Western Digital SATA de 250GB con 16 MB de buffer)
Conexión de Banda ancha (1 MB Download / 256 KB upload)

9.2.2 Plataforma de Software

En lo referente al software de base para las pruebas, las mismas fueron ejecutadas utilizando como servidor de base de datos la versión 2000 (SP4) de MS-SQL Server, corriendo sobre Windows XP Professional 5.01.2600 con SP2. En lo referente al servidor de procesos, se utilizo la versión 4.1.30 de TOMCAT.

9.2.3 Otras Consideraciones

Aunque el alojar el servidor de base de datos y el servidor de procesos en distintos equipos dedicados, optimizaría la performance general del sistema, en este experimento se alojaran ambos en el mismo equipo, por lo cual, los resultados obtenidos deben ser considerados como la cota inferior de performance.

9.2.4 Metodología de Ejecución de pruebas de Performance

Se procedió a seleccionar una serie de consultas sobre el caso de estudio presentado en el capítulo 8, para demostrar la versatilidad del sistema en lo referente al manejo de consultas temporales, así como también consultas de pruebas de la nueva gramática definida.

Estas pruebas fueron llevadas a cabo según los siguientes pasos:

1. Primero se desactivó la utilización del Optimizer (por medio de un flag implementado a los efectos).
2. Para cada consulta a evaluar
   - Se procedió a ejecutarla desde la herramienta Web quedando asentado en un log del sistema los resultados obtenidos de la consulta así como también sus tiempos.
   - Se reinició el Sistema (se reiniciaron los servicios del servidor de Base de Datos y de Aplicaciones para asegurarse de que no quede cacheado ningún dato que pudiese beneficiar a la próxima consulta).
3. Se procedió a activar la utilización del Optimizer y a llevar a cabo los pasos detallados en el paso 2.

9.2.5 Tipos de consultas a testear

De acuerdo al impacto en el modelo del caso de estudio estas consultas se encuentran divididas en:

a. Metaconsultas

Este tipo de consultas son resueltas sin la necesidad de llegar a utilizar las tablas de hechos, siendo solamente necesaria la utilización de los datos incluidos en la metadata del modelo.

b. Consultas con utilización de atributos de instancia

Debido a que en esta implementación los atributos de instancia no son atemporales, definir Vistas Materializadas para su posible elección por parte del Optimizer resulta inviable (debido a que no se encuentra asegurado un grado de agregación para las VMs), por lo cual el Optimizer solo puede aplicar como método de optimización el Query prunning y join optimization (en los casos que lo permitan).

c. Consultas sin utilización de atributos de instancia

Este tipo de consultas permite al optimizador realizar query rewriting (además de query prunning y join optimization) para aprovechar la existencia de Vistas Materializadas cuando resulte posible (en los casos en que la Vista materializada no existe pero el Optimizer determine que su utilización es obligatoria, debido al uso del algoritmo de generación automática de VMs, se procederá a su creación para su posterior utilización en la resolución de la consulta).

A continuación exhibiremos las consultas efectuadas para cada uno de los tipos descriptos. En el Anexo 3 se detallan para las consultas comprendidas en las pruebas, la traducción efectuada por el Parser con y sin el uso de optimización, y su resultado.

a. Metaconsultas.

Q1) Listar las localidades de la provincia de Buenos Aires.

En TOLAP-QL se escribe:

```
SELECT localidadId
 FROM Geografia G
 WHERE RUP(G.localidadId, provinceId:p, t)
       AND p.nombre(t) = 'Buenos Aires';
```

Q2) Indicar si existía una apertura geográfica por provincia para la fecha `'2001/01/03''`.

En TOLAP-QL se escribe:

```
SELECT boolean
 FROM Geografia G
 WHERE RUP(G, provinceId, '2001/01/03');
```

Q3) Indicar si existía una apertura geográfica por provincia para la fecha '2006/06/01'. Esta consulta es la misma que Q2, pero para otro instante, a los efectos de verificar que la evaluación lógica se realiza correctamente.

En TOLAP-QL se escribe:

```
SELECT boolean
 FROM Geografia G
 WHERE RUP(G, provinceId, '2006/06/01');
```

Q4) Listar las localidades y la zona de pertenencia de las mismas a la fecha.

En TOLAP-QL se escribe:

```
SELECT localidadId, X, x
 FROM Geografia G
 WHERE  RUP(G.localidadId, VAR X:VAR x ,NOW);
```

b. Consultas con utilización de atributos de instancia

Q5) Cantidad de préstamos brindados en la provincia de Buenos Aires.

En TOLAP-QL se escribe:

```
SELECT G.provinceId, COUNT(*)
FROM Prestamos P, Geografia G
WHERE P.Geografia = G.bottom
  AND RUP(G, provinceId:prov, P.t)
  AND prov.nombre(P.t) = 'Buenos Aires'
  AND P.amount > 0;
```

Q6) Para cada entidad que asiste a 'SA LA NACION' y a 'EDITORIAL EL ATLANTICO S A' indicar el saldo actual de su cartera de créditos.

En TOLAP-QL se escribe:

```
SELECT P.EntidadesFinancieras, SUM(P.amount)
FROM Prestamos P, Deudores D
WHERE P.Deudores = D.bottom
  AND RUP(D, CUIT:c, P.t) AND c.nombre = 'SA LA NACION'
  AND (RUP(D, CUIT:c1, P.t, c) AND
           c1.nombre = 'EDITORIAL EL ATLANTICO S A');
```

Q7) Clave Tributaria y deuda total en el Sistema Financiero de 'SA LA NACION'.

En TOLAP-QL se escribe:

```
SELECT D.CUIT, SUM(amount)
  FROM Prestamos P, Deudores D, Asistencias A
 WHERE P.Asistencias = A.bottom
   AND P.Deudores = D.bottom
   AND RUP(D, CUIT:c, P.t)
   AND RUP(A, Asistencia, P.t)
   AND c.nombre(P.t) = 'SA LA NACION'
```

Q8) Cantidad de Mujeres que poseen deudas hipotecarias en la provincia de Cordoba.

En TOLAP-QL se escribe:

```
SELECT D.CUIT, SUM(amount)
  FROM Prestamos P, Deudores D, Asistencias A, Geografia G
 WHERE P.Asistencias = A.bottom
   AND P.Deudores = D.bottom
   AND P.Geografia = G.bottom
   AND RUP(D, CUIT:c, P.t)
   AND c.sexo(P.t) = 'F'
   AND RUP(A, Asistencia:a, P.t)
   AND a.descripcion = 'Hipotecarios sobre la vivienda'
   AND RUP(G, provinceId:p, P.t)
   AND p.nombre(P.t) = 'Cordoba'
   AND P.amount > 0
STORE AS Q1;

SELECT COUNT(*)
  FROM Q1;
```

Q9) Listar la distribución de los prestamos por línea de asistencia en la provincia de Buenos Aires al 2004/11/01.

En TOLAP-QL se escribe:

```
SELECT A.Asistencia, SUM(amount)
  FROM Prestamos P, Deudores D, Asistencias A, Geografia G
 WHERE P.Deudores = D.bottom AND
       P.Asistencias = A.bottom AND
       P.Geografia = G.bottom AND
       RUP(A,Asistencia,P.t) AND
       RUP(D,CUIT,P.t) AND
       RUP(G,provinceId:p,P.t) AND
       p.nombre = 'Buenos Aires' AND
       P.t <= '2004-11-01';
```

Q10) Monto total de préstamos brindados en la provincia de Cordoba, en los meses en que en la provincia de Mendoza se prestó más de $200.000.

En TOLAP-QL se escribe:

```
SELECT J.mes AS mes, G.provinceId AS provinceId,
       SUM(amount) AS TotalProvince
  FROM Prestamos P, Geografia G, Tiempo J
 WHERE P.Geografia = G.bottom AND
       RUP(G,provinceId:p,P.t) AND
       P.Tiempo = J.bottom AND
       RUP(J,dia,P.t) AND
       p.nombre = 'Mendoza'
STORE AS Q1;
```

```
SELECT G.provinceId AS provinceId, SUM(amount) AS TotalProvince
  FROM Prestamos P, Geografia G, Q1, Tiempo J
 WHERE P.Geografia = G.bottom AND
       RUP(G,provinceId:p,P.t) AND
       P.Tiempo = J.bottom AND
       RUP(J,dia,P.t) AND
       p.nombre = 'Cordoba' AND
       P.amount > 0 AND
       Q1.TotalProvince > 200000 AND
       Q1.mes = J.mes;
```

Q11) Consulta: *Indicar la cantidad de personas que tomaron prestamos en Cordoba y Buenos Aires en el mismo mes.*

En TOLAP-QL se escribe:

```
SELECT J.mes AS mes,D.CUIT AS Deudor
  FROM Prestamos P, Geografia G, Tiempo J, Deudores D
 WHERE P.Geografia = G.bottom AND
       RUP(G,provinceId:pr,P.t) AND
       P.Tiempo = J.bottom AND
       RUP(J,dia,P.t) AND
       P.Deudores = D.bottom AND
       RUP(D,CUIT,P.t) AND
       pr.nombre = 'Cordoba' AND
       P.amount > 0
STORE AS Q1;

SELECT COUNT(*)
  FROM Prestamos P, Geografia G, Q1, Tiempo J, Deudores D
 WHERE P.Geografia = G.bottom AND
       RUP(G,provinceId:pr,P.t) AND
       P.Tiempo = J.bottom AND
       RUP(J,dia,P.t) AND
       P.Deudores = D.bottom AND
       RUP(D,CUIT,P.t) AND
       pr.nombre = 'Buenos Aires' AND
       P.amount > 0 AND
       Q1.Deudor = D.bottom AND
       Q1.mes = J.mes;
```

Q12) Para cada mes, listar la cantidad de personas que tienen deudas en la provincia de Buenos Aires y Cordoba.

En TOLAP-QL se escribe:

```
SELECT J.mes AS mes, P.Deudores
FROM Prestamos P, Geografia G , Tiempo J
WHERE P.Tiempo = J.bottom
  AND RUP(J,dia,P.t)
  AND P.Geografia = G.bottom
  AND RUP(G, provinceId:pr, P.t) AND pr.nombre = 'Cordoba'
  AND (RUP(G,  provinceId:pr1, P.t) AND
                              pr1.nombre  = 'Buenos Aires')
STORE AS Q1 ;

SELECT Q1.mes, COUNT(*)
  FROM Q1;
```

Q13) Distribución de deuda por clasificación del deudor para las provincias de la región de Cuyo que han brindado prestamos por más de $200.000.

En TOLAP-QL se escribe:

```
SELECT G.provinceId AS provinceId, SUM(P.amount) AS TotalProv
  FROM Prestamos P, Geografia G
 WHERE P.Geografia = G.bottom
   AND RUP(G, Region:r, P.t)
   AND r.nombre(P.t) = 'Cuyo'
   AND P.amount > 0
STORE AS Q1;
SELECT P.Situaciones, SUM(P.amount)
  FROM Prestamos P, Geografia G, Q1
 WHERE P.Geografia = G.bottom
   AND RUP(G, provinceId:p, P.t)
   AND p = Q1.provinceId
   AND Q1.TotalProv > 200000
   AND P.amount > 0;
```

Q14) Para cada mes, listar los deudores que han recibido más de un préstamo de un mismo banco y que además fueron asistidos por más de un banco.

En TOLAP-QL se escribe:

```
SELECT J.mes AS mes, D.CUIT AS CUIT,  E.EntidadId ,
       COUNT(*) AS CantPrestamos
  FROM Prestamos P, Deudores D,  Tiempo J, EntidadesFinancieras E
 WHERE P.Deudores = D.bottom AND
       RUP(D,CUIT,P.t) AND
       P.Tiempo = J.bottom AND
       RUP(J,dia,P.t) AND
       P.EntidadesFinancieras = E.bottom AND
       RUP(E,EntidadId,P.t) AND
       P.amount > 0
STORE AS Q1;

SELECT Q1.mes AS mes, Q1.CUIT AS CUIT, COUNT(*) AS CantPrestadoras
  FROM Q1
 WHERE Q1.CantPrestamos > 1
STORE AS Q2;

SELECT Q2.mes, Q2.CUIT
  FROM Q2
 WHERE Q2.CantPrestadoras > 1 ;
```

c. Consultas sin utilización de atributos de instancia

Q15) Monto total brindado para la asistencia de tipo 5 por cada Entidad Financiera.

En TOLAP-QL se escribe:

```
SELECT E.EntidadId,  SUM(amount)
FROM Prestamos P, EntidadesFinancieras E, Asistencias A
WHERE P.EntidadesFinancieras = E.bottom AND
      P.Asistencias = A.bottom AND
      RUP(A, Asistencia:'5' ,P.t ) AND
      RUP(E, EntidadId, P.t );
```

Q16) Cantidad de deudores que en el año 2004 disminuyeron su deuda respecto del año 2003.

En TOLAP-QL se escribe:

```
SELECT D.CUIT AS CUIT, A.Asistencia AS Asistencia,
       SUM(amount) AS TotalAsistencia
  FROM Prestamos P, Asistencias A, Deudores D
 WHERE P.Asistencias = A.bottom
   AND P.Deudores = D.bottom
   AND RUP(D, CUIT, P.t)
   AND RUP(A, Asistencia, P.t)
   AND P.t >= '2003/01/01 00:00:01' AND P.t < '2004/01/01'
STORE AS Q2003;

SELECT D.CUIT AS CUIT, A.Asistencia AS Asistencia,
       SUM(amount) AS TotalAsistencia
  FROM Prestamos P, Asistencias A, Deudores D
 WHERE P.Asistencias = A.bottom
   AND P.Deudores = D.bottom
   AND RUP(D, CUIT, P.t)
   AND RUP(A, Asistencia, P.t)
   AND P.t >= '2004/01/01 00:00:01' AND P.t < '2005/01/01'
STORE AS Q2004;

SELECT Q2004.CUIT
  FROM Q2003, Q2004
 WHERE Q2003.CUIT = Q2004.CUIT
   AND Q2003.Asistencia = Q2004.Asistencia
   AND Q2003.TotalAsistencia > Q2004.TotalAsistencia
STORE AS QDeudores;

SELECT COUNT(*)
  FROM QDeudores;
```

9.3 Pruebas de Performance

9.3.1 Resultados

El siguiente cuadro muestra los tiempos obtenidos para cada una de las consultas de la sección 9.2

Consulta	Tipo de Consulta	Sin optimización	Con optimización	Tipo de optimización
Q1	A	00:00:00,375	---	---
Q2	A	00:00:00,063	---	---
Q3	A	00:00:00,016	---	---
Q4	A	00:00:00,359	---	---
Q5	B	---	01:16:47,765	1
Q6	B	00:12:58,453	---	---
Q7	B	00:26:51,579	---	---
Q8	B	00:44:58,390	---	---
Q9	B	---	00:03:59,391	1,3
Q10	B	---	05:16:03,844	1
Q11	B	---	01:40:14,063	1
Q12	B	---	64:13:20,453	1
Q13	B	02:49:18,469	02:15:44,625	1,4
Q14	B	02:16:49,297	---	---
Q15	C	00:04:24,406	00:02:00,403	2,4
Q16	C	02:24:15,187	00:03:11,110	3,4

Donde,

A: Metaconsultas.
B: Consultas con utilización de atributos de instancia.
C: Consultas sin utilización de atributos de instancia.
1: Descarte de tabla de hechos.
2: Optimización de joins.
3: Query prunning.
4: Vistas materializadas.

9.3.2 Análisis de los Resultados

De la tabla de la sección 9.3.1, no nos es posible evaluar la ganancia de tiempos debido a la aplicación de la optimización por descarte de tabla de hechos debido a que la implementación efectuada en el traductor *no soporta no aplicar* este tipo de optimización (motivo por el cual no fue posible desactivar esta característica para las pruebas).

Para las pruebas que contemplan optimización por medio de vistas materializadas, se procedió a materializar las vistas detalladas en el Anexo 4 a los efectos de comprobar la selectividad del optimizador así como también medir los beneficios de su aplicación, obteniéndose los siguientes resultados:

Consulta	Sin optimización	Con optimización	Tiempo de creación de la(s) Vm(s) participante(s)	Ganancia
Q13	02:49:18,469	02:15:44,625	00:02:13.719	24,73%
Q15	00:04: 24,406	00:02:00,403	00:04:15,609	119,60%
Q16	02:24:15,187	00:03:11,110	00:29:02,750	4528,90%

En donde puede apreciarse que en todos los casos se obtuvo una ganacia considerable. Nótese que la menor ganacia la obtuvo Q13, en donde la vista materializada es aplicada en la segunda consulta de un programa TOLAP y es debido al primer paso que la mejora de tiempos no es más crucial.

9.4 Resumen

En este capítulo describimos la plataforma de hardware y software utilizada en los experimentos y como los llevamos a cabo. Posteriormente, mostramos los tipos de consultas testeadas y como las mismas se escriben en TOLAP-QL. Por último, mostramos un cuadro con los tiempos obtenidos con y sin el uso de optimización. Posteriormente mostramos otro cuadro con el impacto del uso de vistas materializadas.

Capítulo 10.

CONCLUSIONES

En esta tesis presentamos una herramienta que permite realizar consultas OLAP Temporales en un entorno Web. La herramienta soporta también la aplicación de operadores de actualización de dimensiones, manteniendo la historia de las dimensiones a lo largo del tiempo.

La independización del motor de base de datos, posibilita la utilización de este sistema con cualquier Warehouse pre-existente (con la debida adecuación de tipos de datos y el adecuado upload a un modelo TOLAP).

Así mismo, se ha definido una gramática estilo SQL. La gramática TOLAP-QL permite al usuario utilizar consultas mucho más cercanas a los lenguajes de consultas de base datos relacionales.

La implementación soporta la utilización de atributos temporales, es decir, no solo se mantiene la historia estructural de los esquemas de dimensiones, sino que también mantenemos historia de los atributos de cada miembro de las dimensiones.

En esta tesis se estudiaron e implementaron diversas técnicas de optimización de consultas OLAP Temporales. Estas técnicas incluyen definición manual y automática de vistas materializadas.

Los experimentos mostraron una mejora significativa en los consultas en que se aplicaban algún tipo de optimización, en especial las que utilizaban vistas materializadas.

En cuanto a las mejoras en los tiempos de obtención de resultados, los mismos han evidenciado mejoras de más del 100% en los casos en que las VMs utilizadas poseían una baja cardinalidad (incluso obtuvimos en Q16 un mejora superior al 4500%), sin embargo, en casos de alta cardinalidad en las vistas materializadas se siguió obteniendo mejoras aunque las mismas solo implicaron mejoras cercanas al 25%.

Finalmente, aplicamos nuestro modelo a un caso de estudio real, el cual posee la característica de ser un sistema de tipo VLDB (Very Large Data Base), en el cual se disponía de más de 165.000.000 de registros en la tabla de hecho y una de sus dimensiones, la de Deudores, mantenia una cardinalidad superior a los 6.000.000 por lo cual todas las consultas complejas que realizaban joins entre estas tablas resultan de utilización exaustiva del equipo con altos tiempos de respuesta. Sin embargo, como hemos mostrado en el capitulo 9, la utilización de vistas materializadas logra significativas mejoras al modelo actual.

ANEXO 1: XML SCHEMA DOCUMENTS VALIDATION

Validación del catalogo de dimensiones

A continuación se transcribe el contenido del archivo que define el esquema de validación utilizado para la metadata correspondiente a los esquemas de dimensiones existentes.

dimSchemas.xsd

```
<schema targetNamespace="http://www.example.com/DimSchemas"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:s="http://www.example.com/DimSchemas"
        elementFormDefault="qualified">

   <element name="dimSchemas">
      <complexType>
         <sequence>
            <element name="dimensionSchemas" type="s:schemaType" minOccurs="1"/>
         </sequence>
      </complexType>
   </element>


   <complexType name="schemaType">
      <complexType>
         <element name="dimension" type="s:dimensionType" minOccurs="0"/
maxOccurs="unbounded"/>
      </complexType>
   </complexType>

   <complexType name="dimensionType">
      <attribute name="name" type="string"/>
   </complexType>

</schema>
```

Validación de MetaFacTables

A continuación se presenta el esquema de validación utilizado para la metadata correspondiente a las tablas de hechos.

MetaFactables.xsd

```
<root targetNamespace="http://www.example.com/MetaFactTables"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:m="http://www.example.com/MetaFactTables"
        elementFormDefault="qualified">

   <element name="metaFactTables">
      <complexType>
         <sequence>
            <element name="metaFactTable" type="m:metaFactTableType" minOccurs="0"/>
         </sequence>
      </complexType>
   </element>


   <complexType name="metaFactTableType">
```

```
      <sequence>
         <element name="versions" type="m:versionType" minOccurs="0"
                                        maxOccurs="unbounded"/>
      </sequence>
      <attribute name="name" type="string"/>
      <attribute name="granularity">
         <simpleType>
            <restriction base="string">
               <annotation>
                  <documentation>Y Year - M Month - D Date - h Hour - m Minute
                                                                  - s Second
                  </documentation>
               </annotation>
               <enumeration value="Y"/>
               <enumeration value="M"/>
               <enumeration value="D"/>
               <enumeration value="h"/>
               <enumeration value="m"/>
               <enumeration value="s"/>
            </restriction>
         </simpleType>
      </attribute>
      <attribute name="type">
         <simpleType>
            <annotation>
               <documentation>A Additive - S Semi-Additive</documentation>
            </annotation>
            <restriction base="string">
               <enumeration value="A"/>
               <enumeration value="S"/>
            </restriction>
         </simpleType>
      </attribute>
      <attribute name="measure" type="string"/>
   </complexType>


   <complexType name="versionType">
      <sequence>
         <element name="version" minOccurs="0" maxOccurs="unbounded">
            <complexType>
               <sequence>
                  <element name="levels" type="m:levelsType" minOccurs="0"/>
               </sequence>
               <attribute name="number" type="int"/>
               <attribute name="instantFrom" type="timeInstant"/>
               <attribute name="instantTo" type="timeInstant" use="optional"/>
            </complexType>
         </element>
      </sequence>
   </complexType>


   <complexType name="levelsType">
      <sequence>
         <element name="level" minOccurs="0" maxOccurs="unbounded">
            <complexType>
                <attribute name="dimension" type="string"/>
                <attribute name="name" type="string"/>
            </complexType>
         </element>
      </sequence>
   </complexType>

</root>
```

Validación de dimensiones

A continuación se transcribe el contenido del archivo que define el esquema de validación utilizado para la metadata correspondiente al contenido de cada dimensión.

Dimension.xsd

```
<schema targetNamespace="http://www.example.com/Dimension"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:d="http://www.example.com/Dimension"
        elementFormDefault="qualified">

   <element name="metadata">
      <complexType>
         <sequence>
            <element name="dimension" type="d:dimensionType" minOccurs="0"/>
         </sequence>
      </complexType>
   </element>


   <complexType name="dimensionType">
      <sequence>
         <element name="levels" type="d:levelsType" minOccurs="1"/>
         <element name="rollup" type="d:rollupType" minOccurs="1"/>
      </sequence>
      <attribute name="name" type="string"/>
      <attribute name="description" type="string"/>
      <attribute name="granularity">
         <simpleType>
            <restriction base="string">
               <annotation>
                  <documentation>Y Year - M Month - D Date - h Hour - m Minute
                                                              - s Second
                  </documentation>
               </annotation>
               <enumeration value="Y"/>
               <enumeration value="M"/>
               <enumeration value="D"/>
               <enumeration value="h"/>
               <enumeration value="m"/>
               <enumeration value="s"/>
            </restriction>
         </simpleType>
      </attribute>
      <attribute name="modo">
         <simpleType>
            <restriction base="string">
               <enumeration value="I"/>
               <enumeration value="N"/>
            </restriction>
         </simpleType>
      </attribute>
      <attribute name="tipo">
         <simpleType>
            <restriction base="string">
               <enumeration value="T"/>
               <enumeration value="G"/>
            </restriction>
         </simpleType>
      </attribute>
      <attribute name="instantFrom" type="timeInstant"/>
      <attribute name="instantTo" type="timeInstant" use="optional"/>
   </complexType>
```

```
<complexType name="intervalType">
   <sequence>
      <element name="interval" minOccurs="0" maxOccurs="unbounded">
         <complexType>
            <attribute name="instantFrom" type="timeInstant"/>
            <attribute name="instantTo" type="timeInstant" use="optional"/>
         </complexType>
      </element>
   </sequence>
</complexType>


<complexType name="attributeIntervalType">
   <sequence>
      <element name="attributeInterval" minOccurs="1" maxOccurs="unbounded">
         <complexType>
            <attribute name="instantFrom" type="timeInstant"/>
            <attribute name="instantTo" type="timeInstant" use="optional"/>
         </complexType>
      </element>
   </sequence>
</complexType>


<complexType name="attributeType">
   <sequence>
      <element name="attribute" minOccurs="1" maxOccurs="unbounded">
         <complexType>
            <sequence>
               <element name="attributeIntervals" type="d:attributeIntervalType"
                                          minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="name" type="string"/>
            <attribute name="ttype">
               <simpleType>
                  <restriction base="string">
                     <annotation>
                        <documentation>I Integer - F Float - D Double - S String
                                                    - T Time Instant (datetime)
                        </documentation>
                     </annotation>
                     <enumeration value="I"/>
                     <enumeration value="F"/>
                     <enumeration value="D"/>
                     <enumeration value="S"/>
                     <enumeration value="T"/>
                  </restriction>
               </simpleType>
            </attribute>
         </complexType>
      </element>
   </sequence>
</complexType>


<complexType name="levelsType">
   <sequence>
      <element name="level" minOccurs="0" maxOccurs="unbounded">
         <complexType>
            <sequence>
               <element name="attributes" type="d:attributeType" minOccurs="0"
                                                    maxOccurs="unbounded"/>
               <element name="intervals" type="d:intervalType" minOccurs="0"
                                                   maxOccurs="unbounded"/>
            </sequence>
            <attribute name="name" type="string"/>
            <attribute name="ttype">
```

```
                <simpleType>
                   <restriction base="string">
                      <annotation>
                         <documentation>I Integer - F Float - D Double - S String
                                                          - T Time Instant (datetime)
                         </documentation>
                      </annotation>
                      <enumeration value="I"/>
                      <enumeration value="F"/>
                      <enumeration value="D"/>
                      <enumeration value="S"/>
                      <enumeration value="T"/>
                   </restriction>
                </simpleType>
             </attribute>
          </complexType>
       </element>
    </sequence>
 </complexType>


 <complexType name="rollupType">
    <sequence>
       <element name="levelFrom" minOccurs="0" maxOccurs="unbounded">
          <complexType>
             <sequence>
                <element name="levelTo" minOccurs="0" maxOccurs="unbounded">
                   <complexType>
                      <sequence>
                         <element name="intervals" type="d:intervalType"
                                                   minOccurs="0" maxOccurs="unbounded"/>
                      </sequence>
                      <attribute name="name" type="string"/>
                   </complexType>
                </element>
             </sequence>
             <attribute name="name" type="string"/>
          </complexType>
       </element>
    </sequence>
 </complexType>

</schema>
```

ANEXO 2: DEFINICIÓN DE TOLAP SOAP SERVICE

A continuación se transcribe la definición realizada en el servidor TOMCAT de los servicios Web implementados.

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
id="urn:Tolap">
  <isd:provider type="java"
                scope="Application"
                methods="echo tAddInstance tDelInstance tReclassify tSplit
tMerge tUpdate tGeneralize tDelLevel tRelate tUnrelate tSpecialize
freeElements getParameter getLastUpdate isReadEnabled isLockEnabled
isWriteEnabled read lock write existsDimension getDimension addDimension
getDimensionList getLevel getLevels getBottom getAttribute getAttributes
createAttribute getSnapshot existsRollup getStructureChanges getAinstset
getInstanceAttributes delInstanceAttribute addInstanceAttribute
getAttributeChanges getAttributeValueChanges getRollUpInstance
getInstanceChanges getIdFile getInstanceAttributes closeIntervalAttribute
closeIntervalInstanceAttribute addInstanceAttribute getNow getOperatorsHistory
getInstanceAttributesChanges getMetaFactTables getMetaFactTable
getMetaFactTableVersions getMetaFactTableVersion getMetaFactTableLevels
createMetaFactTable dropMetaFactTable deleteDimension findInstance
findInstanceTo findInstancesToNotRolledup getVMTables dropVM getVMLevels
createVM getLevelsCombinations">
    <isd:java class="tolap.messenger.Messenger" static="false"/>
  </isd:provider>


<isd:faultListener>org.apache.soap.server.DOMFaultListener</isd:faultListener>
  <isd:mappings>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:x="urn:xml-soap-dimensionstructure"
qname="x:dimensionstructure"
             javaType="tolap.structures.DimensionStructure"
             java2XMLClassName="tolap.soap.DimensionStructureSerializer"
             xml2JavaClassName="tolap.soap.DimensionStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:x="urn:xml-soap-snapshotstructure"
qname="x:snapshotstructure"
             javaType="tolap.structures.SnapshotStructure"
             java2XMLClassName="tolap.soap.SnapshotStructureSerializer"
             xml2JavaClassName="tolap.soap.SnapshotStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:x="urn:xml-soap-timeinstantstructure"
qname="x:timeinstantstructure"
             javaType="tolap.structures.TimeInstantStructure"
             java2XMLClassName="tolap.soap.TimeInstantStructureSerializer"
             xml2JavaClassName="tolap.soap.TimeInstantStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:x="urn:xml-soap-intervalstructure"
qname="x:intervalstructure"
             javaType="tolap.structures.IntervalStructure"
             java2XMLClassName="tolap.soap.IntervalStructureSerializer"
             xml2JavaClassName="tolap.soap.IntervalStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:x="urn:xml-soap-levelstructure" qname="x:levelstructure"
             javaType="tolap.structures.LevelStructure"
             java2XMLClassName="tolap.soap.LevelStructureSerializer"
```

```
            xml2JavaClassName="tolap.soap.LevelStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:x="urn:xml-soap-rollupstructure" qname="x:rollupstructure"
            javaType="tolap.structures.RollupStructure"
            java2XMLClassName="tolap.soap.RollupStructureSerializer"
            xml2JavaClassName="tolap.soap.RollupStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:x="urn:xml-soap-attributestructure"
qname="x:attributestructure"
            javaType="tolap.structures.AttributeStructure"
            java2XMLClassName="tolap.soap.AttributeStructureSerializer"
            xml2JavaClassName="tolap.soap.AttributeStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:x="urn:xml-soap-apperrorstructure"
qname="x:apperrorstructure"
            javaType="tolap.structures.AppErrorStructure"
            java2XMLClassName="tolap.soap.AppErrorStructureSerializer"
            xml2JavaClassName="tolap.soap.AppErrorStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:x="urn:xml-soap-instancestructure"
qname="x:instancestructure"
            javaType="tolap.structures.InstanceStructure"
            java2XMLClassName="tolap.soap.InstanceStructureSerializer"
            xml2JavaClassName="tolap.soap.InstanceStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:x="urn:xml-soap-instanceattributestructure"
qname="x:instanceattributestructure"
            javaType="tolap.structures.InstanceAttributeStructure"

java2XMLClassName="tolap.soap.InstanceAttributeStructureSerializer"

xml2JavaClassName="tolap.soap.InstanceAttributeStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:x="urn:xml-soap-operatorstructure"
qname="x:operatorstructure"
            javaType="tolap.structures.OperatorStructure"
            java2XMLClassName="tolap.soap.OperatorStructureSerializer"
            xml2JavaClassName="tolap.soap.OperatorStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:x="urn:xml-soap-metafacttablestructure"
qname="x:metafacttablestructure"
            javaType="tolap.structures.MetaFactTableStructure"
            java2XMLClassName="tolap.soap.MetaFactTableStructureSerializer"
            xml2JavaClassName="tolap.soap.MetaFactTableStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:x="urn:xml-soap-metafacttableversionstructure"
qname="x:metafacttableversionstructure"
            javaType="tolap.structures.MetaFactTableVersionStructure"

java2XMLClassName="tolap.soap.MetaFactTableVersionStructureSerializer"

xml2JavaClassName="tolap.soap.MetaFactTableVersionStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:x="urn:xml-soap-metafacttablelevelstructure"
qname="x:metafacttablelevelstructure"
            javaType="tolap.structures.MetaFactTableLevelStructure"

java2XMLClassName="tolap.soap.MetaFactTableLevelStructureSerializer"

xml2JavaClassName="tolap.soap.MetaFactTableLevelStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
            xmlns:x="urn:xml-soap-materializedviewstructure"
qname="x:materializedviewstructure"
            javaType="tolap.structures.MaterializedViewStructure"

java2XMLClassName="tolap.soap.MaterializedViewStructureSerializer"

xml2JavaClassName="tolap.soap.MaterializedViewStructureSerializer"/>
    <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:x="urn:xml-soap-materializedviewlevelstructure"
qname="x:materializedviewlevelstructure"
            javaType="tolap.structures.MaterializedViewLevelStructure"

java2XMLClassName="tolap.soap.MaterializedViewLevelStructureSerializer"

xml2JavaClassName="tolap.soap.MaterializedViewLevelStructureSerializer"/>
  </isd:mappings>
</isd:service>
```

ANEXO 3: ESPECIFICACIÓN DE TOLAP-QL EN JAVA CUP

A continuación se transcribe el archivo brindado como entrada al JAVA CUP para generar el parser de TOLAP-QL.
Nótese, que aquí se encuentra la definición de la gramatica en forma BNF para TOLAP-QL

```
/*************************************************************************/
/*              Especificacion JAVA_CUP del parser de TOLAP-QL           */
/*************************************************************************/

import java_cup.runtime.Symbol;

/************************ Para uso del Scanner. ****************************/
parser code     {:
    private Scanner scanner;
    public void debug_reduce(int prod_num, int nt_num, int rhs_size) {
        debug_message("# Reduce '" + Token.productionAsString(prod_num) + "'");
    }
    public void debug_shift(Symbol shift_tkn) {
        debug_message("# Shift '" + Token.terminalName(shift_tkn.sym) +
                      "' to state #" + shift_tkn.parse_state);
    }
    public void debug_current(Symbol curr_tkn) {
        if (curr_tkn == null)
            debug_message("# Current symbol is NULL");
        else
            debug_message("# Current symbol is '" +
                          Token.terminalName(curr_tkn.sym) + "'");
        }
:};

init with       {: scanner = new Scanner(); scanner.init();         :};
scan with       {: return Status.reportSymbol(scanner.nextToken()); :};

/****************************** TERMINALES *********************************/
// PALABRAS RESERVADAS
terminal            NULL, SELECT, FROM, WHERE;
terminal            COUNT, SUM, AVG, MAX, MIN;
terminal            NOT, AND, OR, AS;
terminal            VAR;
terminal            RUP;
terminal            STORE;

// SIGNOS DE PUNTUACION
terminal            PYC, PUNTO, COMA, COMILLA, DOSPUNTOS;
terminal            EQ, NE, NEQ, LT, GT, LET, GET;
terminal            MAS, MENOS, POR, DIV;
terminal            LPAR, RPAR;

// IDENTIFICADORES Y CONSTANTES
terminal            ID;
terminal Integer    CONST_INT;
terminal String     CONST_FLOAT;
terminal String     CONST_STR;
terminal String     CONST_DATE;

terminal            BOOLEAN;

/***************************** NO TERMINALES *******************************/
non terminal TolapProgram               tolap_ql_program;
non terminal TolapProgram               tolap_ql_program_list;
non terminal Sentence                   tolap_ql_sentence;
non terminal Select                     select_statement;
non terminal List                       head;
non terminal SelectBody                 body;
non terminal List                       from_clause;
non terminal From                       from_object;
non terminal Where                      where_clause;
non terminal Store                      store_clause;
non terminal IdentifierLevel            identifier_level;
```

```
non terminal IdentifierAttribute        identifier_attribute;
non terminal Atom                       atom;
non terminal Aggregate                  aggregate_atom;
non terminal IdentifierHead             identifier_head;
non terminal Condition                  conditional_or_expression;
non terminal Condition                  conditional_and_expression;
non terminal Expression                 equality_expression;
non terminal Expression                 relational_expression;
non terminal Expression                 unary_expression;
non terminal Expression                 additive_expression;
non terminal Expression                 multiplicative_expression;
non terminal RupExpression              rup_expression;
non terminal RupExpression              rup_condition;
non terminal RupIdentifier              identifier_rup;
non terminal RupIdentifier              identifier_rup_from;
non terminal RupIdentifier              identifier_rup_to;

// PRECEDENCIAS
precedence left     OR;
precedence left     AND;
precedence left     NOT;
precedence nonassoc EQ, NE, NEQ, LET, LT, GET, GT;
precedence left     MAS, MENOS;
precedence left     POR, DIV;
precedence left     PUNTO;

/*************************************************************************************/
/*                              Gramatica del lenguaje                               */
/*************************************************************************************/

start with tolap_ql_program;

tolap_ql_program ::= tolap_ql_program_list:program
{:RESULT = program;:}
        ;

//-------------------------------------------------------------------------------------

tolap_ql_program_list ::= tolap_ql_program_list:program tolap_ql_sentence:sentencia
{:RESULT = program.addProgram(sentencia);:}
        |   /* empty */
{:RESULT = new TolapProgram();:}
        ;

//-------------------------------------------------------------------------------------

tolap_ql_sentence
        ::=  select_statement:select
{:RESULT = select;:}
         ;

//-------------------------------------------------------------------------------------

select_statement
        ::=  SELECT body:cuerpo PYC
{:RESULT = new SelectHeadNull("SELECT", cuerpo);:}
        |
             SELECT head:cabecera body:cuerpo PYC
{:RESULT = new Select("SELECT", cabecera, cuerpo);:}
        |
             SELECT head:cabecera body:cuerpo store_clause:almacena PYC
{:RESULT = new Select("SELECT", cabecera, cuerpo, almacena);:}
        |
             SELECT BOOLEAN body:cuerpo PYC
{:RESULT = new SelectBoolean("SELECT", cuerpo);:}
        ;

//-------------------------------------------------------------------------------------

head ::= atom:atomo
{:RESULT = new SelectHead().add(atomo);:}
        |
           head:cabecera COMA atom:atomo
{:RESULT = cabecera.add(atomo);:}
        ;

//-------------------------------------------------------------------------------------
```

```
atom
        ::= aggregate_atom:agg_atom
{:RESULT = agg_atom;:}
        |
            aggregate_atom:agg_atom AS ID:alias
{:RESULT = new Atom("ATOM_AGGREGATE", agg_atom, new Id(alias));:}
        |
            identifier_head:id_head
{:RESULT = new Atom("ATOM", id_head);:}
        |
            identifier_head:id_head AS ID:alias
{:RESULT = new Atom("ATOM_ALIAS", id_head, new Id(alias));:}
        ;

//-------------------------------------------------------------------------------------

identifier_head ::=
          ID:levelId
{:RESULT = new IdentifierHead("HEAD_LEVEL", new Id(levelId));:}
        |
          ID:dimensionId PUNTO ID:levelId
{:RESULT = new IdentifierHead("HEAD_DIM_LEVEL", new Id(dimensionId), new Id(levelId));:}
        ;

//-------------------------------------------------------------------------------------

identifier_level
        ::= ID:levelId
{:RESULT = new IdentifierLevel("ID_LEVEL", new Id(levelId));:}
        |
          ID:dimensionId PUNTO ID:levelId
{:RESULT = new IdentifierLevel("ID_DIM_LEVEL", new Id(dimensionId), new Id(levelId));:}
        ;

//-------------------------------------------------------------------------------------

aggregate_atom ::=
    SUM LPAR identifier_level:id_level RPAR
{:RESULT = new Aggregate("SUM", id_level);:}
        |
    COUNT LPAR identifier_level:id_level RPAR
{:RESULT = new Aggregate("COUNT", id_level);:}
        |
    COUNT LPAR POR RPAR
{:RESULT = new Aggregate("COUNT", new Id("*"));:}
        |
    AVG LPAR identifier_level:id_level RPAR
{:RESULT = new Aggregate("AVG", id_level);:}
        |
    MAX LPAR identifier_level:id_level RPAR
{:RESULT = new Aggregate("MAX", id_level);:}
        |
    MIN LPAR identifier_level:id_level RPAR
{:RESULT = new Aggregate("MIN", id_level);:}
        ;

//-------------------------------------------------------------------------------------

body ::=
          FROM from_clause:fclause
{:RESULT = new SelectBody("SELECTBODY",fclause);:}
       |  FROM from_clause:fclause where_clause:wclause
{:RESULT = new SelectBody("SELECTBODY",fclause,wclause);:}
    ;

//-------------------------------------------------------------------------------------

from_clause
        ::= from_object:fobject
{:RESULT = new List().add(fobject);:}
        |
          from_clause:fclause COMA from_object:fobject
{:RESULT = fclause.add(fobject);:}
        ;

//-------------------------------------------------------------------------------------
```

```
from_object
        ::= ID:id
{:RESULT = new From("FROM", new Id(id));:}
        |
       ID:id ID:id_alias
{:RESULT = new From("FROM_ALIAS", new Id(id), new Id(id_alias));:}
        |
       ID:id AS ID:id_alias
{:RESULT = new From("FROM_ALIAS", new Id(id), new Id(id_alias));:}
      ;

//-------------------------------------------------------------------------------------

where_clause
        ::= WHERE conditional_or_expression:condorexp
{:RESULT = new Where("WHERE", condorexp);:}
      ;

//-------------------------------------------------------------------------------------

conditional_or_expression
        ::= rup_expression:rupexp
{:RESULT = new Condition("RUPEXP", rupexp);:}
       |
        LPAR rup_expression:rupexp RPAR
{:RESULT = new Condition("RUPEXP", rupexp);:}
       |
        NOT LPAR rup_expression:rupexp RPAR
{:RESULT = new BlockNot("BLOCKNOT", new Condition("RUPEXP",rupexp));:}
       |
        conditional_and_expression:condandexp
{:RESULT = condandexp;:}
       |
        conditional_or_expression:condorexp OR conditional_and_expression:condandexp
{:RESULT = new Or("OREXP_OR_ANDEXP", condorexp, condandexp);:}
       |
        conditional_or_expression:condorexp OR rup_expression:rupexp
{:RESULT = new Or("OREXP_OR_RUPEXP", condorexp, rupexp);:}
       |
        LPAR conditional_or_expression:condorexp OR rup_expression:rupexp RPAR
{:RESULT = new Block("BLOCK", new Or("CONDOREXP_OR_RUPEXP", condorexp, rupexp));:}

       |
       conditional_or_expression:condorexp OR LPAR conditional_and_expression:condandexp RPAR
{:RESULT = new Or("OREXP_OR_BLOCK_ANDEXP", condorexp, new Block("BLOCK", condandexp));:}
       |
        NOT LPAR conditional_or_expression:condorexp OR rup_expression:rupexp RPAR
{:RESULT = new BlockNot("BLOCKNOT", new Or("OREXP_OR_RUPEXP",condorexp,rupexp));:}

       |
       conditional_or_expression:condorexp OR NOT LPAR conditional_and_expression:condandexp RPAR
{:RESULT = new Or("OREXP_OR_BLOCKNOT_ANDEXP",condorexp, new BlockNot("BLOCK_NOT", condandexp));:}
      ;

//-------------------------------------------------------------------------------------

conditional_and_expression
        ::= equality_expression:eqexp
{:RESULT = new Condition("EQEXP",eqexp);:}
         |
         conditional_and_expression:condandexp AND equality_expression:eqexp
{:RESULT = new And("ANDEXP_AND_EQEXP",condandexp,eqexp);:}
         |
         conditional_and_expression:condandexp AND rup_expression:rupexp
{:RESULT = new And("ANDEXP_AND_RUPEXP",condandexp,rupexp);:}
         |
         rup_expression:rupexp AND conditional_and_expression:condandexp
{:RESULT = new And("RUPEXP_AND_ANDEXP",rupexp,condandexp);:}
         |
         LPAR rup_expression:rupexp AND conditional_and_expression:condandexp RPAR
{:RESULT = new Block("BLOCK", new And("RUPEXP_AND_ANDEXP",rupexp,condandexp));:}
         |
         conditional_and_expression:condandexp1 AND LPAR conditional_and_expression:condandexp2
RPAR
{:RESULT = new And("ANDEXP_AND_BLOCK", condandexp1, new Block("BLOCK", condandexp2));:}
         |
```

```
        NOT LPAR rup_expression:rupexp AND conditional_and_expression:condandexp RPAR
{:RESULT = new BlockNot("BLOCK_NOT", new And("RUPEXP_AND_ANDEXP", rupexp, condandexp));:}

        |
        conditional_and_expression:condandexp1 AND NOT LPAR
conditional_and_expression:condandexp2 RPAR
{:RESULT = new And("ANDEXP_AND_BLOCKNOT", condandexp1, new BlockNot("BLOCK_NOT", condandexp2));:}
    ;

//-------------------------------------------------------------------------------------

equality_expression
       ::= relational_expression:relexp
{:RESULT = relexp;:}
         |
         relational_expression:relexp1 EQ relational_expression:relexp2
{:RESULT = new EqualityExpression("REL_EQ_REL",relexp1,relexp2);:}
         |
         unary_expression:unexp1 NEQ unary_expression:unexp2
{:RESULT = new EqualityExpression("COND_NEQ",unexp1,unexp2);:}
         |
         unary_expression:unexp1 NOT EQ unary_expression:unexp2
{:RESULT = new EqualityExpression("COND_NEQ",unexp1,unexp2);:}
         ;

//-------------------------------------------------------------------------------------

relational_expression ::=
    additive_expression:addexp
{:RESULT = addexp;:}
     |
    relational_expression:relexp LT additive_expression:addexp
{:RESULT = new RelationalExpression("LT", relexp, addexp);:}
     |
    relational_expression:relexp GT additive_expression:addexp
{:RESULT = new RelationalExpression("GT", relexp, addexp);:}
     |
    relational_expression:relexp LET additive_expression:addexp
{:RESULT = new RelationalExpression("LET", relexp, addexp);:}
     |
    relational_expression:relexp GET additive_expression:addexp
{:RESULT = new RelationalExpression("GET", relexp, addexp);:}
     ;

//-------------------------------------------------------------------------------------

additive_expression ::=
    multiplicative_expression:mulexp
{:RESULT = mulexp;:}
     |
    additive_expression:addexp MAS multiplicative_expression:mulexp
{:RESULT = new AdditiveExpression("MAS", addexp, mulexp);:}
     |
    additive_expression:addexp MENOS multiplicative_expression:mulexp
{:RESULT = new AdditiveExpression("MENOS", addexp, mulexp);:}
     ;

//-------------------------------------------------------------------------------------

multiplicative_expression ::=
    unary_expression:unexp
{:RESULT = unexp;:}
     |
   multiplicative_expression:mulexp POR multiplicative_expression:mulexp2
{:RESULT = new MultiplicativeExpression("POR", mulexp, mulexp2);:}
     |
   multiplicative_expression:mulexp DIV multiplicative_expression:mulexp2
{:RESULT = new MultiplicativeExpression("DIV", mulexp, mulexp2);:}
     ;

//-------------------------------------------------------------------------------------

unary_expression ::=
       CONST_STR:cadena
{:RESULT = new ValueString(cadena);:}
     |
       CONST_INT:entero
```

```
{:RESULT = new ValueInt(entero.toString());:}
      |
        CONST_FLOAT:flotante
{:RESULT = new ValueFloat(flotante.toString());:}
      |
        CONST_DATE:fecha
{:RESULT = new ValueDate(fecha.toString());:}
      |
        identifier_level:id_level
{:RESULT = id_level;:}
      |
        identifier_attribute:id_attribute LPAR CONST_DATE:fecha RPAR
{:RESULT = new AttributeExpression("ID_ATTR_CONST_TIME", id_attribute, new
ValueDate(fecha.toString()));:}
      |
        identifier_attribute:id_attribute LPAR identifier_level:id_level RPAR
{:RESULT = new AttributeExpression("ID_ATTR_TIME", id_attribute, id_level);:}
      ;

//-------------------------------------------------------------------------------------

identifier_attribute ::=
   ID:alias PUNTO ID:attribute
{:RESULT = new IdentifierAttribute("ATTR", new Id(alias), new Id(attribute));:}
    ;

//-------------------------------------------------------------------------------------

rup_expression ::=
      rup_condition:rup_cond
{:RESULT = rup_cond;:}
   |
      NOT rup_condition:rup_cond
{:RESULT = new RupNot("RUP_NOT", rup_cond);:}
   ;

//-------------------------------------------------------------------------------------

rup_condition ::=
   RUP LPAR identifier_rup_from:idrupfrom COMA identifier_rup_to:idrupto COMA CONST_DATE:fecha
RPAR
{:RESULT = new RupExpression("RUP", idrupfrom, idrupto, new ValueDate(fecha.toString()));:}
   |
   RUP LPAR identifier_rup_from:idrupfrom COMA identifier_rup_to:idrupto COMA
identifier_level:id_level RPAR
{:RESULT = new RupExpression("RUP", idrupfrom, idrupto, id_level);:}
   |
   RUP LPAR identifier_rup_from:idrupfrom COMA identifier_rup_to:idrupto COMA CONST_DATE:fecha
COMA ID:alias_instance RPAR
{:RESULT = new RupExpression("RUP", idrupfrom, idrupto, new ValueDate(fecha.toString()), new
Id(alias_instance));:}
   ;

//-------------------------------------------------------------------------------------

identifier_rup_from ::=
   ID:dimensionId
{:RESULT = new RupNoVarIdentifier("RUP_ID_DIM", new IdentifierDimension("DIMENSION", new
Id(dimensionId)));:}
    |
   ID:dimensionId DOSPUNTOS ID:alias
{:RESULT = new RupNoVarIdentifier("RUP_ID_DIM_ALIAS", new IdentifierDimension("DIMENSION", new
Id(dimensionId)), new Id(alias));:}
    |
   ID:dimensionId PUNTO ID:levelId
{:RESULT = new RupNoVarIdentifier("RUP_ID_DIM_LEVEL", new IdentifierLevel("ID_DIM_LEVEL", new
Id(dimensionId), new Id(levelId)));:}
    |
   ID:dimensionId PUNTO ID:levelId DOSPUNTOS CONST_STR:value
{:RESULT = new RupNoVarIdentifier("RUP_ID_DIM_LEVEL_VALUE", new IdentifierLevel("ID_DIM_LEVEL",
new Id(dimensionId), new Id(levelId)), new ValueString(value.toString()));:}
    |
   ID:dimensionId PUNTO ID:levelId DOSPUNTOS ID:alias
{:RESULT = new RupNoVarIdentifier("RUP_ID_DIM_LEVEL_ALIAS", new IdentifierLevel("ID_DIM_LEVEL",
new Id(dimensionId), new Id(levelId)), new Id(alias));:}
    |
   VAR ID:id
```

```
{:RESULT = new RupVarIdentifier("RUP_VAR_ID", new Id(id));:}
    |
   VAR ID:id1 DOSPUNTOS VAR ID:id2
{:RESULT = new RupVarIdentifier("RUP_VAR_ID_ID", new Id(id1), new Id(id2));:}
    |
   VAR ID:id DOSPUNTOS VAR CONST_STR:value
{:RESULT = new RupVarIdentifier("RUP_VAR_ID_VALUE", new Id(id), new
ValueString(value.toString()));:}
    ;

//--------------------------------------------------------------------------------

identifier_rup_to ::=
   identifier_level:levelId
{:RESULT = new RupNoVarIdentifier("RUP_ID", levelId);:}
    |
   identifier_level:levelId DOSPUNTOS CONST_STR:value
{:RESULT = new RupNoVarIdentifier("RUP_ID_VALUE", levelId,  new ValueString(value.toString()));:}

    |
   identifier_level:levelId DOSPUNTOS ID:alias
{:RESULT = new RupNoVarIdentifier("RUP_ID_ALIAS", levelId, new Id(alias));:}
    |
   VAR ID:id
{:RESULT = new RupVarIdentifier("RUP_VAR_ID", new Id(id));:}
    |
   VAR ID:id1 DOSPUNTOS VAR ID:id2
{:RESULT = new RupVarIdentifier("RUP_VAR_ID_ID", new Id(id1), new Id(id2));:}
    |
   VAR ID:id DOSPUNTOS VAR CONST_STR:value
{:RESULT = new RupVarIdentifier("RUP_VAR_ID_VALUE", new Id(id), new Id(value));:}
    ;

//--------------------------------------------------------------------------------

store_clause ::=
   STORE AS ID:id
{:RESULT = new Store("STORE", new Id(id));:}
   ;
```

ANEXO 4: EJEMPLOS DE TRADUCCIONES TOLAP-QL

A continuación, se brindan ejemplos de consultas y para cada una de ellas:

- Tabla de Símbolos
- Traducción a SQL
- Resultado

Ejemplo 1)

Consulta

```
SELECT P.brand, SUM(amount)
  FROM Sales F, Product P, Geography G
 WHERE F.Product = P.bottom
   AND F.Geography = G.bottom
   AND RUP(G,region:'r1',F.t)
   AND RUP(P,brand,F.t);
```

Tabla de Símbolos

```
Scope Global
    Palabra reservada SELECT
    Palabra reservada FROM
    Palabra reservada WHERE
    Palabra reservada COUNT
    Palabra reservada SUM
    Palabra reservada AVG
    Palabra reservada MAX
    Palabra reservada MIN
    Palabra reservada NOT
    Palabra reservada AND
    Palabra reservada OR
    Palabra reservada AS
    Palabra reservada VAR
    Palabra reservada RUP
    Palabra reservada STORE
    Scope Select
        Alias F Sales Fact
        Alias P Product Dimension
        Alias G Geography Dimension
        Scope Where
```

Traducción a SQL

```
SELECT col1, SUM(col2) AS col2
  FROM (
      SELECT P.brand col1, SUM(amount) col2
            FROM Sales_1 F, DIMENSION_Product P, DIMENSION_Geography G
            WHERE F.Product_itemId = P.itemId AND F.Geography_city =
                  G.city AND instant BETWEEN G.instant_from AND CASE
                  WHEN G.instant_to IS NULL THEN GETDATE() ELSE
                  G.instant_to END AND G.region = 'r1' AND instant
                  BETWEEN P.instant_from AND CASE WHEN P.instant_to IS
                  NULL THEN GETDATE() ELSE P.instant_to END AND P.brand
                  IS NOT NULL GROUP BY P.brand
       UNION
       SELECT P.brand col1, SUM(amount) col2
            FROM Sales_2 F, DIMENSION_Product P, DIMENSION_Geography G
            WHERE F.Product_itemId = P.itemId AND F.Geography_town =
                  G.town AND instant BETWEEN G.instant_from AND CASE
                  WHEN G.instant_to IS NULL THEN GETDATE() ELSE
                  G.instant_to END AND G.region = 'r1' AND instant
                  BETWEEN P.instant_from AND CASE WHEN P.instant_to IS
                  NULL THEN GETDATE() ELSE P.instant_to END AND P.brand
                  IS NOT NULL GROUP BY P.brand
       UNION
       SELECT P.brand col1, SUM(amount) col2
            FROM Sales_3 F, DIMENSION_Product P, DIMENSION_Geography G
            WHERE F.Product_itemId = P.itemId AND F.Geography_square =
                  G.square AND instant BETWEEN G.instant_from AND CASE
                  WHEN G.instant_to IS NULL THEN GETDATE() ELSE
                  G.instant_to END AND G.region = 'r1' AND instant
                  BETWEEN P.instant_from AND CASE WHEN P.instant_to IS
                  NULL THEN GETDATE() ELSE P.instant_to END AND P.brand
                  IS NOT NULL GROUP BY P.brand) UNIONS
 GROUP BY col1
```

Resultado

```
b1 700.00
b2 400.00
b3 111640.00
```

Ejemplo 2)

Consulta

```
SELECT P.brand AS brand, SUM(amount) AS TotalBA
  FROM Sales F, Product P, Geography G
 WHERE F.Product = P.bottom
   AND F.Geography = G.bottom
   AND RUP(P,brand,F.t)
   AND RUP(G,city:c,F.t)
   AND c.name(F.t)='Ottawa'
STORE AS Q1;
SELECT P.brand, SUM(amount)
```

```
  FROM Sales F, Product P, Q1
 WHERE F.Product = P.bottom
   AND RUP(P,brand:b,F.t)
   AND Q1.brand =b
   AND Q1.TotalBA >100000;
```

Tabla de Símbolos

```
Scope Global
    Palabra reservada SELECT
    Palabra reservada FROM
    Palabra reservada WHERE
    Palabra reservada COUNT
    Palabra reservada SUM
    Palabra reservada AVG
    Palabra reservada MAX
    Palabra reservada MIN
    Palabra reservada NOT
    Palabra reservada AND
    Palabra reservada OR
    Palabra reservada AS
    Palabra reservada VAR
    Palabra reservada RUP
    Palabra reservada STORE
    Table Q1 Store
        Scope Store
            Alias F Sales Fact
            Alias P Product Dimension
            Alias G Geography Dimension
            Scope Where
                Alias c G  String
            Column brand P  String
            Column TotalBA Double
    Scope Select
        Alias F Sales Fact
        Alias P Product Dimension
        Table Q1 Store
        Scope Where
            Alias b P  String
```

Traducción a SQL

Creación de VIEW temporal

```
CREATE VIEW STORE_177 AS
SELECT brand, SUM(TotalBA) AS TotalBA
  FROM (
       SELECT P.brand AS brand, SUM(amount) AS TotalBA
             FROM Sales_1 F, DIMENSION_Product P, DIMENSION_Geography G
             WHERE F.Product_itemId = P.itemId AND F.Geography_city =
                   G.city AND instant BETWEEN P.instant_from AND CASE
                   WHEN P.instant_to IS NULL THEN GETDATE() ELSE
                   P.instant_to END AND instant BETWEEN G.instant_from
                   AND CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE
                   G.instant_to END AND G.city IN (SELECT value   FROM
                   tinstance_attribute_SS WHERE  attribute_id = 34 AND
```

```
                value_at = 'Ottawa' AND instant BETWEEN instant_from
                AND CASE WHEN instant_to IS NULL THEN GETDATE() ELSE
                instant_to END) AND P.brand IS NOT NULL GROUP BY
                P.brand
        UNION


        SELECT P.brand AS brand, SUM(amount) AS TotalBA
             FROM Sales_2 F, DIMENSION_Product P, DIMENSION_Geography G
             WHERE F.Product_itemId = P.itemId AND F.Geography_town =
                G.town AND instant BETWEEN P.instant_from AND CASE
                WHEN P.instant_to IS NULL THEN GETDATE() ELSE
                P.instant_to END AND instant BETWEEN G.instant_from
                AND CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE
                G.instant_to END AND G.city IN (SELECT value   FROM
                tinstance_attribute_SS WHERE  attribute_id = 34 AND
                value_at = 'Ottawa' AND instant BETWEEN instant_from
                AND CASE WHEN instant_to IS NULL THEN GETDATE() ELSE
                instant_to END) AND P.brand IS NOT NULL GROUP BY
                P.brand
        UNION
        SELECT P.brand AS brand, SUM(amount) AS TotalBA
             FROM Sales_3 F, DIMENSION_Product P, DIMENSION_Geography G
             WHERE F.Product_itemId = P.itemId AND F.Geography_square =
                G.square AND instant BETWEEN P.instant_from AND CASE
                WHEN P.instant_to IS NULL THEN GETDATE() ELSE
                P.instant_to END AND instant BETWEEN G.instant_from
                AND CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE
                G.instant_to END AND G.city IN (SELECT value   FROM
                tinstance_attribute_SS WHERE  attribute_id = 34 AND
                value_at = 'Ottawa' AND instant BETWEEN instant_from
                AND CASE WHEN instant_to IS NULL THEN GETDATE() ELSE
                instant_to END) AND P.brand IS NOT NULL GROUP BY
                P.brand)
       UNIONS GROUP BY brand
```

Consulta con uso de la vista temporal

```
SELECT col1, SUM(col2) AS col2
  FROM (
       SELECT P.brand col1, SUM(amount) col2
             FROM Sales_1 F, DIMENSION_Product P, STORE_177
             WHERE F.Product_itemId = P.itemId AND instant BETWEEN
                P.instant_from AND CASE WHEN P.instant_to IS NULL THEN
                GETDATE() ELSE P.instant_to END AND STORE_177.brand =
                P.brand AND STORE_177.TotalBA > 100000 AND P.brand IS
                NOT NULL GROUP BY P.brand
        UNION
       SELECT P.brand col1, SUM(amount) col2
             FROM Sales_2 F, DIMENSION_Product P, STORE_177
             WHERE F.Product_itemId = P.itemId AND instant BETWEEN
                P.instant_from AND CASE WHEN P.instant_to IS NULL THEN
                GETDATE() ELSE P.instant_to END AND STORE_177.brand =
                P.brand AND STORE_177.TotalBA > 100000 AND P.brand IS
                NOT NULL GROUP BY P.brand
       UNION
       SELECT P.brand col1, SUM(amount) col2
             FROM Sales_3 F, DIMENSION_Product P, STORE_177
             WHERE F.Product_itemId = P.itemId AND instant BETWEEN
                P.instant_from AND CASE WHEN P.instant_to IS NULL THEN
```

```
                    GETDATE() ELSE P.instant_to END AND STORE_177.brand =
                    P.brand AND STORE_177.TotalBA > 100000 AND P.brand IS
                    NOT NULL GROUP BY P.brand)
 UNIONS GROUP BY col1
```

Resultado

```
b3  111665.00
```

Ejemplo 3)

Consulta

```
SELECT T.week, SUM(qty)
FROM Time T, Services F, Doctor D
WHERE F.Time = T.bottom AND
      F.Doctor = D.bottom AND
      RUP(D,doctorId:d,F.t) AND d.name='Martinez' AND
     (RUP(D,doctorId:d1,F.t,d)AND d1.name='Feinsilver'));
```

Traducción a SQL

```
SELECT col1, SUM(col2) AS col2
  FROM (SELECT T.week col1, SUM(F.qty) col2
          FROM DIMENSION_Time T, Services_1 F, DIMENSION_Doctor D
         WHERE F.Time_day = T.day
           AND F.Doctor_doctorId = D.doctorId
           AND F.instant BETWEEN D.instant_from
           AND CASE WHEN D.instant_to IS NULL THEN GETDATE()
                                        ELSE D.instant_to END
           AND D.doctorId IN
                       (SELECT value FROM tinstance_attribute_SS WHERE
                   attribute_id = 33 AND value_at = 'Martinez')
           AND NOT EXISTS (SELECT 1
                             FROM Services_1 F1, DIMENSION_Doctor D
                            WHERE F.instant = F1.instant
                              AND F.instant BETWEEN D.instant_from AND
                              CASE WHEN D.instant_to IS NULL THEN
                              GETDATE() ELSE D.instant_to END
                              AND F1.Doctor_doctorId = D.doctorId
                              AND D.doctorId IN (SELECT value   FROM
                              tinstance_attribute_SS WHERE  attribute_id
                              = 33 AND value_at = 'Feinsilver'
                              AND F.instant BETWEEN instant_from AND CASE
                              WHEN instant_to IS NULL THEN GETDATE() ELSE
                              instant_to))
         AND T.week IS NOT NULL GROUP BY T.week
         UNION
         SELECT T.week col1, SUM(F.qty) col2
               FROM DIMENSION_Time T, Services_2 F, DIMENSION_Doctor D
               WHERE F.Time_day = T.day
                   AND F.Doctor_doctorId = D.doctorId
                   AND F.instant BETWEEN D.instant_from AND CASE WHEN
                   D.instant_to IS NULL THEN GETDATE() ELSE D.instant_to
                   END AND D.doctorId IN (SELECT value   FROM
                         tinstance_attribute_SS WHERE  attribute_id = 33
                         AND value_at = 'Martinez')
                   AND NOT EXISTS (SELECT 1
```

```
                              FROM Services_2 F1, DIMENSION_Doctor D
                             WHERE F.instant = F1.instant
                               AND F.instant BETWEEN D.instant_from AND
                                CASE WHEN D.instant_to IS NULL THEN
                                GETDATE() ELSE D.instant_to END
                               AND F1.Doctor_doctorId = D.doctorId
                               AND D.doctorId IN (SELECT value   FROM
                                tinstance_attribute_SS WHERE attribute_id =
                                33 AND value_at = 'Feinsilver'
                               AND F.instant BETWEEN instant_from AND CASE
                                WHEN instant_to IS NULL THEN GETDATE() ELSE
                                instant_to))
                     AND T.week IS NOT NULL
               GROUP BY T.week
         UNION
         SELECT T.week col1, SUM(F.qty) col2
               FROM DIMENSION_Time T, Services_3 F, DIMENSION_Doctor D
               WHERE F.Time_day = T.day AND F.Doctor_doctorId =
                   D.doctorId AND F.instant BETWEEN D.instant_from
                   AND CASE WHEN D.instant_to IS NULL THEN GETDATE()
                   ELSE D.instant_to END AND D.doctorId IN
                         (SELECT value   FROM tinstance_attribute_SS
                               WHERE  attribute_id = 33 AND value_at =
                               'Martinez')
                   AND NOT EXISTS (SELECT 1
                               FROM Services_3 F1, DIMENSION_Doctor D
                               WHERE F.instant = F1.instant
                                AND F.instant BETWEEN D.instant_from AND
                               CASE WHEN D.instant_to IS NULL THEN
                               GETDATE() ELSE D.instant_to END
                                AND F1.Doctor_doctorId = D.doctorId
                                AND D.doctorId IN (SELECT value
                                      FROM tinstance_attribute_SS
                                      WHERE attribute_id = 33 AND value_at
                                            = 'Feinsilver'AND F.instant
                                      BETWEEN instant_from AND CASE WHEN
                                      instant_to IS NULL THEN GETDATE()
                                      ELSE instant_to))
                   AND T.week IS NOT NULL
         GROUP BY T.week) UNIONS
GROUP BY col1
```

Tabla de Símbolos

```
Scope Global
    Palabra reservada SELECT
    Palabra reservada FROM
    Palabra reservada WHERE
    Palabra reservada COUNT
    Palabra reservada SUM
    Palabra reservada AVG
    Palabra reservada MAX
    Palabra reservada MIN
    Palabra reservada NOT
    Palabra reservada AND
    Palabra reservada OR
    Palabra reservada AS
    Palabra reservada VAR
    Palabra reservada RUP
    Palabra reservada STORE
```

```
Scope Select
        Alias T Time Dimension
        Alias F Services Fact
        Alias D Doctor Dimension
        Scope Where
           Alias d D  String
           Scope BlockNot
              Alias d1 D  String
```

En este caso se observa el scope del Block NOT con la variable d1 como parte de él.

Resultado

```
w1 130
```

Ejemplo 4)

Consulta

```
SELECT itemId, X, x
  FROM Product P
 WHERE RUP(P.itemId, VAR X:VAR x ,'2006/02/05 15:04:00');
```

Tabla de Símbolos

```
Scope Global
    Palabra reservada SELECT
    Palabra reservada FROM
    Palabra reservada WHERE
    Palabra reservada COUNT
    Palabra reservada SUM
    Palabra reservada AVG
    Palabra reservada MAX
    Palabra reservada MIN
    Palabra reservada NOT
    Palabra reservada AND
    Palabra reservada OR
    Palabra reservada AS
    Palabra reservada VAR
    Palabra reservada RUP
    Palabra reservada STORE
    Scope Select
        Alias P Product Dimension
        Scope Where
            VarLevel X P
            VarInstance x P
```

En este caso se observan las definiciones de las variables de tipo VarLevel y VarInstance representando las rollups e instancias que están relacionadas con el ínfimo de la Dimension Product.

Traducción a SQL

```
SELECT DISTINCT P.itemId, 'category' AS category, 'brand' AS brand,
       category AS instance_category, brand AS instance_brand
 FROM DIMENSION_Product P
 WHERE convert(datetime, '2006/02/05 15:04:00', 121) BETWEEN
       P.instant_from AND CASE WHEN P.instant_to IS NULL THEN GETDATE()
       ELSE P.instant_to END
```

Resultado

```
i1  category  brand  c1  b1
i2  category  brand  c1  b2
i3  category  brand  c2  b3
i4  category  brand  c2  b3
```

En la traducción a SQL, se disponen las columnas y se corresponde el resultado con el orden posicional, por ej, en este caso, la disposición es la siguiente: itemId, "Nombre del Nivel1 rollup con el de la sentencia RUP", "Nombre del Nivel2 rollup con el de la sentencia RUP", "Instancia de Nivel1, Instancia de Nivel2.

Siempre se indican como columnas los nombres de los niveles hacia izquierda y posicionalmente las instancias (a partir de la columna siguiente al último nivel) a la derecha.

En este ejemplo: i1 de itemId, "category", "brand", c1 (instancia de category), b1 (instancia de brand).

ANEXO 5: TEST DEL CASO DE ESTUDIO

Consultas del caso de estudio

Q1) Consulta: *Listar las localidades de la provincia de Buenos Aires.*

En TOLAP-QL se escribe:

```
SELECT localidadId
 FROM Geografia G
 WHERE RUP(G.localidadId, provinceId:p, t)
       AND p.nombre(t) = 'Buenos Aires';
```

Traducción

```
SELECT DISTINCT G.localidadId
FROM DIMENSION_Geografia G
WHERE G.instant_from BETWEEN G.instant_from AND CASE
      WHEN G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to END AND
      G.provinceId IN (
            SELECT value   FROM tinstance_attribute_NS
            WHERE  attribute_id = 54 AND value_at = 'Buenos Aires' AND
            G.instant_from BETWEEN instant_from AND
            CASE WHEN instant_to IS NULL THEN GETDATE() ELSE instant_to
            END)
```

Tiempo de Ejecución : 00:00:00,375

Q2) Consulta: *Indicar si existía una apertura geográfica por provincia para la fecha* '2001/01/03''.

En TOLAP-QL se escribe:

```
SELECT boolean
 FROM Geografia G
 WHERE RUP(G, provinceId, '2001/01/03');
```

Traducción

```
SELECT DISTINCT 1
FROM DIMENSION_Geografia G
WHERE convert(datetime, '2001/01/03', 121) BETWEEN G.instant_from AND
      CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to
                                                          END
```

Tiempo de Ejecución : 00:00:00,063

Q3) Consulta: *Indicar si existía una apertura geográfica por provincia para la fecha '2006/06/01'.* Esta consulta es la misma que Q2, pero para otro instante, a los efectos de verificar que la evaluación lógica se realiza correctamente.

En TOLAP-QL se escribe:

```
SELECT boolean
 FROM Geografia G
 WHERE RUP(G, provinceId, '2006/06/01');
```

Traducción

```
SELECT DISTINCT 1
FROM DIMENSION_Geografia G
WHERE convert(datetime, '2006/06/01', 121) BETWEEN G.instant_from AND
      CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to
                                                                  END
```

Tiempo de Ejecución : 00:00:00,016

Q4) Consulta: *Listar las localidades y la zona de pertenencia de las mismas a la fecha.*

En TOLAP-QL se escribe:

```
SELECT localidadId, X, x
 FROM Geografia G
 WHERE  RUP(G.localidadId, VAR X:VAR x ,NOW);
```

Traducción

```
SELECT DISTINCT G.localidadId, 'provinceId' AS provinceId,
      provinceId AS instance_provinceId
FROM DIMENSION_Geografia G
WHERE convert(datetime, '2007-04-18 01:56:44', 121)
      BETWEEN G.instant_from AND
      CASE WHEN G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to
                                                                  END
```

Tiempo de Ejecución : 00:00:00,359

Q5) Consulta: *Cantidad de prestamos brindados en la provincia de Buenos Aires.*

En TOLAP-QL se escribe:

```
SELECT G.provinceId, COUNT(*)
FROM Prestamos P, Geografia G
WHERE P.Geografia = G.bottom
  AND RUP(G, provinceId:prov, P.t)
  AND prov.nombre(P.t) = 'Buenos Aires'
  AND P.amount > 0;
```

Traducción (Sin optimización)

```
SELECT col1, SUM(col2) AS col2
      FROM (
            SELECT G.provinceId col1, COUNT(*) col2
            FROM Prestamos_2 P, DIMENSION_Geografia G
            WHERE P.Geografia_provinceId = G.provinceId AND P.instant
                  BETWEEN G.instant_from AND CASE WHEN G.instant_to IS NULL
                  THEN GETDATE() ELSE G.instant_to END AND G.provinceId IN
                        (SELECT value   FROM tinstance_attribute_NS
                              WHERE  attribute_id = 54 AND
                               value_at = 'Buenos Aires' AND
                               P.instant BETWEEN instant_from AND CASE WHEN
                               instant_to IS NULL THEN GETDATE() ELSE
                              instant_to
                               END)
                  AND P.amount > 0 AND G.provinceId IS NOT NULL
            GROUP BY G.provinceId
            UNION
            SELECT G.provinceId col1, COUNT(*) col2
            FROM Prestamos_3 P, DIMENSION_Geografia G
            WHERE P.Geografia_localidadId = G.localidadId AND P.instant
                  BETWEEN G.instant_from AND CASE WHEN G.instant_to IS NULL
                  THEN GETDATE() ELSE G.instant_to END AND G.provinceId IN
                        (SELECT value   FROM tinstance_attribute_NS
                              WHERE  attribute_id = 54 AND
                               value_at = 'Buenos Aires' AND
                               P.instant BETWEEN instant_from AND CASE WHEN
                               instant_to IS NULL THEN GETDATE() ELSE
                        instant_to
                               END)
                  AND P.amount > 0 AND G.provinceId IS NOT NULL
            GROUP BY G.provinceId)
      UNIONS GROUP BY col1
```

Tiempo de Ejecución : 01:16:47,765

Traducción (Con optimizacion)

No aplica debido a la condición: `prov.nombre(P.t) = 'Buenos Aires' y P.amount > 0`

Q6) Consulta: *Para cada entidad que asiste a 'SA LA NACION' y a 'EDITORIAL EL ATLANTICO S A' indicar el saldo actual de su cartera de créditos.*

En TOLAP-QL se escribe:

```
SELECT P.EntidadesFinancieras, SUM(P.amount)
FROM Prestamos P, Deudores D
WHERE P.Deudores = D.bottom
  AND RUP(D, CUIT:c, P.t) AND c.nombre = 'SA LA NACION'
  AND (RUP(D, CUIT:c1, P.t, c) AND
            c1.nombre = 'EDITORIAL EL ATLANTICO S A');
```

Traducción (Sin optimizacion)

```
SELECT col1, SUM(col2) AS col2
      FROM (
            SELECT P.EntidadesFinancieras_EntidadId col1, SUM(P.amount) col2
                  FROM Prestamos_1 P, DIMENSION_Deudores D
                  WHERE P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
                   D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
            GETDATE()
                   ELSE D.instant_to END AND D.CUIT IN
                        (SELECT value   FROM tinstance_attribute_SS
                              WHERE  attribute_id = 44 AND
                               value_at = 'SA LA NACION' AND P.instant BETWEEN
                               instant_from AND CASE WHEN instant_to IS NULL
                        THEN
                               GETDATE() ELSE instant_to END)
                   AND EXISTS (SELECT 1 FROM Prestamos_1 P1,
                                        DIMENSION_Deudores D
                                 WHERE P.instant = P1.instant AND P.instant
                                 BETWEEN D.instant_from AND CASE WHEN
                                 D.instant_to IS NULL THEN GETDATE() ELSE
                                 D.instant_to END AND P1.Deudores_CUIT =
                                D.CUIT
                                 AND D.CUIT IN
                                      (SELECT value FROM
                                tinstance_attribute_SS
                                        WHERE  attribute_id = 44 AND
                                        value_at = 'EDITORIAL EL ATLANTICO
                                      S A'
                                        AND P.instant BETWEEN instant_from
                                      AND
                                        CASE WHEN instant_to IS NULL THEN
                                        GETDATE() ELSE instant_to END)
                              )
                  GROUP BY P.EntidadesFinancieras_EntidadId
            UNION
```

```
SELECT P.EntidadesFinancieras_EntidadId col1, SUM(P.amount) col2
      FROM Prestamos_2 P, DIMENSION_Deudores D
      WHERE P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
       D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
GETDATE()
       ELSE D.instant_to END AND D.CUIT IN
            (SELECT value   FROM tinstance_attribute_SS
                  WHERE  attribute_id = 44 AND
                   value_at = 'SA LA NACION' AND P.instant BETWEEN
                   instant_from AND CASE WHEN instant_to IS NULL
            THEN
                   GETDATE() ELSE instant_to END)
      AND EXISTS (SELECT 1 FROM Prestamos_2 P1,
                              DIMENSION_Deudores D
                        WHERE P.instant = P1.instant AND P.instant
                        BETWEEN D.instant_from AND CASE WHEN
                        D.instant_to IS NULL THEN GETDATE() ELSE
                        D.instant_to END AND P1.Deudores_CUIT =
                       D.CUIT
                        AND D.CUIT IN
                              (SELECT value FROM
                       tinstance_attribute_SS
                              WHERE  attribute_id = 44 AND
                              value_at = 'EDITORIAL EL ATLANTICO
                       S A'
                              AND P.instant BETWEEN instant_from
                       AND
                              CASE WHEN instant_to IS NULL THEN
                              GETDATE() ELSE instant_to END)
                  )
      GROUP BY P.EntidadesFinancieras_EntidadId
UNION
SELECT P.EntidadesFinancieras_EntidadId col1, SUM(P.amount) col2
      FROM Prestamos_3 P, DIMENSION_Deudores D
      WHERE P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
       D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
GETDATE()
       ELSE D.instant_to END AND D.CUIT IN
            (SELECT value   FROM tinstance_attribute_SS
                  WHERE  attribute_id = 44 AND
                   value_at = 'SA LA NACION' AND P.instant BETWEEN
                   instant_from AND CASE WHEN instant_to IS NULL
            THEN
                   GETDATE() ELSE instant_to END)
      AND EXISTS (SELECT 1 FROM Prestamos_3 P1,
                               DIMENSION_Deudores D
                        WHERE P.instant = P1.instant AND P.instant
                        BETWEEN D.instant_from AND CASE WHEN
                        D.instant_to IS NULL THEN GETDATE() ELSE
                        D.instant_to END AND P1.Deudores_CUIT =
                  D.CUIT
                        AND D.CUIT IN
                              (SELECT value FROM
                              tinstance_attribute_SS
                              WHERE  attribute_id = 44 AND
                              value_at = 'EDITORIAL EL ATLANTICO
                       S A'
                              AND P.instant BETWEEN instant_from
                       AND
                              CASE WHEN instant_to IS NULL THEN
                              GETDATE() ELSE instant_to END)
```

```
                              )
                    GROUP BY P.EntidadesFinancieras_EntidadId)
UNIONS GROUP BY col1
```

Tiempo de Ejecución : 00:12:58,453

Traducción (Con optimizacion)

No aplica debido a la condición : `c.nombre = 'SA LA NACION' y c1.nombre = 'EDITORIAL EL ATLANTICO S A');`

Q7) Consulta: *Listar la Clave Tributaria y deuda total en el Sistema Financiero de 'SA LA NACION'.*

En TOLAP-QL se escribe:

```
SELECT D.CUIT, SUM(amount)
  FROM Prestamos P, Deudores D, Asistencias A
 WHERE P.Asistencias = A.bottom
   AND P.Deudores = D.bottom
   AND RUP(D, CUIT:c, P.t)
   AND RUP(A, Asistencia, P.t)
   AND c.nombre(P.t) = 'SA LA NACION'
```

Traducción (Sin optimizacion)

```
SELECT col1, SUM(col2) AS col2
      FROM (
            SELECT D.CUIT col1, SUM(P.amount) col2
              FROM Prestamos_1 P, DIMENSION_Deudores D,
                  DIMENSION_Asistencias A
              WHERE P.Asistencias_Asistencia = A.Asistencia AND
                  P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
            D.instant_from AND
                  CASE WHEN D.instant_to IS NULL THEN GETDATE() ELSE
            D.instant_to
                  END AND P.instant BETWEEN A.instant_from AND CASE WHEN
                  A.instant_to IS NULL THEN GETDATE() ELSE A.instant_to END
            AND
                  D.CUIT IN (
                        SELECT value   FROM tinstance_attribute_SS
                           WHERE  attribute_id = 44 AND value_at = 'SA LA
                  NACION'
                              AND P.instant BETWEEN instant_from AND CASE
                        WHEN
```

```
                        instant_to IS NULL THEN GETDATE() ELSE
                  instant_to
                        END)
          AND D.CUIT IS NOT NULL
    GROUP BY D.CUIT
  UNION
  SELECT D.CUIT col1, SUM(P.amount) col2
    FROM Prestamos_2 P, DIMENSION_Deudores D,
        DIMENSION_Asistencias A
    WHERE P.Asistencias_Asistencia = A.Asistencia AND
        P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
  D.instant_from AND
        CASE WHEN D.instant_to IS NULL THEN GETDATE() ELSE
  D.instant_to
        END AND P.instant BETWEEN A.instant_from AND CASE WHEN
        A.instant_to IS NULL THEN GETDATE() ELSE A.instant_to END
  AND
        D.CUIT IN (
              SELECT value   FROM tinstance_attribute_SS
                 WHERE  attribute_id = 44 AND value_at = 'SA LA
              NACION'
                    AND P.instant BETWEEN instant_from AND CASE WHEN
                    instant_to IS NULL THEN GETDATE() ELSE
              instant_to
                    END)
          AND D.CUIT IS NOT NULL
     GROUP BY D.CUIT
  UNION
  SELECT D.CUIT col1, SUM(P.amount) col2
    FROM Prestamos_3 P, DIMENSION_Deudores D,
        DIMENSION_Asistencias A
    WHERE P.Asistencias_Asistencia = A.Asistencia AND
        P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
  D.instant_from AND
        CASE WHEN D.instant_to IS NULL THEN GETDATE() ELSE
  D.instant_to
        END AND P.instant BETWEEN A.instant_from AND CASE WHEN
        A.instant_to IS NULL THEN GETDATE() ELSE A.instant_to END
  AND
        D.CUIT IN (
              SELECT value   FROM tinstance_attribute_SS
                 WHERE  attribute_id = 44 AND value_at = 'SA LA
        NACION'
                    AND P.instant BETWEEN instant_from AND CASE WHEN
                    instant_to IS NULL THEN GETDATE() ELSE
              instant_to
                    END)
          AND D.CUIT IS NOT NULL
    GROUP BY D.CUIT)
UNIONS GROUP BY col1
```

Tiempo de Ejecución : 00:26:51,579

Traducción (Con optimizacion)

No aplica debido a la condición : `c.nombre(P.t) = 'SA LA NACION'`

Q8) Consulta: *Listar la cantidad de Mujeres que poseen deudas hipotecarias en la provincia de Cordoba.*

En TOLAP-QL se escribe:

```
SELECT D.CUIT, SUM(amount)
  FROM Prestamos P, Deudores D, Asistencias A, Geografia G
 WHERE P.Asistencias = A.bottom
   AND P.Deudores = D.bottom
   AND P.Geografia = G.bottom
   AND RUP(D, CUIT:c, P.t)
   AND c.sexo(P.t) = 'F'
   AND RUP(A, Asistencia:a, P.t)
   AND a.descripcion = 'Hipotecarios sobre la vivienda'
   AND RUP(G, provinceId:p, P.t)
   AND p.nombre(P.t) = 'Cordoba'
   AND P.amount > 0
STORE AS Q1;

SELECT COUNT(*)
  FROM Q1;
```

Traducción (Sin optimizacion)

```
CREATE VIEW STORE_204 AS
      SELECT col1, SUM(col2) AS col2
            FROM (
                  SELECT D.CUIT col1, SUM(P.amount) col2
                    FROM Prestamos_1 P, DIMENSION_Deudores D,
                        DIMENSION_Asistencias A, DIMENSION_Geografia G
                    WHERE P.Asistencias_Asistencia = A.Asistencia AND
                        P.Deudores_CUIT = D.CUIT AND P.Geografia_Region =
                  G.Region
                        AND P.instant BETWEEN D.instant_from AND CASE WHEN
                        D.instant_to IS NULL THEN GETDATE() ELSE D.instant_to
                  END
                        AND D.CUIT IN (
                              SELECT value   FROM tinstance_attribute_SS
                              WHERE  attribute_id = 48 AND value_at = 'F' AND
                                    P.instant BETWEEN instant_from AND CASE
                              WHEN
                                    instant_to IS NULL THEN GETDATE() ELSE
                                    instant_to END)
                        AND P.instant BETWEEN A.instant_from AND CASE WHEN
                        A.instant_to IS NULL THEN GETDATE() ELSE A.instant_to
                  END
                        AND A.Asistencia IN (
                              SELECT value   FROM tinstance_attribute_NS
                              WHERE  attribute_id = 38 AND
                                    value_at = 'Hipotecarios sobre la vivienda'
                              AND
                                    P.instant BETWEEN instant_from AND CASE
                              WHEN
                                    instant_to IS NULL THEN GETDATE() ELSE
```

```
                  instant_to END)
      AND P.instant BETWEEN G.instant_from AND CASE WHEN
      G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to
END
      AND G.provinceId IN (
            SELECT value   FROM tinstance_attribute_NS
            WHERE  attribute_id = 54 AND value_at =
      'Cordoba' AND
                  P.instant BETWEEN instant_from AND CASE
            WHEN
                  instant_to IS NULL THEN GETDATE() ELSE
                  instant_to END)
      AND P.amount > 0 AND D.CUIT IS NOT NULL
  GROUP BY D.CUIT
UNION
SELECT D.CUIT col1, SUM(P.amount) col2
  FROM Prestamos_2 P, DIMENSION_Deudores D,
      DIMENSION_Asistencias A, DIMENSION_Geografia G
  WHERE P.Asistencias_Asistencia = A.Asistencia AND
      P.Deudores_CUIT = D.CUIT AND P.Geografia_provinceId =
      G.provinceId AND P.instant BETWEEN D.instant_from AND
CASE
      WHEN D.instant_to IS NULL THEN GETDATE() ELSE
D.instant_to
      END AND D.CUIT IN (
            SELECT value   FROM tinstance_attribute_SS
            WHERE  attribute_id = 48 AND value_at = 'F' AND
                  P.instant BETWEEN instant_from AND CASE
            WHEN
                  instant_to IS NULL THEN GETDATE() ELSE
                  instant_to END)
      AND P.instant BETWEEN A.instant_from AND CASE WHEN
      A.instant_to IS NULL THEN GETDATE() ELSE A.instant_to
END
      AND A.Asistencia IN (
            SELECT value   FROM tinstance_attribute_NS
            WHERE  attribute_id = 38 AND
                  value_at = 'Hipotecarios sobre la vivienda'
            AND
                  P.instant BETWEEN instant_from AND CASE
            WHEN
                  instant_to IS NULL THEN GETDATE() ELSE
                  instant_to END)
      AND P.instant BETWEEN G.instant_from AND CASE WHEN
      G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to
END
      AND G.provinceId IN (
            SELECT value   FROM tinstance_attribute_NS
            WHERE  attribute_id = 54 AND value_at =
      'Cordoba' AND
                  P.instant BETWEEN instant_from AND CASE
            WHEN
                  instant_to IS NULL THEN GETDATE() ELSE
                  instant_to END)
      AND P.amount > 0 AND D.CUIT IS NOT NULL
  GROUP BY D.CUIT
UNION
SELECT D.CUIT col1, SUM(P.amount) col2
  FROM Prestamos_3 P, DIMENSION_Deudores D,
      DIMENSION_Asistencias A, DIMENSION_Geografia G
  WHERE P.Asistencias_Asistencia = A.Asistencia AND
```

```
                        P.Deudores_CUIT = D.CUIT AND P.Geografia_localidadId =
                        G.localidadId AND P.instant BETWEEN D.instant_from AND
                  CASE
                        WHEN D.instant_to IS NULL THEN GETDATE() ELSE
                  D.instant_to
                        END AND D.CUIT IN (
                              SELECT value   FROM tinstance_attribute_SS
                              WHERE  attribute_id = 48 AND value_at = 'F' AND
                                    P.instant BETWEEN instant_from AND CASE
                              WHEN
                                    instant_to IS NULL THEN GETDATE() ELSE
                                    instant_to END)
                        AND P.instant BETWEEN A.instant_from AND CASE WHEN
                        A.instant_to IS NULL THEN GETDATE() ELSE A.instant_to
                  END
                        AND A.Asistencia IN (
                              SELECT value   FROM tinstance_attribute_NS WHERE
                              attribute_id = 38 AND
                              value_at = 'Hipotecarios sobre la vivienda' AND
                              P.instant BETWEEN instant_from AND CASE WHEN
                              instant_to IS NULL THEN GETDATE() ELSE
                        instant_to
                              END)
                        AND P.instant BETWEEN G.instant_from AND CASE WHEN
                        G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to
                        END
                        AND G.provinceId IN (
                              SELECT value   FROM tinstance_attribute_NS
                              WHERE  attribute_id = 54 AND value_at =
                              'Cordoba' AND
                                    P.instant BETWEEN instant_from AND CASE
                              WHEN
                                    instant_to IS NULL THEN GETDATE() ELSE
                                    instant_to END)
                        AND P.amount > 0 AND D.CUIT IS NOT NULL
                  GROUP BY D.CUIT)
      UNIONS GROUP BY col1;

SELECT COUNT(*) col1 FROM STORE_204
```

Tiempo de Ejecución : 00:44:58,390

Traducción (Con optimizacion)

```
No aplica debido a la condición : a.descripcion = 'Hipotecarios sobre la
vivienda'  y
                        p.nombre(P.t) = 'Cordoba'
```

Q9) Consulta: *Listar la distribución de los prestamos por línea de asistencia en la provincia de Buenos Aires al 2004/11/01.*

En TOLAP-QL se escribe:

```
SELECT A.Asistencia, SUM(amount)
  FROM Prestamos P, Deudores D, Asistencias A, Geografia G
 WHERE P.Deudores = D.bottom AND
       P.Asistencias = A.bottom AND
       P.Geografia = G.bottom AND
       RUP(A,Asistencia,P.t) AND
       RUP(D,CUIT,P.t) AND
       RUP(G,provinceId:p,P.t) AND
       p.nombre = 'Buenos Aires' AND
       P.t <= '2004-11-01';
```

Traducción (Sin optimizacion)

```
SELECT col1, SUM(col2) AS col2
      FROM (SELECT A.Asistencia col1, SUM(P.amount) col2
              FROM Prestamos_2 P, DIMENSION_Deudores D, DIMENSION_Asistencias
            A,
                   DIMENSION_Geografia G
              WHERE P.Deudores_CUIT = D.CUIT AND P.Asistencias_Asistencia =
                         A.Asistencia AND P.Geografia_provinceId = G.provinceId
            AND
                         P.instant BETWEEN A.instant_from AND CASE WHEN
            A.instant_to
                         IS NULL THEN GETDATE() ELSE A.instant_to END AND
            P.instant
                         BETWEEN D.instant_from AND CASE WHEN D.instant_to IS
            NULL
                         THEN GETDATE() ELSE D.instant_to END AND P.instant
            BETWEEN
                         G.instant_from AND CASE WHEN G.instant_to IS NULL THEN
                         GETDATE() ELSE G.instant_to END AND G.provinceId IN (
                                    SELECT value   FROM tinstance_attribute_NS
                                    WHERE  attribute_id = 54 AND value_at =
                                          'Buenos Aires' AND P.instant BETWEEN
                                          instant_from AND CASE WHEN instant_to
                                    IS
                                          NULL THEN GETDATE() ELSE instant_to
                                    END)
                         AND P.instant <= '2004-11-01' AND A.Asistencia IS NOT
NULL
              GROUP BY A.Asistencia
            UNION
            SELECT A.Asistencia col1, SUM(P.amount) col2
              FROM Prestamos_3 P, DIMENSION_Deudores D, DIMENSION_Asistencias
            A,
                   DIMENSION_Geografia G
              WHERE P.Deudores_CUIT = D.CUIT AND P.Asistencias_Asistencia =
                         A.Asistencia AND P.Geografia_localidadId =
                     G.localidadId
                         AND P.instant BETWEEN A.instant_from AND CASE WHEN
                         A.instant_to IS NULL THEN GETDATE() ELSE A.instant_to
                     END
                         AND P.instant BETWEEN D.instant_from AND CASE WHEN
                         D.instant_to IS NULL THEN GETDATE() ELSE D.instant_to
                     END
                         AND P.instant BETWEEN G.instant_from AND CASE WHEN
                         G.instant_to IS NULL THEN GETDATE() ELSE G.instant_to
                     END
```

```
                        AND G.provinceId IN (
                                    SELECT value    FROM tinstance_attribute_NS
                                    WHERE  attribute_id = 54 AND value_at =
                                          'Buenos Aires' AND P.instant BETWEEN
                                          instant_from AND CASE WHEN instant_to
                                    IS
                                          NULL THEN GETDATE() ELSE instant_to
                                    END)
                        AND P.instant <= '2004-11-01' AND A.Asistencia IS NOT
NULL
             GROUP BY A.Asistencia)
      UNIONS GROUP BY col1
```

Tiempo de Ejecución : 00:03:59,391

Traducción (Con optimizacion)

No aplica debido a la condición : `p.nombre = 'Buenos Aires'`

Q10) Consulta: *Indicar cual fue el monto total de prestamos brindados en la provincia de Cordoba, en los meses en que en la provincia de Mendoza se presto más de $200.000.*

En TOLAP-QL se escribe:

```
SELECT J.mes AS mes, G.provinceId AS provinceId,
       SUM(amount) AS TotalProvince
  FROM Prestamos P, Geografia G, Tiempo J
 WHERE P.Geografia = G.bottom AND
       RUP(G,provinceId:p,P.t) AND
       P.Tiempo = J.bottom AND
       RUP(J,dia,P.t) AND
       p.nombre = 'Mendoza'
STORE AS Q1;

SELECT G.provinceId AS provinceId, SUM(amount) AS TotalProvince
  FROM Prestamos P, Geografia G, Q1, Tiempo J
 WHERE P.Geografia = G.bottom AND
       RUP(G,provinceId:p,P.t) AND
       P.Tiempo = J.bottom AND
       RUP(J,dia,P.t) AND
       p.nombre = 'Cordoba' AND
       P.amount > 0 AND
       Q1.TotalProvince > 200000 AND
       Q1.mes = J.mes;
```

Traducción (Sin optimización)

```
CREATE VIEW STORE_195 AS
      SELECT mes, provinceId, SUM(TotalProvince) AS TotalProvince
```

```
            FROM (
                  SELECT J.mes AS mes, G.provinceId AS provinceId,
            SUM(P.amount)
                        AS TotalProvince
                    FROM Prestamos_2 P, DIMENSION_Geografia G,
                  DIMENSION_Tiempo J
                    WHERE P.Geografia_provinceId = G.provinceId AND P.instant
                              BETWEEN G.instant_from AND CASE WHEN
                  G.instant_to IS
                              NULL THEN GETDATE() ELSE G.instant_to END AND
                              P.Tiempo_dia = J.dia AND P.instant BETWEEN
                              J.instant_from AND CASE WHEN J.instant_to IS
                  NULL
                              THEN GETDATE() ELSE J.instant_to END AND
                  G.provinceId
                              IN (
                                    SELECT value   FROM tinstance_attribute_NS
                                    WHERE  attribute_id = 54 AND value_at =
                                          'Mendoza' AND P.instant BETWEEN
                                          instant_from AND CASE WHEN instant_to
                              IS
                                          NULL THEN GETDATE() ELSE instant_to
                              END)
                              AND J.mes IS NOT NULL AND G.provinceId IS NOT
NULL
                    GROUP BY J.mes, G.provinceId
                  UNION
                  SELECT J.mes AS mes, G.provinceId AS provinceId,
            SUM(P.amount)
                        AS TotalProvince
                    FROM Prestamos_3 P, DIMENSION_Geografia G,
                  DIMENSION_Tiempo J
                    WHERE P.Geografia_localidadId = G.localidadId AND
                  P.instant
                              BETWEEN G.instant_from AND CASE WHEN
                  G.instant_to IS
                              NULL THEN GETDATE() ELSE G.instant_to END AND
                              P.Tiempo_dia = J.dia AND P.instant BETWEEN
                              J.instant_from AND CASE WHEN J.instant_to IS
                  NULL
                              THEN GETDATE() ELSE J.instant_to END AND
                  G.provinceId
                              IN (
                                    SELECT value   FROM tinstance_attribute_NS
                                    WHERE  attribute_id = 54 AND value_at =
                                          'Mendoza' AND P.instant BETWEEN
                                          instant_from AND CASE WHEN instant_to
                              IS
                                          NULL THEN GETDATE() ELSE instant_to
                              END)
                              AND J.mes IS NOT NULL AND G.provinceId IS NOT
                              NULL
                    GROUP BY J.mes, G.provinceId)
      UNIONS GROUP BY mes, provinceId;


SELECT provinceId, SUM(TotalProvince) AS TotalProvince
      FROM (
            SELECT G.provinceId AS provinceId, SUM(P.amount) AS TotalProvince
              FROM Prestamos_2 P, DIMENSION_Geografia G, STORE_195,
                        DIMENSION_Tiempo J
```

```
  WHERE P.Geografia_provinceId = G.provinceId AND P.instant
BETWEEN
            G.instant_from AND CASE WHEN G.instant_to IS NULL THEN
            GETDATE() ELSE G.instant_to END AND P.Tiempo_dia =
J.dia
            AND P.instant BETWEEN J.instant_from AND CASE WHEN
            J.instant_to IS NULL THEN GETDATE() ELSE J.instant_to
END
            AND G.provinceId IN (
                        SELECT value FROM
                        instance_attribute_NS
                        WHERE  attribute_id = 54 AND value_at
                        =
                              'Cordoba' AND P.instant BETWEEN
                              instant_from AND CASE WHEN
                              instant_to IS NULL THEN
                        GETDATE()
                              ELSE instant_to END)
            AND P.amount > 0 AND STORE_195.TotalProvince > 200000
      AND
            STORE_195.mes = J.mes AND G.provinceId IS NOT NULL
  GROUP BY G.provinceId
UNION
SELECT G.provinceId AS provinceId, SUM(P.amount) AS TotalProvince
  FROM Prestamos_3 P, DIMENSION_Geografia G, STORE_195,
       DIMENSION_Tiempo J
  WHERE P.Geografia_localidadId = G.localidadId AND P.instant
BETWEEN
            G.instant_from AND CASE WHEN G.instant_to IS NULL THEN
            GETDATE() ELSE G.instant_to END AND P.Tiempo_dia =
J.dia
            AND P.instant BETWEEN J.instant_from AND CASE WHEN
            J.instant_to IS NULL THEN GETDATE() ELSE J.instant_to
END
            AND G.provinceId IN (
                        SELECT value FROM
                  tinstance_attribute_NS
                        WHERE  attribute_id = 54 AND value_at
                  =
                              'Cordoba' AND P.instant BETWEEN
                              instant_from AND CASE WHEN
                              instant_to IS NULL THEN
                  GETDATE()
                              ELSE instant_to END)
            AND P.amount > 0 AND STORE_195.TotalProvince > 200000
      AND
            STORE_195.mes = J.mes AND G.provinceId IS NOT NULL
  GROUP BY G.provinceId)
UNIONS GROUP BY provinceId
```

Tiempo de Ejecución : 05:16:03,844

Traducción (Con optimizacion)

No aplica debido a la condición : `p.nombre = 'Mendoza'`

Q11) Consulta: *Indicar la cantidad de personas que tomaron prestamos en Cordoba y Buenos Aires en el mismo mes.*

En TOLAP-QL se escribe:

```
SELECT J.mes AS mes,D.CUIT AS Deudor
  FROM Prestamos P, Geografia G, Tiempo J, Deudores D
 WHERE P.Geografia = G.bottom AND
       RUP(G,provinceId:pr,P.t) AND
       P.Tiempo = J.bottom AND
       RUP(J,dia,P.t) AND
       P.Deudores = D.bottom AND
       RUP(D,CUIT,P.t) AND
       pr.nombre = 'Cordoba' AND
       P.amount > 0
STORE AS Q1;

SELECT COUNT(*)
  FROM Prestamos P, Geografia G, Q1, Tiempo J, Deudores D
 WHERE P.Geografia = G.bottom AND
       RUP(G,provinceId:pr,P.t) AND
       P.Tiempo = J.bottom AND
       RUP(J,dia,P.t) AND
       P.Deudores = D.bottom AND
       RUP(D,CUIT,P.t) AND
       pr.nombre = 'Buenos Aires' AND
       P.amount > 0 AND
       Q1.Deudor = D.bottom AND
       Q1.mes = J.mes;
```

Traducción (Sin optimización)

```
CREATE VIEW STORE_196 AS
      SELECT DISTINCT J.mes AS mes, D.CUIT AS Deudor
            FROM Prestamos_2 P, DIMENSION_Geografia G, DIMENSION_Tiempo J,
                  DIMENSION_Deudores D
            WHERE P.Geografia_provinceId = G.provinceId AND P.instant BETWEEN
                  G.instant_from AND CASE WHEN G.instant_to IS NULL THEN
      GETDATE()
                  ELSE G.instant_to END AND P.Tiempo_dia = J.dia AND P.instant
                  BETWEEN J.instant_from AND CASE WHEN J.instant_to IS NULL
      THEN
                  GETDATE() ELSE J.instant_to END AND P.Deudores_CUIT = D.CUIT
      AND
                  P.instant BETWEEN D.instant_from AND CASE WHEN D.instant_to
      IS
                  NULL THEN GETDATE() ELSE D.instant_to END AND G.provinceId
      IN
                        (SELECT value    FROM tinstance_attribute_NS
                         WHERE  attribute_id = 54 AND value_at = 'Cordoba'
                        AND
                              P.instant BETWEEN instant_from AND CASE WHEN
```

```
                              instant_to IS NULL THEN GETDATE() ELSE
                        instant_to
                          END)
                  AND P.amount > 0 AND J.mes IS NOT NULL AND D.CUIT IS NOT
            NULL
      UNION
      SELECT DISTINCT J.mes AS mes, D.CUIT AS Deudor
            FROM Prestamos_3 P, DIMENSION_Geografia G, DIMENSION_Tiempo J,
                  DIMENSION_Deudores D
            WHERE P.Geografia_localidadId = G.localidadId AND P.instant
      BETWEEN
                  G.instant_from AND CASE WHEN G.instant_to IS NULL THEN
      GETDATE()
                  ELSE G.instant_to END AND P.Tiempo_dia = J.dia AND P.instant
                  BETWEEN J.instant_from AND CASE WHEN J.instant_to IS NULL
      THEN
                  GETDATE() ELSE J.instant_to END AND P.Deudores_CUIT = D.CUIT
      AND
                  P.instant BETWEEN D.instant_from AND CASE WHEN D.instant_to
      IS
                  NULL THEN GETDATE() ELSE D.instant_to END AND G.provinceId
      IN
                        (SELECT value   FROM tinstance_attribute_NS
                         WHERE  attribute_id = 54 AND value_at = 'Cordoba' AND
                              P.instant BETWEEN instant_from AND CASE WHEN
                              instant_to IS NULL THEN GETDATE() ELSE
                  instant_to
                        END)
                  AND P.amount > 0 AND J.mes IS NOT NULL AND D.CUIT IS NOT
                  NULL;


SELECT COUNT(*) col1
      FROM Prestamos_2 P, DIMENSION_Geografia G, STORE_196, DIMENSION_Tiempo
J,
            DIMENSION_Deudores D
      WHERE P.Geografia_provinceId = G.provinceId AND P.instant BETWEEN
            G.instant_from AND CASE WHEN G.instant_to IS NULL THEN GETDATE()
ELSE
            G.instant_to END AND P.Tiempo_dia = J.dia AND P.instant BETWEEN
            J.instant_from AND CASE WHEN J.instant_to IS NULL THEN GETDATE()
ELSE
            J.instant_to END AND P.Deudores_CUIT = D.CUIT AND P.instant
BETWEEN
            D.instant_from AND CASE WHEN D.instant_to IS NULL THEN GETDATE()
ELSE
            D.instant_to END AND G.provinceId IN
                        (SELECT value   FROM tinstance_attribute_NS
                          WHERE  attribute_id = 54 AND value_at = 'Buenos
                        Aires'
                              AND P.instant BETWEEN instant_from AND CASE WHEN
                              instant_to IS NULL THEN GETDATE() ELSE
                        instant_to
                         END)
            AND P.amount > 0 AND STORE_196.Deudor = D.CUIT AND STORE_196.mes =
            J.mes
```

Tiempo de Ejecución : 01:40:14,063

Traducción (Con optimizacion)

No aplica debido a la condición : `pr.nombre = 'Cordoba' y P.amount > 0`

Q12) Consulta: *Indicar para cada mes, la cantidad de personas que tienen deudas en la provincia de Buenos Aires y Cordoba.*

En TOLAP-QL se escribe:

```
SELECT J.mes AS mes, P.Deudores
FROM Prestamos P, Geografia G , Tiempo J
WHERE P.Tiempo = J.bottom
  AND RUP(J,dia,P.t)
  AND P.Geografia = G.bottom
  AND RUP(G, provinceId:pr, P.t) AND pr.nombre = 'Cordoba'
  AND (RUP(G,  provinceId:pr1, P.t) AND
                              pr1.nombre  = 'Buenos Aires')
STORE AS Q1 ;

SELECT Q1.mes, COUNT(*)
  FROM Q1;
```

Traducción (Sin optimización)

```
CREATE VIEW STORE_205 AS
      SELECT DISTINCT J.mes AS mes, P.Deudores_CUIT col2
            FROM Prestamos_2 P, DIMENSION_Geografia G, DIMENSION_Tiempo J
            WHERE P.Tiempo_dia = J.dia AND P.instant BETWEEN J.instant_from
            AND
                  CASE WHEN J.instant_to IS NULL THEN GETDATE() ELSE
            J.instant_to
                  END AND P.Geografia_provinceId = G.provinceId AND P.instant
                  BETWEEN G.instant_from AND CASE WHEN G.instant_to IS NULL
            THEN
                  GETDATE() ELSE G.instant_to END AND G.provinceId IN
                              (SELECT value   FROM tinstance_attribute_NS
                               WHERE  attribute_id = 54 AND value_at =
                              'Cordoba'
                                    AND P.instant BETWEEN instant_from AND CASE
                                    WHEN instant_to IS NULL THEN GETDATE() ELSE
                                    instant_to END)
                  AND EXISTS
                              (SELECT 1 FROM Prestamos_2 P1,
                        DIMENSION_Geografia G
                               WHERE P.instant = P1.instant AND P.instant
                        BETWEEN
                                    G.instant_from AND CASE WHEN G.instant_to
                        IS
                                    NULL THEN GETDATE() ELSE G.instant_to END
                        AND
                                    P1.Geografia_provinceId = G.provinceId AND
                                    G.provinceId IN
```

```
                                        (SELECT value FROM
                                    tinstance_attribute_NS
                                         WHERE  attribute_id = 54 AND
                                    value_at =
                                                'Buenos Aires' AND P.instant
                                                BETWEEN instant_from AND CASE
                                    WHEN
                                                instant_to IS NULL THEN
                                    GETDATE()
                                                ELSE instant_to END)
                               ) AND J.mes IS NOT NULL
      UNION
      SELECT DISTINCT J.mes AS mes, P.Deudores_CUIT col2
            FROM Prestamos_3 P, DIMENSION_Geografia G, DIMENSION_Tiempo J
            WHERE P.Tiempo_dia = J.dia AND P.instant BETWEEN J.instant_from
            AND
                  CASE WHEN J.instant_to IS NULL THEN GETDATE() ELSE
            J.instant_to
                  END AND P.Geografia_localidadId = G.localidadId AND
            P.instant
                  BETWEEN G.instant_from AND CASE WHEN G.instant_to IS NULL
            THEN
                  GETDATE() ELSE G.instant_to END AND G.provinceId IN
                              (SELECT value FROM tinstance_attribute_NS
                               WHERE  attribute_id = 54 AND value_at =
                              'Cordoba'
                                    AND P.instant BETWEEN instant_from AND CASE
                                    WHEN instant_to IS NULL THEN GETDATE() ELSE
                                    instant_to END)
                  AND EXISTS
                              (SELECT 1 FROM Prestamos_3 P1,
                        DIMENSION_Geografia G
                               WHERE P.instant = P1.instant AND P.instant
                        BETWEEN
                                    G.instant_from AND CASE WHEN G.instant_to
                        IS
                                    NULL THEN GETDATE() ELSE G.instant_to END
                        AND
                                    P1.Geografia_localidadId = G.localidadId
                        AND
                                    G.provinceId IN
                                          (SELECT value FROM
                                    tinstance_attribute_NS
                                           WHERE  attribute_id = 54 AND
                                    value_at =
                                                'Buenos Aires' AND P.instant
                                                BETWEEN instant_from AND CASE
                                    WHEN
                                                instant_to IS NULL THEN
                                    GETDATE()
                                                ELSE instant_to END)
                               ) AND J.mes IS NOT NULL;


SELECT STORE_205.mes col1, COUNT(*) col2 FROM STORE_205  GROUP BY
STORE_205.mes
```

Tiempo de Ejecución : 64:13:20,453

Traducción (Con optimización)

No aplica debido a la condición : `pr.nombre = 'Cordoba' y pr1.nombre = 'Buenos Aires'`

Q13) Consulta: *Listar la distribución de deuda por clasificación del deudor para las provincias de la región de Cuyo que han brindado prestamos por más de $200.000.*

En TOLAP-QL se escribe:

```
SELECT G.provinceId AS provinceId, SUM(P.amount) AS TotalProv
  FROM Prestamos P, Geografia G
 WHERE P.Geografia = G.bottom
   AND RUP(G, Region:r, P.t)
   AND r.nombre(P.t) = 'Cuyo'
   AND P.amount > 0
STORE AS Q1;
SELECT P.Situaciones, SUM(P.amount)
  FROM Prestamos P, Geografia G, Q1
 WHERE P.Geografia = G.bottom
   AND RUP(G, provinceId:p, P.t)
   AND p = Q1.provinceId
   AND Q1.TotalProv > 200000
   AND P.amount > 0;
```

Traducción (Sin optimización)

```
CREATE VIEW STORE_200 AS
      SELECT provinceId, SUM(TotalProv) AS TotalProv
            FROM (SELECT G.provinceId AS provinceId, SUM(P.amount) AS
      TotalProv
                        FROM Prestamos_2 P, DIMENSION_Geografia G
                        WHERE P.Geografia_provinceId = G.provinceId AND
                  P.instant
                          BETWEEN G.instant_from AND CASE WHEN G.instant_to IS
                  NULL
                          THEN GETDATE() ELSE G.instant_to END AND G.Region IN
                              (SELECT value   FROM tinstance_attribute_NS
                                    WHERE  attribute_id = 55 AND value_at =
                              'Cuyo'
                                    AND P.instant BETWEEN instant_from AND CASE
                                    WHEN instant_to IS NULL THEN GETDATE() ELSE
                                    instant_to END)
                          AND P.amount > 0 AND G.provinceId IS NOT NULL
                        GROUP BY G.provinceId
                  UNION
                  SELECT G.provinceId AS provinceId, SUM(P.amount) AS
                  TotalProv
                        FROM Prestamos_3 P, DIMENSION_Geografia G
                        WHERE P.Geografia_localidadId = G.localidadId AND
                  P.instant
                          BETWEEN G.instant_from AND CASE WHEN G.instant_to IS
                  NULL
                          THEN GETDATE() ELSE G.instant_to END AND G.Region IN
```

```
                              (SELECT value   FROM tinstance_attribute_NS
                                    WHERE  attribute_id = 55 AND value_at =
                              'Cuyo'
                                    AND P.instant BETWEEN instant_from AND
                              CASE
                                    WHEN instant_to IS NULL THEN GETDATE()
                              ELSE
                                    instant_to END)
                        AND P.amount > 0 AND G.provinceId IS NOT NULL
                      GROUP BY G.provinceId)
      UNIONS GROUP BY provinceId;


SELECT col1, SUM(col2) AS col2 FROM
      (SELECT P.Situaciones_SituacionId col1, SUM(P.amount) col2
            FROM Prestamos_2 P, DIMENSION_Geografia G, STORE_200
            WHERE P.Geografia_provinceId = G.provinceId AND P.instant
                  BETWEEN G.instant_from AND CASE WHEN G.instant_to IS NULL
            THEN
                  GETDATE() ELSE G.instant_to END AND G.provinceId =
                  STORE_200.provinceId AND STORE_200.TotalProv > 200000 AND
                  P.amount > 0
            GROUP BY P.Situaciones_SituacionId
      UNION
      SELECT P.Situaciones_SituacionId col1, SUM(P.amount) col2
            FROM Prestamos_3 P, DIMENSION_Geografia G, STORE_200
            WHERE P.Geografia_localidadId = G.localidadId AND P.instant
      BETWEEN
                  G.instant_from AND CASE WHEN G.instant_to IS NULL THEN
            GETDATE()
                  ELSE G.instant_to END AND G.provinceId =
            STORE_200.provinceId AND
                  STORE_200.TotalProv > 200000 AND P.amount > 0
            GROUP BY P.Situaciones_SituacionId)
UNIONS GROUP BY col1
```

Tiempo de Ejecución : 02:39:18,469

Traducción (Con optimización)

No aplica debido a la condición : `r.nombre(P.t) = 'Cuyo'`

Q14) Consulta: *Para cada mes, listar los deudores que han recibido más de un préstamo de un mismo banco y que además fueron asistidos por más de un banco.*

En TOLAP-QL se escribe:

```
SELECT J.mes AS mes, D.CUIT AS CUIT,  E.EntidadId ,
       COUNT(*) AS CantPrestamos
  FROM Prestamos P, Deudores D,  Tiempo J, EntidadesFinancieras E
 WHERE P.Deudores = D.bottom AND
       RUP(D,CUIT,P.t) AND
```

```
            P.Tiempo = J.bottom AND
            RUP(J,dia,P.t) AND
            P.EntidadesFinancieras = E.bottom AND
            RUP(E,EntidadId,P.t) AND
            P.amount > 0
        STORE AS Q1;

        SELECT Q1.mes AS mes, Q1.CUIT AS CUIT, COUNT(*) AS CantPrestadoras
          FROM Q1
         WHERE Q1.CantPrestamos > 1
        STORE AS Q2;

        SELECT Q2.mes, Q2.CUIT
          FROM Q2
         WHERE Q2.CantPrestadoras > 1 ;
```

Traducción (Sin optimización)

```
CREATE VIEW STORE_206 AS
    SELECT mes, CUIT, col3, SUM(CantPrestamos) AS CantPrestamos
        FROM (
            SELECT J.mes AS mes, D.CUIT AS CUIT, E.EntidadId col3,
        COUNT(*)
                AS CantPrestamos
              FROM Prestamos_1 P, DIMENSION_Deudores D, DIMENSION_Tiempo
            J,
                DIMENSION_EntidadesFinancieras E
              WHERE P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
                D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
                GETDATE() ELSE D.instant_to END AND P.Tiempo_dia =
            J.dia
                AND P.instant BETWEEN J.instant_from AND CASE WHEN
                J.instant_to IS NULL THEN GETDATE() ELSE J.instant_to
            END
                AND P.EntidadesFinancieras_EntidadId = E.EntidadId AND
                P.instant BETWEEN E.instant_from AND CASE WHEN
            E.instant_to
                IS NULL THEN GETDATE() ELSE E.instant_to END AND
            P.amount >
                0 AND J.mes IS NOT NULL AND D.CUIT IS NOT NULL AND
                E.EntidadId IS NOT NULL
              GROUP BY J.mes, D.CUIT, E.EntidadId
            UNION
            SELECT J.mes AS mes, D.CUIT AS CUIT, E.EntidadId col3,
            COUNT(*)
                AS CantPrestamos
              FROM Prestamos_2 P, DIMENSION_Deudores D, DIMENSION_Tiempo
            J,
                DIMENSION_EntidadesFinancieras E
              WHERE P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
                D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
                GETDATE() ELSE D.instant_to END AND P.Tiempo_dia =
            J.dia
                AND P.instant BETWEEN J.instant_from AND CASE WHEN
                J.instant_to IS NULL THEN GETDATE() ELSE J.instant_to
            END
                AND P.EntidadesFinancieras_EntidadId = E.EntidadId AND
                P.instant BETWEEN E.instant_from AND CASE WHEN
            E.instant_to
```

```
                    IS NULL THEN GETDATE() ELSE E.instant_to END AND
                P.amount >
                    0 AND J.mes IS NOT NULL AND D.CUIT IS NOT NULL AND
                    E.EntidadId IS NOT NULL
                  GROUP BY J.mes, D.CUIT, E.EntidadId
                UNION
                SELECT J.mes AS mes, D.CUIT AS CUIT, E.EntidadId col3,
                COUNT(*)
                    AS CantPrestamos
                  FROM Prestamos_3 P, DIMENSION_Deudores D, DIMENSION_Tiempo
                J,
                    DIMENSION_EntidadesFinancieras E
                  WHERE P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
                    D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
                    GETDATE() ELSE D.instant_to END AND P.Tiempo_dia =
                J.dia
                    AND P.instant BETWEEN J.instant_from AND CASE WHEN
                    J.instant_to IS NULL THEN GETDATE() ELSE J.instant_to
                END
                    AND P.EntidadesFinancieras_EntidadId = E.EntidadId AND
                    P.instant BETWEEN E.instant_from AND CASE WHEN
                E.instant_to
                    IS NULL THEN GETDATE() ELSE E.instant_to END AND
                P.amount >
                    0 AND J.mes IS NOT NULL AND D.CUIT IS NOT NULL AND
                    E.EntidadId IS NOT NULL
                  GROUP BY J.mes, D.CUIT, E.EntidadId)
      UNIONS GROUP BY mes, CUIT, col3;

CREATE VIEW STORE_207 AS
      SELECT STORE_206.mes AS mes, STORE_206.CUIT AS CUIT, COUNT(*) AS
            CantPrestadoras
        FROM STORE_206
        WHERE STORE_206.CantPrestamos > 1 GROUP BY STORE_206.mes,
      STORE_206.CUIT;


SELECT DISTINCT STORE_207.mes, STORE_207.CUIT
      FROM STORE_207
      WHERE STORE_207.CantPrestadoras > 1
```

Tiempo de Ejecución : 02:16:49,297

Traducción (Con optimizacion)

No aplica debido a la condición : `P.amount > 0`

Q15) Consulta: *Monto total brindado para la de asistencia de tipo 5 por cada Entidad Financiera.*

En TOLAP-QL se escribe:

```
SELECT E.EntidadId,  SUM(amount)
FROM Prestamos P, EntidadesFinancieras E, Asistencias A
```

```
            WHERE P.EntidadesFinancieras = E.bottom AND
                  P.Asistencias = A.bottom AND
                  RUP(A, Asistencia:'5' ,P.t ) AND
                  RUP(E, EntidadId, P.t );
```

Traducción (Sin optimización)

```
SELECT col1, SUM(col2) AS col2
FROM (SELECT E.EntidadId col1, SUM(P.amount) col2
            FROM Prestamos_1 P, DIMENSION_EntidadesFinancieras E,
                  DIMENSION_Asistencias A
            WHERE P.EntidadesFinancieras_EntidadId = E.EntidadId AND
             P.Asistencias_Asistencia = A.Asistencia AND
             P.instant BETWEEN A.instant_from AND CASE WHEN A.instant_to IS
      NULL
             THEN  GETDATE() ELSE A.instant_to END AND A.Asistencia = '5' AND
             P.instant  BETWEEN E.instant_from AND CASE WHEN E.instant_to IS
            NULL
             THEN GETDATE() ELSE E.instant_to END AND E.EntidadId IS NOT NULL
            GROUP BY E.EntidadId
      UNION
      SELECT E.EntidadId col1, SUM(P.amount) col2
            FROM Prestamos_2 P, DIMENSION_EntidadesFinancieras E,
                  DIMENSION_Asistencias A
            WHERE P.EntidadesFinancieras_EntidadId = E.EntidadId AND
                  P.Asistencias_Asistencia = A.Asistencia AND P.instant
            BETWEEN
                  A.instant_from AND CASE WHEN A.instant_to IS NULL THEN
            GETDATE()
                  ELSE A.instant_to END AND A.Asistencia = '5' AND P.instant
                  BETWEEN E.instant_from AND CASE WHEN E.instant_to IS NULL
            THEN
                  GETDATE() ELSE E.instant_to END AND E.EntidadId IS NOT NULL
            GROUP BY E.EntidadId
      UNION
      SELECT E.EntidadId col1, SUM(P.amount) col2
            FROM Prestamos_3 P, DIMENSION_EntidadesFinancieras E,
                  DIMENSION_Asistencias A
            WHERE P.EntidadesFinancieras_EntidadId = E.EntidadId AND
                  P.Asistencias_Asistencia = A.Asistencia AND P.instant
            BETWEEN
                  A.instant_from AND CASE WHEN A.instant_to IS NULL THEN
            GETDATE()
                  ELSE A.instant_to END AND A.Asistencia = '5' AND P.instant
                  BETWEEN E.instant_from AND CASE WHEN E.instant_to IS NULL
            THEN
                  GETDATE() ELSE E.instant_to END AND E.EntidadId IS NOT NULL
            GROUP BY E.EntidadId)
UNIONS GROUP BY col1
```

Tiempo de Ejecución : 00:04:24,406

Traducción (Con optimización)

```
SELECT col1, SUM(col2) AS col2
      FROM (
            SELECT EntidadesFinancieras_EntidadId col1, SUM(MEASURE_SUM) col2
                  FROM VMQ1_1 WHERE Asistencias_Asistencia = '5'
                  GROUP BY EntidadesFinancieras_EntidadId
            UNION
            SELECT EntidadesFinancieras_EntidadId col1, SUM(MEASURE_SUM) col2
                  FROM VMQ1_2 WHERE Asistencias_Asistencia = '5'
                  GROUP BY EntidadesFinancieras_EntidadId
            UNION
            SELECT EntidadesFinancieras_EntidadId col1, SUM(P.amount) col2
                  FROM Prestamos_3 P WHERE Asistencias_Asistencia = '5'
                  GROUP BY EntidadesFinancieras_EntidadId)
      UNIONS GROUP BY col1
```

Tiempo de Ejecución : 00:02:00,403

Q16) Consulta: *Cantidad de deudores que en el año 2004 disminuyeron su deuda respecto del año 2003.*

En TOLAP-QL se escribe:

```
SELECT D.CUIT AS CUIT, A.Asistencia AS Asistencia,
       SUM(amount) AS TotalAsistencia
  FROM Prestamos P, Asistencias A, Deudores D
 WHERE P.Asistencias = A.bottom
   AND P.Deudores = D.bottom
   AND RUP(D, CUIT, P.t)
   AND RUP(A, Asistencia, P.t)
   AND P.t >= '2003/01/01 00:00:01' AND P.t < '2004/01/01'
STORE AS Q2003;

SELECT D.CUIT AS CUIT, A.Asistencia AS Asistencia,
       SUM(amount) AS TotalAsistencia
  FROM Prestamos P, Asistencias A, Deudores D
 WHERE P.Asistencias = A.bottom
   AND P.Deudores = D.bottom
   AND RUP(D, CUIT, P.t)
   AND RUP(A, Asistencia, P.t)
   AND P.t >= '2004/01/01 00:00:01' AND P.t < '2005/01/01'
STORE AS Q2004;

SELECT Q2004.CUIT
  FROM Q2003, Q2004
 WHERE Q2003.CUIT = Q2004.CUIT
   AND Q2003.Asistencia = Q2004.Asistencia
   AND Q2003.TotalAsistencia > Q2004.TotalAsistencia
STORE AS QDeudores;

SELECT COUNT(*)
  FROM QDeudores;
```

Traducción (Sin optimización)

```
CREATE VIEW STORE_201 AS
      SELECT CUIT, Asistencia, SUM(TotalAsistencia) AS TotalAsistencia
            FROM (
                  SELECT D.CUIT AS CUIT, A.Asistencia AS Asistencia,
            SUM(P.amount)
                        AS TotalAsistencia
                    FROM Prestamos_1 P, DIMENSION_Asistencias A,
                         DIMENSION_Deudores D
                    WHERE P.Asistencias_Asistencia = A.Asistencia AND
                        P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
                        D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
                        GETDATE() ELSE D.instant_to END AND P.instant BETWEEN
                        A.instant_from AND CASE WHEN A.instant_to IS NULL THEN
                        GETDATE() ELSE A.instant_to END AND D.CUIT IS NOT NULL
                  AND
                        A.Asistencia IS NOT NULL
                  GROUP BY D.CUIT, A.Asistencia
                  UNION
                  SELECT D.CUIT AS CUIT, A.Asistencia AS Asistencia,
                  SUM(P.amount)
                        AS TotalAsistencia
                    FROM Prestamos_2 P, DIMENSION_Asistencias A,
                         DIMENSION_Deudores D
                    WHERE P.Asistencias_Asistencia = A.Asistencia AND
                        P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
                        D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
                        GETDATE() ELSE D.instant_to END AND P.instant BETWEEN
                        A.instant_from AND CASE WHEN A.instant_to IS NULL THEN
                        GETDATE() ELSE A.instant_to END AND D.CUIT IS NOT NULL
                  AND
                        A.Asistencia IS NOT NULL
                  GROUP BY D.CUIT, A.Asistencia
                  UNION
                  SELECT D.CUIT AS CUIT, A.Asistencia AS Asistencia,
                  SUM(P.amount)
                        AS TotalAsistencia
                    FROM Prestamos_3 P, DIMENSION_Asistencias A,
                         DIMENSION_Deudores D
                    WHERE P.Asistencias_Asistencia = A.Asistencia AND
                        P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
                        D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
                        GETDATE() ELSE D.instant_to END AND P.instant BETWEEN
                        A.instant_from AND CASE WHEN A.instant_to IS NULL THEN
                        GETDATE() ELSE A.instant_to END AND D.CUIT IS NOT NULL
                  AND
                        A.Asistencia IS NOT NULL
                  GROUP BY D.CUIT, A.Asistencia)
      UNIONS GROUP BY CUIT, Asistencia;


CREATE VIEW STORE_202 AS
      SELECT CUIT, Asistencia, SUM(TotalAsistencia) AS TotalAsistencia
```

```
            FROM (
                  SELECT D.CUIT AS CUIT, A.Asistencia AS Asistencia,
            SUM(P.amount)
                        AS TotalAsistencia
                    FROM Prestamos_1 P, DIMENSION_Asistencias A,
                         DIMENSION_Deudores D
                    WHERE P.Asistencias_Asistencia = A.Asistencia AND
                        P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
                        D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
                        GETDATE() ELSE D.instant_to END AND P.instant BETWEEN
                        A.instant_from AND CASE WHEN A.instant_to IS NULL THEN
                        GETDATE() ELSE A.instant_to END AND D.CUIT IS NOT NULL
                  AND
                        A.Asistencia IS NOT NULL GROUP BY D.CUIT, A.Asistencia
                  UNION
                  SELECT D.CUIT AS CUIT, A.Asistencia AS Asistencia,
                  SUM(P.amount)
                        AS TotalAsistencia
                    FROM Prestamos_2 P, DIMENSION_Asistencias A,
                         DIMENSION_Deudores D
                    WHERE P.Asistencias_Asistencia = A.Asistencia AND
                        P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
                        D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
                        GETDATE() ELSE D.instant_to END AND P.instant BETWEEN
                        A.instant_from AND CASE WHEN A.instant_to IS NULL THEN
                        GETDATE() ELSE A.instant_to END AND D.CUIT IS NOT NULL
                  AND
                        A.Asistencia IS NOT NULL GROUP BY D.CUIT, A.Asistencia
                  UNION
                  SELECT D.CUIT AS CUIT, A.Asistencia AS Asistencia,
                  SUM(P.amount)
                        AS TotalAsistencia
                    FROM Prestamos_3 P, DIMENSION_Asistencias A,
                         DIMENSION_Deudores D
                    WHERE P.Asistencias_Asistencia = A.Asistencia AND
                        P.Deudores_CUIT = D.CUIT AND P.instant BETWEEN
                        D.instant_from AND CASE WHEN D.instant_to IS NULL THEN
                        GETDATE() ELSE D.instant_to END AND P.instant BETWEEN
                        A.instant_from AND CASE WHEN A.instant_to IS NULL THEN
                        GETDATE() ELSE A.instant_to END AND D.CUIT IS NOT NULL
                        AND
                        A.Asistencia IS NOT NULL
                    GROUP BY D.CUIT, A.Asistencia)
      UNIONS GROUP BY CUIT, Asistencia;


CREATE VIEW STORE_203 AS
            SELECT DISTINCT STORE_202.CUIT
              FROM STORE_201, STORE_202
              WHERE STORE_201.CUIT = STORE_202.CUIT AND
                  STORE_201.Asistencia = STORE_202.Asistencia AND
                  STORE_201.TotalAsistencia > STORE_202.TotalAsistencia;


SELECT COUNT(*) col1 FROM STORE_203;
```

Tiempo de Ejecución: 02:24:15,187

Traducción (Con optimización)

```
CREATE VIEW STORE_216 AS
      SELECT Deudores_CUIT AS CUIT, Asistencias_Asistencia AS Asistencia,
             SUM(MEASURE_SUM) AS TotalAsistencia
      FROM VMQ5_2003
      GROUP BY Deudores_CUIT, Asistencias_Asistencia;

CREATE VIEW STORE_217 AS
      SELECT Deudores_CUIT AS CUIT, Asistencias_Asistencia AS Asistencia,
             SUM(MEASURE_SUM) AS TotalAsistencia
      FROM VMQ5_2004
      GROUP BY Deudores_CUIT, Asistencias_Asistencia;

CREATE VIEW STORE_218 AS
      SELECT DISTINCT STORE_217.CUIT
      FROM STORE_216, STORE_217
      WHERE STORE_216.CUIT = STORE_217.CUIT AND
            STORE_216.Asistencia = STORE_217.Asistencia AND
            STORE_216.TotalAsistencia > STORE_217.TotalAsistencia;

SELECT COUNT(*) col1 FROM STORE_218
```

Tiempo de Ejecución : 00:03:11,110

ANEXO 6: VMS UTILIZADAS EN EL CASO DE ESTUDIO.

Definición de VMQ1_1

```
CREATE TABLE VMQ1_1(EntidadesFinancieras_EntidadId
numeric(10),Asistencias_Asistencia numeric(10),MEASURE_COUNT NUMERIC(20,4),
MEASURE_SUM NUMERIC(20,4), MEASURE_AVG NUMERIC(20,4), MEASURE_MAX
NUMERIC(20,4), MEASURE_MIN NUMERIC(20,4))
```

Consulta de llenado:

```
INSERT INTO VMQ1_1 SELECT
FT.EntidadesFinancieras_EntidadId,FT.Asistencias_Asistencia,COUNT(*),
SUM(FT.amount), AVG(FT.amount), MAX(FT.amount), MIN(FT.amount) FROM
Prestamos_1 FT GROUP BY
FT.EntidadesFinancieras_EntidadId,FT.Asistencias_Asistencia
```

Tiempo de generación de la VM : 00:02:01,703

Definición de VMQ1_2

```
CREATE TABLE VMQ1_2(EntidadesFinancieras_EntidadId
numeric(10),Asistencias_Asistencia numeric(10),MEASURE_COUNT NUMERIC(20,4),
MEASURE_SUM NUMERIC(20,4), MEASURE_AVG NUMERIC(20,4), MEASURE_MAX
NUMERIC(20,4), MEASURE_MIN NUMERIC(20,4))
```

Consulta de llenado:

```
INSERT INTO VMQ1_2 SELECT
FT.EntidadesFinancieras_EntidadId,FT.Asistencias_Asistencia,COUNT(*),
SUM(FT.amount), AVG(FT.amount), MAX(FT.amount), MIN(FT.amount) FROM
Prestamos_2 FT GROUP BY
FT.EntidadesFinancieras_EntidadId,FT.Asistencias_Asistencia
```

Tiempo de generación de la VM : 00:02:13,906

Definición de VMQ5_2003

```
CREATE TABLE VMQ5_2003(Deudores_CUIT varchar(11),Asistencias_Asistencia
numeric(10),MEASURE_COUNT NUMERIC(20,4), MEASURE_SUM NUMERIC(20,4),
MEASURE_AVG NUMERIC(20,4), MEASURE_MAX NUMERIC(20,4), MEASURE_MIN
NUMERIC(20,4))
```

Consulta de llenado:

```
INSERT INTO VMQ5_2003 SELECT
FT.Deudores_CUIT,FT.Asistencias_Asistencia,COUNT(*), SUM(FT.amount),
AVG(FT.amount), MAX(FT.amount), MIN(FT.amount) FROM Prestamos_1 FT GROUP BY
FT.Deudores_CUIT,FT.Asistencias_Asistencia
```

Tiempo de generación de la VM : 00:14:19,594

Definición de VMQ5_2004

```
CREATE TABLE VMQ5_2004(Deudores_CUIT varchar(11),Asistencias_Asistencia
numeric(10),MEASURE_COUNT NUMERIC(20,4), MEASURE_SUM NUMERIC(20,4),
MEASURE_AVG NUMERIC(20,4), MEASURE_MAX NUMERIC(20,4), MEASURE_MIN
NUMERIC(20,4))
```

Consulta de llenado:

```
INSERT INTO VMQ5_2004 SELECT
FT.Deudores_CUIT,FT.Asistencias_Asistencia,COUNT(*), SUM(FT.amount),
AVG(FT.amount), MAX(FT.amount), MIN(FT.amount) FROM Prestamos_2 FT GROUP BY
FT.Deudores_CUIT,FT.Asistencias_Asistencia
```

Tiempo de generación de la VM : 00:14:43,156

Definición de VMQ4_2

```
CREATE TABLE VMQ4_2(Situaciones_SituacionId numeric(10),Geografia_provinceId
numeric(10),MEASURE_COUNT NUMERIC(20,4), MEASURE_SUM NUMERIC(20,4),
MEASURE_AVG NUMERIC(20,4), MEASURE_MAX NUMERIC(20,4), MEASURE_MIN
NUMERIC(20,4))
```

Consulta de llenado:

```
INSERT INTO VMQ4_2 SELECT
FT.Situaciones_SituacionId,FT.Geografia_provinceId,COUNT(*), SUM(FT.amount),
AVG(FT.amount), MAX(FT.amount), MIN(FT.amount) FROM Prestamos_2 FT GROUP BY
FT.Situaciones_SituacionId,FT.Geografia_provinceId
```

Tiempo de generación de la VM : 00:02:13,719

Definición de VMQ14_1

```
CREATE TABLE VMQ14_1(EntidadesFinancieras_EntidadId numeric(10),Deudores_CUIT
varchar(11),Tiempo_mes varchar(20),MEASURE_COUNT NUMERIC(20,4), MEASURE_SUM
NUMERIC(20,4), MEASURE_AVG NUMERIC(20,4), MEASURE_MAX NUMERIC(20,4),
MEASURE_MIN NUMERIC(20,4))
```

Consulta de llenado:

```
INSERT INTO VMQ14_1 SELECT
FT.EntidadesFinancieras_EntidadId,FT.Deudores_CUIT,dimension_Tiempo.mes,COUNT(
*), SUM(FT.amount), AVG(FT.amount), MAX(FT.amount), MIN(FT.amount) FROM
Prestamos_1 FT, dimension_Tiempo WHERE FT.Tiempo_dia = dimension_Tiempo.dia
AND instant BETWEEN dimension_Tiempo.instant_from AND CASE WHEN
dimension_Tiempo.instant_to IS NULL THEN GETDATE() ELSE
dimension_Tiempo.instant_to END GROUP BY
FT.EntidadesFinancieras_EntidadId,FT.Deudores_CUIT,dimension_Tiempo.mes
```

Tiempo de generación de la VM : 00:24:40,187

Definición de VMQ14_2

```
CREATE TABLE VMQ14_2(EntidadesFinancieras_EntidadId numeric(10),Deudores_CUIT
varchar(11),Tiempo_mes varchar(20),MEASURE_COUNT NUMERIC(20,4), MEASURE_SUM
NUMERIC(20,4), MEASURE_AVG NUMERIC(20,4), MEASURE_MAX NUMERIC(20,4),
MEASURE_MIN NUMERIC(20,4))
```

Consulta de llenado:

```
INSERT INTO VMQ14_2 SELECT
FT.EntidadesFinancieras_EntidadId,FT.Deudores_CUIT,dimension_Tiempo.mes,COUNT(
*), SUM(FT.amount), AVG(FT.amount), MAX(FT.amount), MIN(FT.amount) FROM
Prestamos_2 FT, dimension_Tiempo WHERE FT.Tiempo_dia = dimension_Tiempo.dia
AND instant BETWEEN dimension_Tiempo.instant_from AND CASE WHEN
dimension_Tiempo.instant_to IS NULL THEN GETDATE() ELSE
dimension_Tiempo.instant_to END GROUP BY
FT.EntidadesFinancieras_EntidadId,FT.Deudores_CUIT,dimension_Tiempo.mes
```

Tiempo de generación de la VM : 00:28:48,219

BIBLIOGRAFIA

[BEKMW04] Bebel,B., Eder,J., Koncilia, C., Morzy, T. and Wrembel, R. (2004) `Creation and management of versions in multiversion data warehouse', Proceedings of SAC, pp. 717-723, Nicosia, Cyprus.

[BSSJ98] Bliujute,R., Saltenis, S., Slivinsjas, G. and Jensen, G. (1998) 'Systematic change management in dimensional data warehousing', Technical Report TR-23, Time Center, Computer Science Department, University of Arizona.

[BTM01] Binh, N.T., Tjoa, A.M and Mangisengi, O. (2001) `Meta Cube-X: An XML Metadata Foundation for Inter-operability Search among Web Data Warehouses', Proceedings of DMDW, Interlaken, Switzerland.

[CT98] Cabibbo, L. and Torlone, R. (1998) `A Logical Approach to Multidimensional Database`s', Proceedings of the 6th International Conference on Extending Database Technology, pp. 253-269, Valencia, Spain.

[EKM02] Eder, J., Koncilia, C. and Morzy, T. (2002) `The COMET Metamodel for Temporal Data Warehouses', Proceedings of CAiSE, pp. 83-99, Toronto, Canada.

[HBH03] H¨ummer, W., Bauer, A. and Harde, G. (2003), `XCube: XML for data warehouses', Proceedings of DOLAP, pp. 33-40, New Orleans, LA.

[Kim96] R Kimball, R. (1996) The data warehouse toolkit, J.Wiley and Sons Inc., NYC, NY.

[Le98] Lehner, W. (1998) `Modeling Large OLAP Scenarios', Proceedings of the 6th International Conference on Extending Database Technology, pp. 153-157, Valencia, Spain.

[MACM01] Marian, A., Abiteboul, S., Cobena, G. and Mignet (2001) `Change-Centric Management of Versions in an XML Warehouse', Proceedings of the 27th Int. Conf. on Very Large Databases, pp. 581-590, Rome, Italy.

[Mit03] Mitra, N. (Ed.) (2003) SOAP Version 1.2 Part 0: Primer, W3C Recommendation 24 June 2003.

[Ped99] Pedersen,T.B & Jensen,C. (1999) 'Multidimensional Data Modeling for Complex Data', Proceedings of 1999 Int. Conference on Data Engineering(IEEE/ICDE'99), 336-345, Sidney, Australia.

[MV00] Mendelzon, A.O. and Vaisman, A. (2000) `Temporal queries in OLAP', Proceedings of the 26th Int. Conf. on Very Large Databases,pp. 242-253, Cairo, Egypt.

[MV03] Mendelzon, A.O. and Vaisman, A. (2003) Time in multidimensional databases, Multidimensional Databases: Problems and Solutions, Idea Group, Hershey, PA.

[NNNT03] Niemi, T., Niinim¨aki, M., Nummenmaa, J. and Peter Thanisch (2003) `Applying Grid Technologies to XML Based OLAP Cube Construction', Proceedings of DMDW, Berlin, Germany.

[Sno95] Snodgrass, R. (1995) The TSQL2 temporal query language, Kluwer Academic Publishers, Boston, MA.

[SOAP] W3C SOAP Specifications, http://www.w3.org/TR/soap

[TSMBM06] Thompson, H.S., Sperberg-McQueen, C.M., Mendelsohn,N., Beech, D. and Maloney, M. (2006) `XML Schema 1.1 Part 1: Structures', W3C Working Draft, 30 March, 2006.

[Vai01] Vaisman, A. (2001) `Updates, View Maintenance and Time Management in Multidimensional Databases', Phd Thesis, http://www.cs.toronto.edu/ avaisman/publications.

[VIK106] Alejandro A. Vaisman, Adrian Izquierdo, Marcelo Ktenas. Web-enabled Temporal OLAP. LA-WEB, Puebla, Mexico, 2006

[VIK206] Alejandro A. Vaisman, Adrian Izquierdo, Marcelo Ktenas. A Web-based Architecture for Temporal OLAP. International Journal of Web Engineering and Technology on Data Warehousing In Web, Mobile, and Wireless Environments. En prensa.

[VBR03] Vrdoljak,B., Banek, M. and Rizzi, S. (2003) `Designing Web Warehouses from XML Schemas', Proceedings of DaWaK, pp. 89-98, Prague, Czech Republic.

[XMLA] XML for Analysis, http://xmla.org

[Xylm] Xyleme Project, http://www.xyleme.com