Tesis de Licenciatura en Ciencias de la Computación

Agregación Espacial: Resolución de Consultas OLAP sobre Sistemas de Información Geográfica

Ariel Escribano

Noviembre de 2006

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Director:
Alejandro Ariel Vaisman

Índice

Resumen

La presente tesis describe una implementación de la integración de sistemas de Información Geográfica (GIS) con sistemas OLAP. El modelo implementado está formado por información de aplicación, contenida en un data warehouse, e información geográfica, básicamente un mapa con elementos geométricos almacenados en diferentes capas. La información geométrica que compone el mapa se organiza en un conjunto de jerarquías que conforman lo que se denomina una dimensión GIS. Adicionalmente, los elementos de la parte geométrica se relacionan con información de aplicación almacenada externamente, en general en un data warehouse. En base a este modelo, podremos resolver consultas que involucren agregación o sumarización de ambos tipos de información. Estas consultas se expresan tanto sobre la parte geométrica como sobre la información de aplicación, permitiendo navegar a posteriori los resultados con herramientas OLAP.

La implementación desarrollada permite resolver cuatro tipos de consultas diferentes: (a) Consultas integradas GIS-OLAP, de la forma "Obtener el total de ventas por producto, de todas las sucursales que se encuentran en ciudades cruzadas por un río". Esta consulta involucra los distintos tipos de información, y su resultado puede luego explotarse utilizando herramientas OLAP tradicionales. (b) Consultas de Agregación Geométrica, como "total de sucursales en estados con más de tres aeropuertos", que sumariza valores asociados a geometrías. (c) consultas típicas de sistemas GIS. (d) Consultas OLAP tradicionales.

Cabe destacar que, en la actualidad, no existe ningún software comercial que implemente un modelo OLAP espacial como el propuesto en esta tesis, ni tampoco existe una solución para resolver consultas de agregación geométrica o que integre la información almacenada en sistemas GIS y OLAP.

Para la resolución de consultas de agregación geométrica se implementó un proceso de precálculo de información que se ejecuta en forma off-line. Este proceso calcula la denominada "subpoligonización" del mapa, dando origen a un conjunto de subgeometrías usadas para reorganizar la dimensión GIS y para dividir los valores de los hechos de los componentes geométricos originales en objetos más pequeños (subgeometrías), que pueden compartirse entre diferentes componentes. En el proceso de precálculo se computa la superposición u overlay de las capas que contienen los componentes geométricos que componen el mapa. Este método toma la idea de las técnicas de materialización de vistas, utilizadas tradicionalmente en bases de datos relacionales, y en optimización de consultas en OLAP. En esta tesis comparamos, además, la eficiencia del precómputo del overlay de las capas, con los métodos tradicionalmente utilizados en las herramientas GIS, basados en índices multidimensionales, como R-Trees en sus diferentes variantes.

La implementación fue realizada en lenguaje Java, utilizando tecnologías open source para OLAP y GIS (como Mondrian, JUMP y PostGIS).

Abstract

This thesis describes an implementation for integrating Geographic Information Systems (GIS) with OLAP systems. The implemented model consists of application data contained in a data warehouse, and geographic data (a map with geometric elements stored in different layers). Geometric information that composes the map is organized in a set of hierarchies that conforms a GIS dimension. Additionally, elements from the geometric section are related to externally stored application data (usually, a data warehouse). Based on this model, we can answer queries aggregating both kinds of data. These queries are expressed over the geometric and application parts, allowing to navigate the query results using OLAP tools.

Our implementation supports four types of queries: (a) GIS-OLAP integrated queries, like "Give me the total sales per product in all stores located in cities crossed by a river". This query involves different types of data, and its result can be exploited using traditional OLAP tools. (b) Geometric aggregation queries, like "number of stores in states with more than three airports", that summarizes values associated to geometries. (c) Typical GIS queries. (d) Typical OLAP queries.

It is important to emphasize that nowadays there is no commercial software that implements an OLAP spatial model like the one proposed in this thesis, neither exists a solution that solves geometric aggregation queries or allows GIS and OLAP data integration.

An off-line executed preprocessing stage was implemented for solving geometric aggregation queries. This process generates the so-called "subpolygonization" of the map, creating a set of subgeometries used for reorganizing the GIS dimension and for splitting original geometric components and their associated fact values into smaller objects (subgeometries) which can be shared between different components. The overlay of map layers that contains geometric components is also computed during this preprocessing stage. The idea of this method is borrowed from view materialization techniques traditionally used in relational data bases and OLAP query optimization. Furthermore, in this thesis we compare layer overlay "materialization" efficiency with traditional GIS tools methods, based in multidimensional indexes like R-Trees and its variations.

The implementation was developed using Java programming language and open source technology for OLAP and GIS (such as Mondrian, JUMP and PostGIS).

1 Introducción

Un Sistema de Información Geográfica, o **GIS** (por Geographic Information System) es un sistema que permite la creación, almacenamiento, análisis y manejo de información espacial, representada por geometrías agrupadas en capas (layers), junto con atributos relacionados. Un sistema de este tipo permite integrar, almacenar, editar, analizar y mostrar información geográfica. La Figura 1 muestra un ejemplo de una aplicación GIS llamada ArcGis[1], donde se muestra el mapa de una parte de Europa.

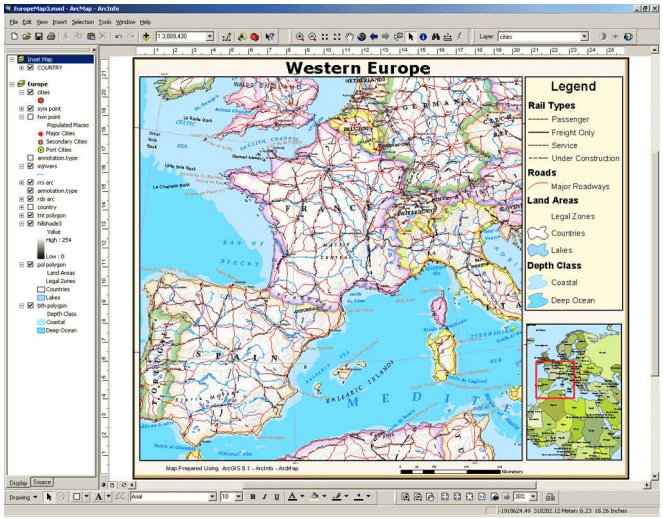


Figura 1: Vista de software ArcGIS.

OLAP (On Line Analytical Processing) es un conjunto de herramientas y algoritmos que apuntan a brindar una respuesta rápida a consultas de tipo analítico. Una base de datos configurada para OLAP utiliza un modelo multidimensional que facilita la ejecución de estas consultas analíticas. La información OLAP está organizada, generalmente, en forma de dimensiones (en tablas de dimensión) y valores que representan hechos, representados como tablas de hechos (fact tables). Las aplicaciones típicas de OLAP están relacionadas con reporte de ventas, marketing, management, reportes financieros, etc., e involucran casi siempre operaciones de sumarización. Consultas típicas son de la forma "total de ventas semanales por región".

1.1 Motivación

En la Figura 2 se muestra un mapa compuesto por cuatro layers: uno que contiene un bounding box [2] rectangular (gris), otro con dos polígonos que representan estados (verde), un tercer layer con una polilínea que representa un río (azul). Finalmente, existe un layer con cinco puntos que representan sucursales (negro).

[1] El ejemplo fue bajado de internet de la dirección:
http://www.esri.com/software/arcgis/arcview/graphics/av_qualitymapping.gif
[2] Un bounding box es una figura que determina el contorno de un mapa, es decir, la región geográfica que dicho mapa abarca.

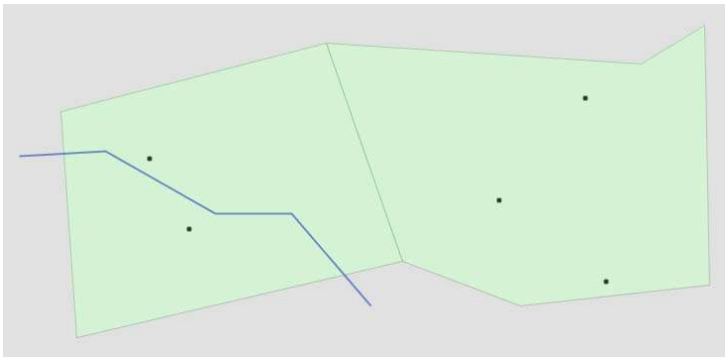


Figura 2: Mapa con cuatro layers.

En general, los sistemas GIS permiten solo un número limitado de agregaciones de medidas geográficas (p. ej., distancia y área). Por otra parte, las herramientas OLAP actuales no soportan interacción con información espacial. Como consecuencia de esto, no existe una única solución suficientemente flexible e integradora que permita realizar consultas que abarquen información geográfica y multidimensional. Las siguientes consultas (basadas en una zona seleccionada sobre el mapa de la Figura 2) sirven de ejemplo de lo anterior:

1- "Sucursales que se encuentran en estados cruzados por ríos" dentro de la zona.
2- "Cantidad de unidades vendidas y dinero facturado, por sucursal, por producto". Los resultados deberán ser "navegables" con herramientas OLAP (por ejemplo, roll-up de sucursales a ciudades que las contienen).
3- "Área y población estimada para cada estado" dentro de la zona (asumiendo que los estados tienen asociado el total de población).
4- Consultar los estados como en el ejemplo anterior, pero con la condición adicional de que sólo se tomen en cuenta los "estados cruzados por al menos un río".
5- "Cantidad de ventas en las sucursales" dentro de la zona seleccionada y navegar el resultado, obteniendo el total de ventas y detalle de ventas por productos a través del tiempo durante el último año.

Las consultas 1 y 2 corresponden a consultas GIS y consultas OLAP, respectivamente. A las consultas del tipo de la 3 (agregación geométrica y agregación de medidas asociadas a las geometrías) y de la 4 (agregación geométrica y de medidas, integradas con consulta espacial) las llamamos consultas de agregación geométrica. Finalmente, a consultas del tipo de la 5 (que integra información espacial con información OLAP) las denominamos consultas GIS-OLAP.
Las consultas GIS u OLAP pueden ser resueltas por sistemas GIS o sistemas OLAP independientes, mientras que no sucede lo mismo con las consultas de agregación geométrica o las consultas GIS-OLAP, que requieren de sistemas que integren ambas soluciones. Ello requiere, previamente, un modelo de datos que soporte este tipo de consultas. En este trabajo nos basaremos en el modelo introducido en [4]**,** que utilizaremos en nuestra implementación.

1.2 Contribución

La solución propuesta se denomina Piet[3] y está construida sobre los conceptos y definiciones introducidos en [4] y que se resumen y discuten en la Sección 2. Piet resuelve consultas de agregación geométrica a partir de la denominada subpoligonización de los layers del mapa, que transforma una dimensión GIS inicial a partir de la descomposición de las geometrías originales en subgeometrías, la asociación de estas subgeometrías a los hechos, y del precálculo de la superposición de los componentes geométricos del mapa. En particular, la ejecución de consultas geométricas se resuelve utilizando el precálculo del overlay (en reemplazo de técnicas clásicas como R-Trees [3]), mientras que las consultas OLAP puras son ejecutadas sobre un motor OLAP externo (Mondrian [15]). Finalmente, las consultas GIS-OLAP, que involucran tanto la parte geométrica como

[3] Se denominó Piet a la solución por ser el nombre de pila del pintor Mondrian, que da nombre al motor OLAP utilizado en la solución.

la analítica (de aplicación), son ejecutadas utilizando la información precalculada e información adicional para relacionar datos de la parte geométrica con datos de aplicación.

El proceso de precálculo offline (**proceso de precálculo de información)** que genera la ejecución de consultas en Piet, puede realizarse en forma gráfica (mediante una interfaz que se anexa al sistema JUMP [24]), o través de interfases que soportan consultas escritas en un lenguaje que llamamos GISOLAP-QL. Este lenguaje utiliza sintaxis MDX [13] para expresar consultas OLAP y una sintaxis desarrollada especialmente para soportar consultas espaciales. La respuesta a las consultas incluye facilidades de navegación al estilo OLAP para las consultas de tipo multidimensionales.

1.3 Organización del documento

En la Sección 2 se comentan los antecedentes y trabajos relacionados con el problema a tratar, y se describe el modelo de datos y marco teórico en el que se basa la implementación. En la Sección 3 se presenta la arquitectura conceptual y la arquitectura implementada, mientras que la Sección 4 discute la implementación de la arquitectura. El detalle de los algoritmos y procedimientos más importantes se presenta en la Sección 5. La Sección 6 presenta los experimentos realizados y discute sus resultados, concluyendo en la Sección 7.

2 Marco teórico

En esta sección se introduce el marco teórico sobre el que se basa la solución propuesta, comentando también el trabajo relacionado existente en el área.

2.1 Trabajo Relacionado

Aunque algunos autores han mencionado los beneficios de combinar GIS y OLAP, no se ha realizado mucho trabajo en este campo.
En lo que respecta a un modelo teórico, Miquel [14] propone utilizar OLAP para integrar data sets de estructuras diferentes, obtenidos a través de diferentes observaciones a lo largo del tiempo. Rivest [23] introduce el concepto de SOLAP (por Spatial OLAP), y describe las propiedades y herramientas que un sistema SOLAP debería tener. Sin embargo, no presentan un modelo formal al respecto. Han [5] utilizó técnicas OLAP para materializar objetos espaciales seleccionados, proponiendo un "Spatial Data Cube" que permite realizar únicamente agregaciones sobre dichos objetos espaciales. Pedersen y Tryfona [19] propusieron pre-agregación de hechos espaciales. Primero pre-procesaban estos hechos, computando las partes disjuntas para poder agregarlas luego. Sin embargo, ignoraron completamente la parte geométrica y no incluyeron tipos de geometrías adicionales a polígonos. Por lo tanto, consultas como "Dar el total de población en ciudades cruzadas por ríos" no puede ser consultado en este modelo. Rao [19, 23] combina OLAP con GIS para consultar spatial data warehouses utilizando R-Trees para acceder a la información de las tablas de hechos. Luego se evalúa el data warehouse de la manera clásica OLAP. Por lo tanto el modelo aprovecha las ventajas provistas por las jerarquías OLAP para localizar información en el R-Tree que indexa la tabla de hechos. En este caso, aunque las medidas no son objetos espaciales, también ignoran la parte geométrica, limitando el espectro de consultas que pueden resolverse con el modelo. Se asume que existe alguna tabla de hechos que contiene los identificadores de objetos espaciales. Además, estos objetos sólo pueden ser puntos, lo cual es poco realista en un ambiente GIS, donde tipos diferentes de objetos aparecen en diferentes layers.
Papadias [17, 18] trabajó en el área de indexar data warehouses espaciales y espacio temporales, aunque esto no tiene relación con el trabajo presentado en esta tesis.
Finalmente, Kuper y Scholl [11] sugirieron el probable aporte de técnicas de constraint sobre base de datos en GIS. Sin embargo, no consideraron agregaciones espaciales ni técnicas OLAP.

2.2 Modelo conceptual

El modelo de datos en el que se basa el trabajo, es el presentado en [4]. En esta sección presentamos sus principales características.

El modelo de representación de geometrías utilizado es el modelo vectorial, en el que la información de cada layer está compuesta por un conjunto finito de tuplas de la forma "(geometría, atributos)", donde la geometría puede ser un punto, una polilínea o un polígono.

Una dimensión GIS se define como un conjunto de jerarquías, organizado como es usual, en esquemas e instancias. El esquema de una dimensión GIS es un conjunto de jerarquías donde cada jerarquía describe un layer o capa de información geométrica. Un nivel dentro de la jerarquía corresponde a un tipo de geometría (punto, línea, etc.). En la Figura 3 se muestra un esquema de dimensión GIS, identificando los 3 sectores que lo componen: geométrico, algebraico y de aplicación.

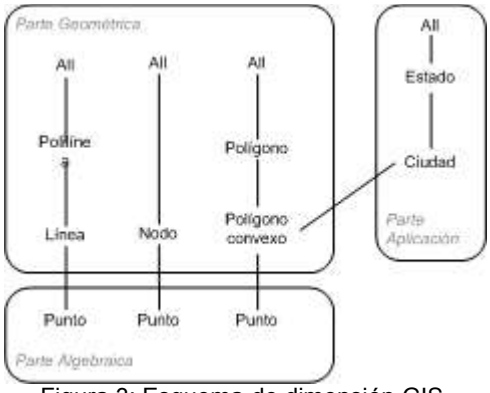


Figura 3: Esquema de dimensión GIS.

La parte geométrica contiene información en forma de un número finito de geometrías que componen el mapa y se utilizará para resolver la sección geométrica de una consulta (p.ej., encontrar todos los polígonos que forman un país). Los elementos de la parte geométrica pueden tener asociados facts (hechos) representados a través de los atributos. Los facts podrán estar cuantificados por un valor, denominado medida (measure).
La parte algebraica está compuesta por el nivel inferior de todas las jerarquías del esquema de dimensión GIS, denominada "Punto". En esta sección la información se representa como conjuntos infinitos de puntos asociados a cada layer. Las geometrías de la parte algebraica están representadas por funciones algebraicas que describen el conjunto infinito de puntos representados por la geometría. Estas formulas están incluidas en las funciones de roll-up que implementan las dependencias funcionales desde el nivel "Punto" hacia las geometrías de la parte geométrica, permitiendo "subir" por las jerarquías de la dimensión GIS.
Los niveles de la parte geométrica pueden estar asociados con conceptos de aplicación, como polígonos con ciudades o polilíneas con ríos, que tendrán una estructura OLAP clásica.

Sea **L** un conjunto de nombres de layers, **A** un conjunto de nombres de atributos y **G** un conjunto de geometrías. Cada elemento a de **A** tiene asociado un conjunto de valores dom(a). **G** contiene las siguientes geometrías: punto, nodo, línea, polilínea, polígono convexo, polígono y el elemento distinguido "All". Cada geometría g de **G** tiene un dominio asociado dom(g).

Definición 1: Esquema de una dimensión GIS.
Un esquema de dimensión GIS es un conjunto finito de jerarquías G_{sch} = {HG1 ,…HGk}
Para un conjunto de geometrías $G \subseteq$ **G**, una jerarquía HG se define como:

(i) Existe una relación $\leq$ sobre los niveles en HG tal que $\leq^*$, su clausura transitiva y reflexiva, es un orden parcial; si g_i y g_j son niveles en HG, y $g_i \leq^* g_j$, existe una dependencia funcional $g_i \rightarrow g_j$;
(ii) Punto $\leq^*$ g, para todo g $\in$ HG; entonces, hay una dependencia funcional desde Punto a todos los niveles en la jerarquía, y Punto es el nivel inferior único en HG;
(iii) Existe un miembro distinguido "All" al tope de la jerarquía;
(iv) No hay ejes redundantes en HG;
(v) Existe una función total AtG, que va de A $\rightarrow$ G tal que, para cada atributo a $\in$A, AtG(a) = g, denotando que g es el nivel (la geometría) a la que pertenece el atributo.

Definición 2: Instancia de una dimensión GIS.
Dado un esquema de dimensión GIS G_{sch} = {H_{G1}, …H_{Gk}}, una instancia de dimensión GIS es una tupla (G_{sch}, F, A); donde F es un conjunto de funciones de la forma $f^{gj\ gk}{}_{Gi}$ de dom(g_j) $\rightarrow$ dom(g_k), que corresponde a cada par de niveles talque $g_j \leq g_k$. Llamamos función de roll-up a cada función de la forma $f^{gj\ gk}{}_{Gi}$ en F. Adicionalmente, dados dos niveles de dimensión gl_1,gl_k, tal que $gl_1 \leq^* gl_k$, y si existe más de un camino desde gl1 a gl_k, las funciones de roll-up en F son las mismas, sin importar el camino tomado (p.ej., $f^{g1\ gn}{}_{Gi} \circ_{\ldots} \circ f^{gj\ gk}{}_{Gi} = f^{g1\ gm}{}_{Gi} \circ_{\ldots} \circ f^{gr\ gk}{}_{Gi}$).
Finalmente, para cada par (g, a) $\in$ G_i x A talque At_{Gi}(a) = g, existe una función $\alpha^{ag}{}_{Gi} \in$ A de dom(a) $\rightarrow$ dom(g).

Definición 3: Tabla de hechos GIS.
Dada una geometría g en una jerarquía H_G de un esquema de dimensión GIS G_{sch} y una lista de medidas M = (M_1, …, M_k), una tabla de hechos GIS es una tupla FT = (g, M). Llamaremos Tabla de hechos básica a una

tupla BFT = (Punto, M). Una instancia de una tabla de hechos GIS es una función que mapea valores en dom (g) a valores en $dom(M_1)x\ldots x\ dom(M_k)$: la instancia de una tabla de hechos básica mapea valores en $\Re^2$ x L a valores en $dom(M_1)$ x … x $dom(M_k)$.

Definición 4: Agregación geométrica.
Dada una dimensión GIS como se describe en las definiciones 1 y 2, una agregación geométrica es una expresión de la forma

$$\iint_C h(x, y)\, dx\, dy,$$

donde $C = \{(x, y) \in \Re^2 \mid \varphi(x, y)\}$.
φ es una fórmula en un multi-sorted logic $\mathcal{L}$ sobre $\Re$, $\mathcal{L}$ y A. El vocabulario de $\mathcal{L}$ contiene los nombres de las funciones que parecen en $\mathcal{F}$ y $\mathcal{A}$, junto con las funciones binarias + y x en reales, los predicados binarios < en reales y las constantes reales 0 y 1. Adicionalmente, constantes en layers y atributos pueden aparecer en el vocabulario de $\mathcal{L}$. Fórmulas atómicas en $\mathcal{L}$ se combinan con los operadores lógicos estándar: $\wedge$, $\vee$, $\neg$ y cuantificadores existenciales y universales sobre variables reales y variables de atributos. Además, h es una función integrable construida sobre elementos de $\mathcal{F} \cup \mathcal{A} \cup \{1, f_t\}$ utilizando composición, producto y adición.

Ejemplo 1: Supongamos la consulta "Población total de las ciudades cruzadas por el río Danubio".

$$Q_{Top} \equiv \iint_C ft^{pobl}(x, y, Lc)\, dx\, dy, \text{ donde}$$
$$C = \{(x, y) \in \Re^2 \mid (\exists x')(\exists y')(\exists x'')(\exists y'')(\exists_c \in dom(NCiudad))$$
$$(\alpha_{Lr}^{NRio,Polilínea}(\text{`Danubio'}) = f_{Lr}^{Punto,Polilínea}(x', y', Lr) \wedge$$
$$\alpha_{Lr}^{NCiudad,Polilínea}(c) = f_{Lc}^{Punto,Polígono}(x', y', Lc)) \wedge$$
$$\alpha_{Lr}^{NCiudad,Polígono}(c) = f_{Lc}^{Punto,Polígono}(x, y, Lc))\}$$

En el Ejemplo 1 se puede ver que, al utilizar la Definición 4 para el cálculo de agregaciones geométricas, surgen dos puntos que puede ser muy costoso resolver: debemos recurrir al nivel base de la jerarquía ("Punto") para armar el área "C" donde aplicar la integral y además, como consecuencia de esto, tendremos que calcular la integral teniendo en cuenta valores de la función ft^{pobl} asociados a cada uno de estos puntos.

Evaluación de consultas de agregación geométrica
Se define la clase de consultas sumables como la formada por aquellas consultas en las se puede evitar el cálculo de la integral de h(x,y) de la Definición 4, permitiendo una evaluación más eficiente de la consulta, a costa de perdida de precisión, ya que las medidas estarán asociadas a componentes geométricos y no al nivel inferior de la jerarquía ("Punto").

Definición 5: Consultas sumables.
Una consulta de agregación geométrica $Q = \iint_C h(x, y)\, dx\, dy$ es sumable si y sólo el conjunto condición C define un conjunto finito de elementos de alguna geometría g y Q puede reescribirse como:

$$\sum_{g \in C} h'(g),$$

donde h' es construida a partir de elementos de $\mathcal{F} \cup \mathcal{A} \cup \{1, f_t\}$ utilizando composición, producto y adición.

Ejemplo 2: La consulta del Ejemplo 1 es sumable, y puede expresarse como:

$$Q_{Top} \equiv \sum_{g_{id} \in C'} ft'^{pobl}(g_{id}, L_c)\ ft^{area}(g_{id}, L_c)$$

Podemos observar que C' devuelve un conjunto finito de identificadores de polígonos (en vez de puntos, como el ejemplo anterior), ft'^{pobl} mapea polígonos con densidad de población por área y ft^{area} mapea polígonos con áreas. Esto permite evitar el cálculo de la integral sobre la función h(x,y) de la Definición 4.

Operación de overlay (superposición)
Muchas consultas en GIS incluyen operaciones entre geometrías de distintos layers (intersección, unión, etc.) que requieren su overlay (superposición) para su resolución.

Definición 6: Overlay de geometrías.
Definimos el overlay de un par de geometrías g_1 y g_2 como el conjunto de geometrías O, donde cada geometría o_i perteneciente a O está incluida tanto en g_1 como en g_2. Las geometrías o_i podrán ser puntos, polilínea y polígonos.

Siguiendo la Definición 1, se puede observar que el cálculo del overlay no es una operación sencilla ya que para resolverlo se tendría que visitar la parte algebraica de la dimensión.

Partiendo de un bounding box definimos los siguientes conceptos que permitirán, entre otras cosas, un cálculo eficaz de la operación de overlay.

Definición 7: Carrier set de un layer.
El carrier set C_{pl} de una polilínea pl consiste en todas las líneas que comparten infinitamente tantos puntos con la polilínea como sea posible, junto con las dos líneas que pasan por los extremos de pl en forma perpendicular a los segmentos de los extremos de pl. El carrier set C_{pg} de un polígono pg es el conjunto de líneas que comparten infinitamente tantos puntos como sea posible con los límites de pg. Finalmente, el carrier set C_p de un punto p consiste de las líneas horizontal y vertical que se intersecan con p.
El carrier set C_L de un layer L es la unión de los carrier sets de puntos, polilíneas y polígonos contenidos en L. A cada línea que forma parte de un carrier set la denominaremos carrier line.

Definición 8. Poligonización convexa de un layer.
Sea C_L el carrier set del layer L, y sea B un bounding box. El conjunto de polígonos abiertos convexos, segmentos de líneas abiertos y puntos inducidos por C_L que están estrictamente dentro de B se denomina poligonización convexa de L, y se nota CP(L).

Definición 9. Subpoligonización y subgeometrías.
Dados dos layers L_1 y L_2 y sus carrier sets C_{L1} y C_{L2} ; la subpoligonización de L_1 de acuerdo a L_2, denotada CSP (L_1, L_2) es un refinamiento de la poligonización convexa de L_1, computada particionando cada polígono convexo abierto y cada segmento de línea abierto en él, con los carriers de C_{L2} . Llamaremos subnodos, subpolígonos y sublíneas a los puntos, polígonos convexos abiertos y segmentos de línea abierto generados, y subgeometrías a su conjunto.

El uso de la subpoligonización para el precálculo del overlay permite mantener la coherencia en las geometrías superpuestas (si utilizáramos directamente las geometrías que se superponen para precalcular el overlay, podría pasar que me queden geometrías inválidas, como polígonos a los que le falta una línea correspondiente a otro layer) y puede brindarnos información adicional al overlay (por ejemplo, al analizar la superposición de un río que nace dentro de un estado, no sólo sabremos si se superponen o no, sino que además contaremos con el punto exacto donde se origina el río.

Transformación de dimensión GIS usando la subpoligonización
Basándonos en las definiciones anteriores, podemos modificar la dimensión GIS utilizando la información creada durante la subpoligonización, como se muestra en la Figura 4. Para ello, usamos el concepto de actualización de dimensión, y el conjunto de operadores introducido en [7].

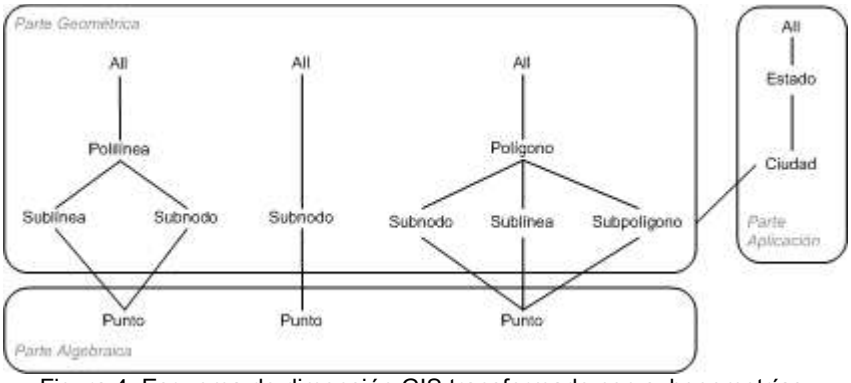


Figura 4: Esquema de dimensión GIS transformado con subgeometrías.

En la figura se puede ver como se agregó un nuevo nivel a la jerarquía de la Figura 3, donde se incluyen las subgeometrías relacionadas con las geometrías superiores (en la jerarquía), generadas durante la subpoligonización. Llamaremos a este proceso de transformación del esquema GIS **asociación de subgeometrías**.

Adicionalmente deberá existir un proceso de **propagación de medidas**, donde se asignará a cada subgeometría del nuevo nivel el valor proporcional de las medidas de las geometrías del mapa a la que se asocia.

Evaluación de consultas OLAP espaciales usando subpoligonización
Las herramientas de consultas en sistemas GIS permiten, entre otras cosas, superponer distintos layers de un mapa para obtener objetos de interés en un área. Para lograr esto, suelen utilizar una combinación de estructuras especiales combinadas con métodos de indexación como R-Trees. Basándonos en el concepto de materialización de vistas utilizado en OLAP para lograr optimizaciones en consultas [6]**,** proponemos precalcular el overlay de los layers de un mapa previo a la ejecución de cualquier consulta geométrica. Se puede demostrar que el efecto del uso de esta información precalculada, sobre consultas sumables que incluyen el overlay de información de distintos layers hace que dichas consultas pueden evaluarse sin acceder a la parte algebraica y además permite almacenar las hechos asociados a las medidas en el nivel más bajo de la parte geométrica (subgeometrías).

Ejemplo 3: Obtener el total de aeropuertos en Seattle.

$$Q_{Overlay} = \sum_{g_{id} \in C_O} 1$$

$$C_O = \{ g_{id} \in G_{id} \mid (\exists a \in dom(NAeropuerto)) (\alpha_{La}^{NAeropuerto,Node}(a) = (g_{id.}, L_a)) \wedge (\alpha_{Lc}^{NCiudad,Polígono}('Seattle') = f_{Lc}^{Nodo,Polígono}(g_{id.}, L_c))\}$$

Aquí se utiliza el precálculo del overlay para simplificar la obtención de las geometrías que cumplen con la condición geométrica de la consulta (superposición de aeropuertos con una ciudad) y una medida (count) para obtener el resultado de la consulta.

Las siguientes propiedades son utilizadas en la solución y permiten: asignar una cota superior a la cantidad de subgeometrías generadas con la subpoligonización y utilizar un orden flexible en los layers a superponer para la generación de la subpoligonización de un conjunto de layers.

Propiedad 1. Complejidad de la subpoligonización.
Dada la subpoligonización inducida por N carriers:

(i) El número de subnodos generados es como mucho N(N-1)/2 ;
(ii) El número sublíneas generados es como mucho N^2;
(iii) El número de subpolígonos generados es como mucho $N^2/2 + N/2 + 1$

Propiedad 2. Conmutatividad y asociatividad.
La operación de overlay producida por la subpoligonización inducida por un conjunto de carriers es conmutativa y asociativa.

3 Arquitectura

Esta sección describe la arquitectura de la solución partiendo de una visión conceptual, para luego describir la arquitectura concreta finalmente implementada. También se incluye una descripción de los componentes de software que forman parte de la implementación y la estructura del repositorio de datos utilizado.

3.1 Arquitectura conceptual

En la Figura 5 se muestra la arquitectura conceptual de una solución, basada en el marco teórico, para un modelo de OLAP espacial que permita ejecutar los tipos de consultas deseados, es decir, consultas de agregación geométrica, consultas geométricas, consultas OLAP y consultas GIS-OLAP.

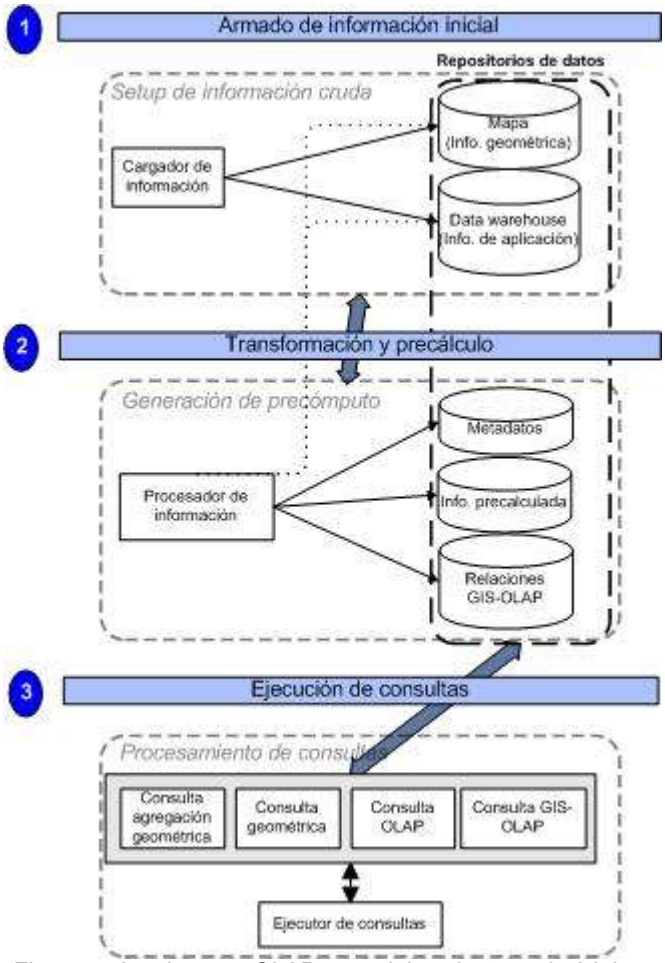


Figura 5: Arquitectura OLAP espacial en 3 etapas/módulos.

La arquitectura puede dividirse en 3 grandes "módulos", cada uno asociado a una etapa de ejecución dentro del ambiente OLAP espacial. Estos módulos permitirán el armado y almacenamiento del repositorio base sobre el que se realizarán consultas (llamaremos información cruda[4] a este tipo de información), la generación y almacenamiento de datos precalculados y finalmente la ejecución de consultas sobre el repositorio de datos (utilizando la información precalculada).

[4] Información que no contiene datos precalculados (p.ej.: la información de aplicación contenida en el data warehouse).

El primer módulo ("Setup de información cruda") permitirá realizar la etapa de "Armado de información inicial", que consistirá de la recolección y consolidación de información a utilizar en el ambiente OLAP espacial, que será finalmente consultada por los usuarios. Estos datos se almacenarán en forma de un data warehouse (información de aplicación) y en un mapa (información geométrica, que tendrá alguna relación con la información de aplicación). El proceso de armado del ambiente se realizará mediante un "cargador de información" que definirá la forma de extracción y procesamiento de los datos para el armado del data warehouse y del mapa a partir de información de fuentes de datos externas al ambiente.

El segundo módulo ("Generador de información precalculada"), permitirá desarrollar la etapa de "transformación y precálculo de información", que consistirá en generar e incorporar en el repositorio de datos, información precalculada. Su componente principal será un "procesador de información" que en forma offline (al estilo OLAP) tomará la información cruda contenida en el data warehouse y generará:

- "Información precalculada": compuesta por las subgeometrías generadas con la subpoligonización del mapa y los valores de los hechos asociados a cada subgeometría asociada con un componente geométrico del mapa (armando la información en forma de dimensión GIS). Además se calcularán y almacenarán los datos de overlay entre los componentes geométricos.
- "Metadatos": información de la estructura del data warehouse, el mapa, y datos comunes a ambos.
- "Relaciones GIS-OLAP", con las relaciones entre componentes geométricos del mapa y datos del data warehouse (p.ej., relación de un punto en el mapa, que representa una sucursal, con la información de la sucursal contenida en el data warehouse).

El tercer módulo, "Procesador de consultas", permitirá realizar la etapa de "Ejecución de consultas" mediante un "Ejecutor de consultas". El ejecutor se encargará de resolver las consultas de agregación geométrica, geométricas, OLAP o GIS-OLAP ingresadas por el usuario, basándose en los datos crudos (información de aplicación y geométrica) y los datos precalculados generados en las etapas anteriores.

3.2 Implementación de la Arquitectura

La solución implementada (Piet) fue diseñada a partir de la arquitectura conceptual y las definiciones presentadas en la Sección 2. La implementación se realizó a partir de un módulo principal (Piet-Engine) que brinda servicios para precálculo de información y ejecución de consultas a tres interfases: Piet-File, Piet-Web y Piet-Jump. Estas interfases proveen a los usuarios acceso a los servicios de Piet-Engine a través de los siguientes medios: archivos de configuración, consultas Piet-Queries y en forma de consulta gráfica. La Figura 6 representa un esquema de la arquitectura implementada.

Descripción de la arquitectura
La arquitectura implementada está compuesta por dos grandes ambientes: el ambiente externo, que corresponde a un ambiente utilizado para el armado de los datos que estarán disponibles en Piet, y el ambiente Piet, que constituye el ambiente de la implementación y será utilizado por los usuarios finales.

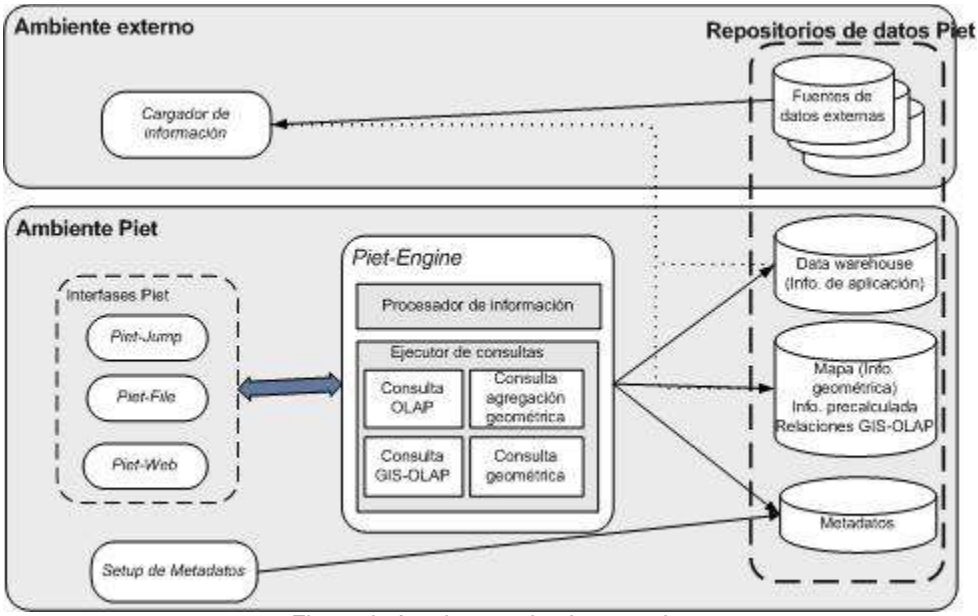


Figura 6: Arquitectura implementada.

El ambiente externo contiene las "fuentes de datos externas" y un procedimiento "Cargador de información" (manual o automático), que extrae información de las fuentes de datos y la almacena en un data warehouse (información de aplicación) y en un mapa relacionado con este (información geométrica).

El ambiente Piet esta compuesto por una serie de módulos que realizan las tareas de "transformación y precálculo" y "ejecución de consultas" introducidas en la sección anterior, trabajando sobre un conjunto de repositorios de información al que llamaremos "Repositorio de datos Piet". El data warehouse y el mapa creados desde el ambiente externo pasarán a formar parte de este repositorio de datos Piet.

Los módulos que forman parte del ambiente Piet son:

- Piet-Engine: constituye la parte central de la implementación ya que lleva adelante los procesos de precálculo de información y ejecución de consultas OLAP, GIS-OLAP y de agregación geométrica[5]. Estos procesos están disponibles en forma de servicios.
- Interfases Piet: conjunto de interfases que brindan al usuario diferentes formas de utilizar los servicios provistos por Piet-Engine. Se implementaron 3 interfases:
    - Piet-Jump: interfaz de tipo gráfica, utilizada para la manipulación de información GIS (espacial), la generación de asociaciones entre componentes geométricos e información de aplicación (del data warehouse), y el armado de consultas de agregación geométrica.
    - Piet-Web: interfaz de tipo gráfica Web, utilizada para la ejecución de consultas OLAP, consultas geométricas (GIS) o consultas GIS-OLAP.
    - Piet-File: interfaz que funciona por medio de archivos de configuración y puede ejecutar todos los servicios provistos por Piet-Engine.
- Setup de metadatos: proceso que define los metadatos que describen el data warehouse y el mapa definidos a partir del ambiente externo y que forman parte del repositorio de datos Piet, junto con la relación entre la información de ambos (información de aplicación relacionada con componentes geométricos).

El repositorio de datos Piet utilizado por los módulos del ambiente Piet esta compuesto por:

- Metadatos, con información sobre:
    - Definiciones OLAP: definición de dimensiones, jerarquías, medidas y cubos OLAP.
    - Dimensiones OLAP: dimensiones OLAP.

[5] Las consultas de agregación geométrica incluyen a las consultas geométricas y a las sumables, según las definiciones de la Sección 2.

    - Metadatos Piet: definición de la dimensión GIS (estructura de los componentes geométricos del mapa) y de las relaciones con información del data warehouse.
- Data warehouse: definido desde el ambiente externo.
- Mapa: definido desde el ambiente externo.
- Información precalculada: subgeometrías resultantes de la subpoligonización, asociación de subgeometrías con componentes geométricos, propagación de medidas y overlay de componentes geométricos.
- Relaciones GIS-OLAP: contiene las asociaciones entre componentes geométricos del mapa y datos contenidos en el data warehouse.

Ejecución de la arquitectura
En la Figura 7 se incluye el esquema de la ejecución de la arquitectura teniendo en cuenta el orden en que debe realizarse cada tarea, los componentes a utilizar y los usuarios especializados encargados de llevarlas adelante.

El ambiente externo será utilizado por un "Administrador de información" que deberá saber que datos de las fuentes de datos externos deberán incluirse en el repositorio de datos Piet. Un "Administrador Piet" será el encargado de configurar el ambiente Piet y ejecutar el precálculo de información, para que finalmente los "Usuarios Piet" ejecuten sus consultas sobre los datos que forman parte del ambiente.

Basándonos en la Figura 7, el armado de un ambiente Piet comienza con el "Armado de información inicial" por parte del usuario "Administrador de la información" utilizando procesos y herramientas fuera de Piet:
- Recopila información de aplicación de las fuentes de datos externas y con ella diseña un data warehouse y lo almacena en el repositorio de datos Piet.
- Define un mapa GIS con información geométrica relacionable con algunos de los datos del data warehouse y almacena sus layers en el repositorio de datos Piet.

A continuación, el "Administrador Piet" se encarga de la ejecución de la tarea de "Transformación y precálculo":
- Realiza el "Setup de Metadatos", definiendo en forma manual los metadatos con información de aplicación (OLAP) y la que relaciona componentes geométricos con información del data warehouse.
- Utiliza la interfaz Piet-JUMP para definir asociaciones entre componentes geométricos e información del data warehouse. Por ejemplo, supongamos que en el mapa existe un layer que ubica "sucursales" en diferentes lugares y que en el data warehouse se almacena el detalle de ventas por sucursal. En este caso, el administrador asociará cada componentes geométricos que representa una sucursal con la correspondiente sucursal del data warehouse.
- Utiliza la interfaz Piet-JUMP o Piet-File para ejecutar el servicio de precálculo del overlay: la interfaz envía los datos ingresados por el usuario al Piet-Engine y este realiza el precálculo de información almacenando los resultados en el repositorio de datos Piet.

Finalmente, el "usuario Piet" utiliza las interfases Piet-JUMP, Piet-Web o Piet-File para ejecutar consultas geométricas, de agregación geométrica, OLAP o GIS-OLAP sobre el repositorio de datos Piet (información precalculada, de aplicación y geométrica).

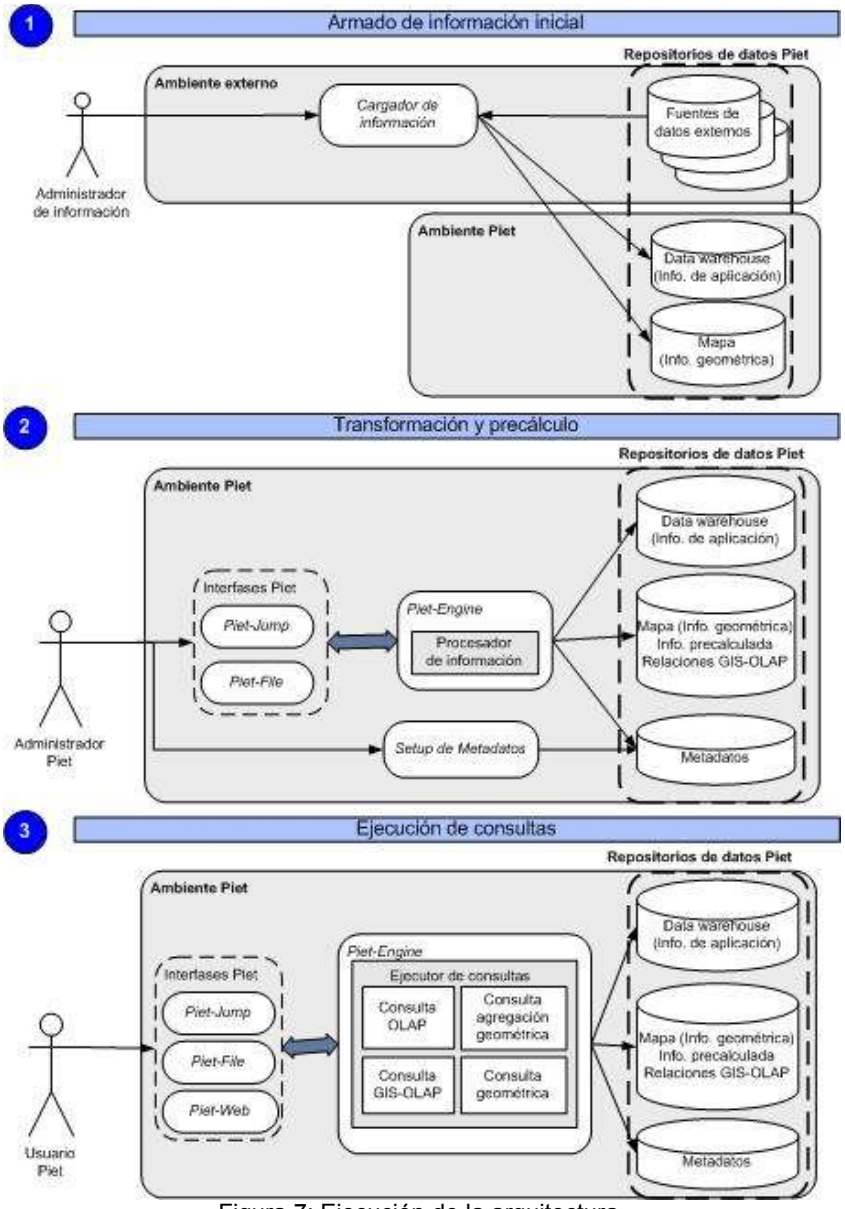


Figura 7: Ejecución de la arquitectura.

3.3 Resumen de componentes Piet

En esta sección se resumen los componentes que forman parte de Piet, incluyendo la plataforma de ejecución y las librerías externas que se utilizan para llevar adelante los servicios ofrecidos. En la Figura 8 se muestran los componentes Piet y su interacción.

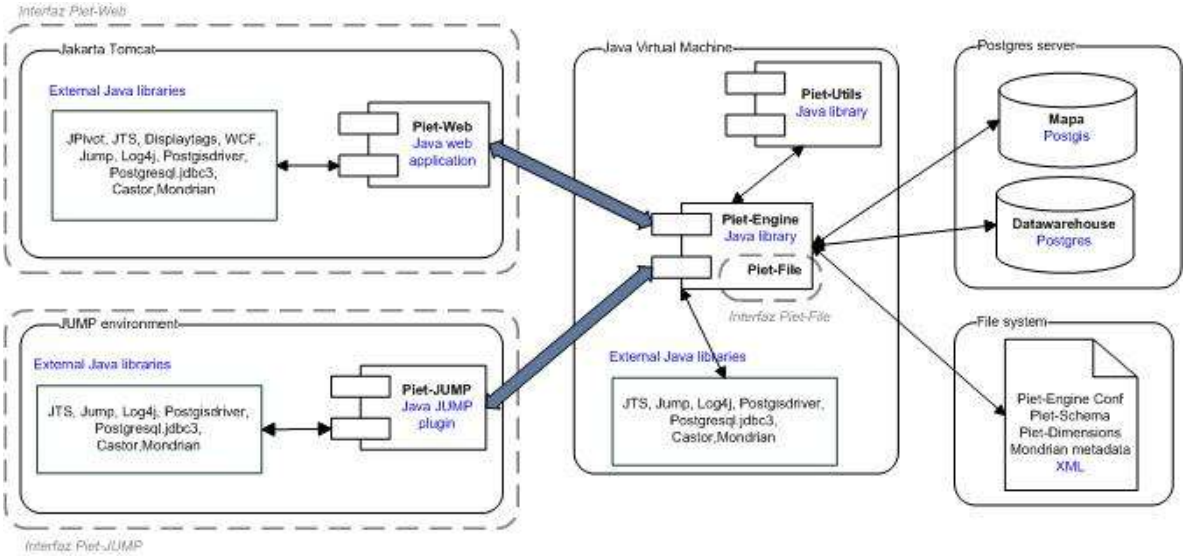


Figura 8: Componentes Piet y su interacción.

La parte central de la implementación (Piet-Engine) es una Java Library (JAR[6]) que se ejecuta en una Java Virtual Machine[7] a través de alguna de las interfases implementadas.
Piet-Engine utiliza la librería Piet-Utils (JAR) y una serie de librerías externas para llevar adelante sus tareas. Las librerías más importantes son: JTS [24] (modificado), JUMP [24], Log4j [12], PostGISdriver [8], Postgresql.jdbc3 [9], Castor [1] y Mondrian [15].

La librería Piet-Utils incluye estructuras de representación especiales para líneas, carrier lines y bounding box, operaciones geométricas optimizadas para la subpoligonización y facilidades para el intercambio de información con el repositorio Piet, herramientas para comparación y ordenamiento de listas de geometrías, y otras utilidades. En la Sección 5 se desarrollan las estructuras de representación y las principales operaciones incluidas en esta librería.

Dentro del JAR Piet-Engine se incluye la interfaz Piet-File. Esta interfaz puede utilizarse ejecutando directamente el JAR de Piet-Engine.

La interfaz Piet-Web es una Java Web Application (WAR[8]), que debe ser ejecutada en un servidor de aplicaciones Java[9]. Las librerías externas más importantes utilizadas por Piet-Web son: JPivot [10], JTS [24], Displaytags [2], WCF [25], JUMP [24], Log4j [12], PostGISdriver [8], Postgresql.jdbc3 [9], Castor [1] y Mondrian [15].

La interfaz Piet-JUMP está compuesta por un conjunto de plugins para JUMP (JAR), que debe ser utilizado en una instalación JUMP, junto con una serie de librerías externas, como por ejemplo: JTS [24], JUMP [24], Log4j [12], PostGISdriver [8], Postgresql.jdbc3 [9], Castor [1] y Mondrian [15].

En todos los ambientes la versión de la librería JTS debe ser la versión modificada para Piet, que se menciona en la Sección 5.

Con respecto a los repositorios de datos, la implementación actual soporta únicamente el motor de base de datos PostgreSQL [21], donde el data warehouse puede almacenarse en una base PostgreSQL y el mapa en una base de datos PostGIS [20] (base PostgreSQL con plugin PostGIS instalado), o toda la información puede

[6] Un archivo JAR es una librería que contiene una serie de clases Java. Estas clases pueden ser utilizadas por terceros, o formar parte de de uno o más programas Java.
[7] Máquina virtual que permite la ejecución (interpretación) de programas Java compilados.
[8] Un archivo WAR es una librería Java que agrupa todos los recursos que componen una aplicación Java Web.
[9] Un servidor de aplicaciones Java es un software que permite la ejecución de aplicaciones Java Web (WAR).

centralizarse en una única base PostGIS[10]. La información precalculada se almacenará en la base de datos que contiene al mapa[11].

Finalmente, la información de configuración y los metadatos para Piet-Engine y Mondrian se almacena en el File System, en forma de los siguientes tipos de documentos XML: Piet-Engine configuration, Piet-Schema, Piet-Dimensions y los metadatos para Mondrian.

En la Sección 4 se describen todos los componentes y documentos desarrollados.

3.4 Estructura de repositorio de datos Piet

En esta sección se describe el diseño del repositorio (base de datos PostgreSQL con plugin PostGIS instalado) que se definió para la solución y se mencionan otras alternativas analizadas.

Un ambiente Piet de ejemplo tendría inicialmente la estructura de base de datos que se muestra en la Figura 9.

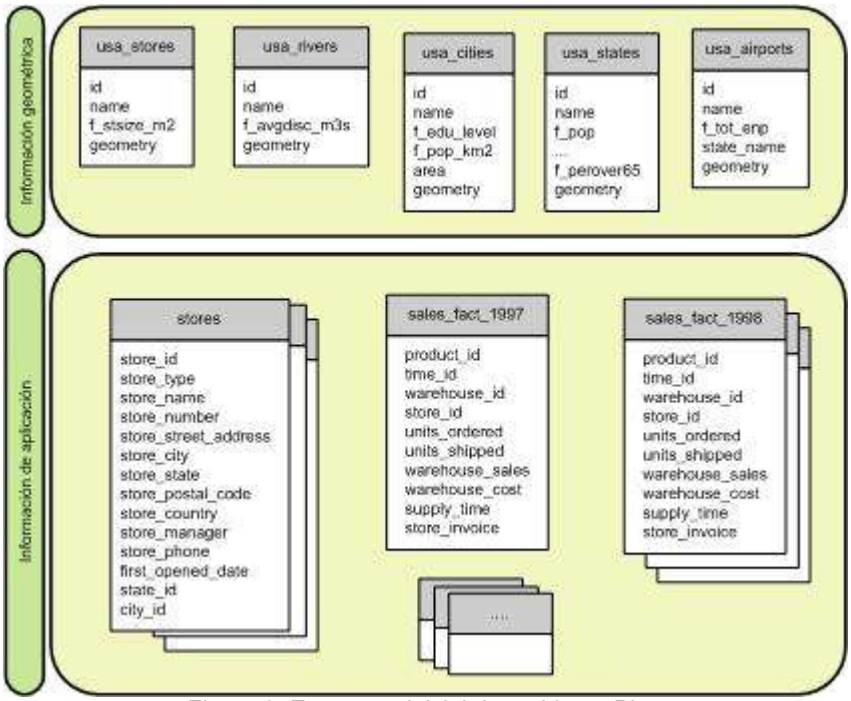


Figura 9: Estructura inicial de ambiente Piet.

En el repositorio Piet, previo al proceso de precálculo de información, se pueden clasificar los datos en forma lógica en:

- Información de aplicación: correspondiente a un data warehouse, con información multidimensional navegable con herramientas OLAP
- Información geométrica: correspondiente a los layers que conforman el mapa, incluyendo sus componentes geométricos y atributos relacionados.

[10] Las bases Postgis complementan a las bases PostgreSQL con soporte para información geográfica, manteniendo toda la funcionalidad de las bases PostgreSQL.
[11] En general, en este documento tomamos como modelo un ambiente con una única base Postgis centralizada (que incluye la información de aplicación y del mapa).

En la Figura 9, la información de aplicación contiene definiciones de:

- Sucursales (tabla "stores"): con datos como la descripción de la sucursal ("store_name"), un id único ("store_id") y el id de la ciudad y el estado ("city_id" y "state_id").
- Hechos: tablas de hechos con detalle de ventas de cada año ("sales_fact_1997" y "sales_fact_1998").
- Tablas adicionales que completan el data warehouse.

La información geométrica contiene una tabla por cada layer del mapa: sucursales ("usa_stores"), ríos ("usa_rivers"), estados ("usa_states"), ciudades ("usa_cities") y aeropuertos ("usa_airports"). Cada tabla-layer tiene un identificador único, información de las geometrías que contienen y los atributos relacionados a cada una (p.ej., "f_pop_km2", "area", "name", etc.).

Tomando el repositorio Piet del ejemplo como estado inicial de un ambiente Piet, el administrador Piet deberá ejecutar el precálculo de información sobre todos los layers y crear las asociaciones entre la parte geométrica y la parte de aplicación.
En la Figura 10 se muestra el repositorio Piet luego de realizar estas tareas. En la figura se puede ver que se agregan al repositorio Piet las áreas lógicas de asociación GIS-OLAP y de información precalculada. Estas áreas contienen:

- Información de asociación GIS-OLAP: con las relaciones entre componentes geométricos y tablas OLAP por medio de los identificadores únicos de cada dato. Esta información es la que permite relacionar la parte de aplicación con la parte geométrica.
- Información precalculada: almacena información generada por el proceso de precálculo. Esta información es utilizada para crear la dimensión GIS incluyendo las subgeometrías, que se usará para resolver consultas de agregación geométrica en el ambiente Piet.

La información precalculada está compuesta por información temporal (cálculos auxiliares), con tablas que almacenan las subgeometrías generadas en la subpoligonización, e información final, con tablas que almacenan datos del overlay de componentes geométricos y relaciones entre geometrías y subgeometrías.
En la tabla gis_mappings se almacenan los layers que forman parte de cada combinación procesada, junto con un identificador único para cada combinación. Esta información es utilizada durante el proceso de precálculo para crear un conjunto de datos con información temporal y final para cada combinación. En la Figura 10 se muestra una única combinación con id 1.

Los datos temporales contiene las subgeometrías generadas (una tabla por cada tipo de subgeometría) y los carrier sets (tablas tmp_point_1, tmp_linestring_1, tmp_polygon_1 y tmp_carrierset_1). En cada tabla se incluye un identificador único para cada subgeometría o carrier line, y la geometría que representa utilizando la representación vectorial.

La información de asociación de subgeometrías se almacena utilizando una tabla para cada tipo de subgeometría (tablas gis_point_1, gis_linestring_1, gis_polygon_1). En la Figura 10 se puede ver que por cada asociación se almacena el identificador único de la subgeometría, la representación de la subgeometría en forma vectorial y el identificador único del componente geométrico, junto con los valores proporcionales de los hechos definidos en el componente geométrico original (campos f_stsize_m2, area, f_pop, etc.).

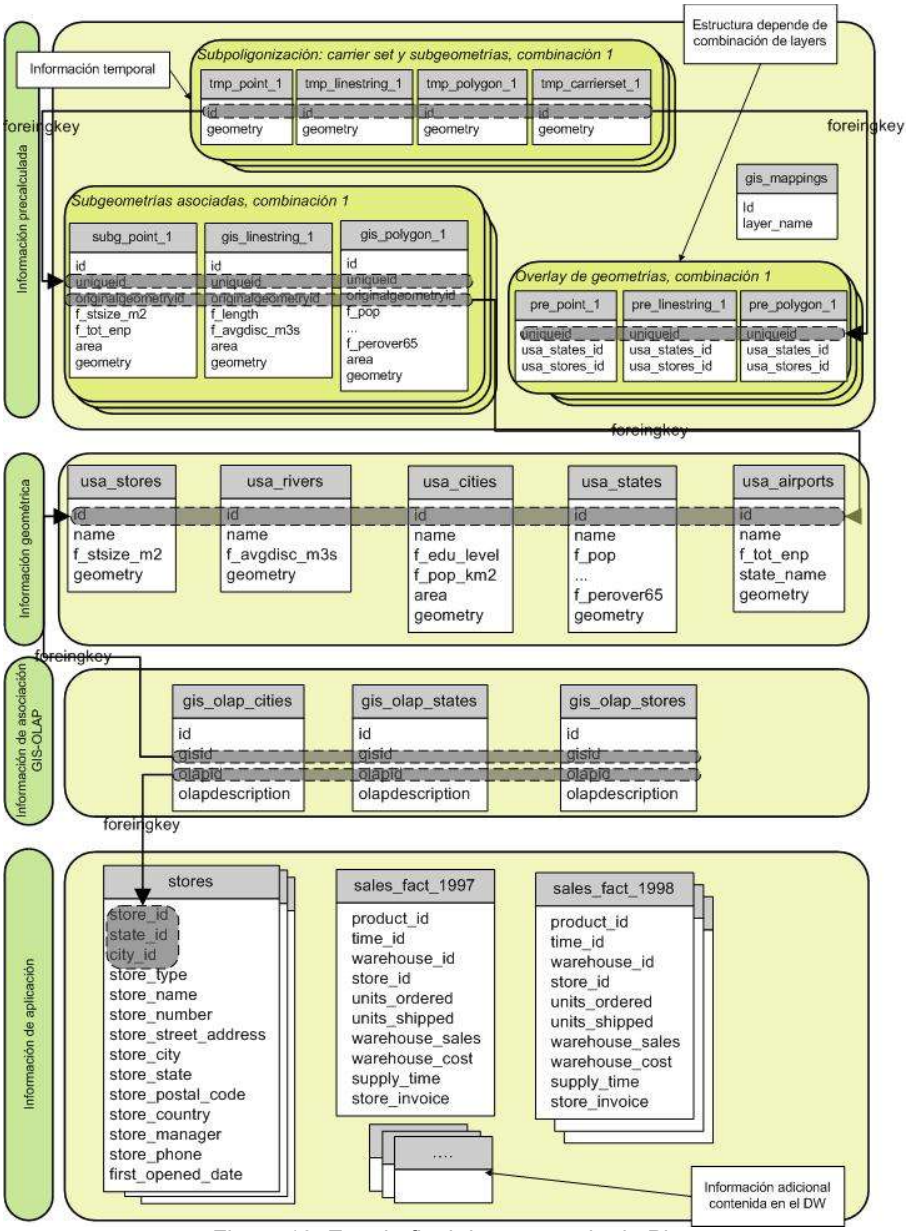


Figura 10: Estado final de un repositorio Piet.

Por último, la información de overlay es almacenada utilizando una tabla para cada tipo de subgeometría (tablas pre_point_1, pre_linestring_1 y pre_polygon_1). La estructura de estas tablas se define a partir de la combinación de layers. En el ejemplo se incluye sólo el overlay entre los layers usa_states y usa_stores, que correspondería a la combinación 1 de layers (la que se incluye en el ejemplo). En este caso, se almacena el identificador único de la subgeometría común a ambos layers y los identificadores de los componentes geométricos de los layers mencionados que comparten dicha subgeometría.

En el caso en que se utilizara los métodos con grilla mencionados en la Sección 5, la información temporal incluirá los datos de estructura de la grilla (cuadrantes) y relaciones con subgeometrías y componentes geométricos, para cada combinación de layers.

Alternativas de almacenamiento evaluadas
Algunas de las alternativas de almacenamiento evaluadas (no implementadas) fueron:

- Almacenar todas las subgeometrías generadas durante la subpoligonización en una tabla única, para cada combinación de layers: esto se descartó porque se consideró que el tamaño de la tabla final podría ser demasiado grande, perjudicando la performance de las etapas de asociación y propagación de hechos.
- Almacenar la información de asociación de subgeometrías en tablas separadas por layer: se consideró que esta forma de almacenamiento aumentaría considerablemente la complejidad de la generación de consultas SQL a partir de sentencias GISOLAP-QL, sin agregar beneficios importantes.

4 Descripción de la implementación

El objetivo de Piet es implementar un modelo de OLAP espacial que permita a los usuarios la ejecución de consultas geométricas puras, de agregación geométrica, OLAP y GIS-OLAP integradas, sobre un repositorio de datos compuesto por información de aplicación (data warehouse) e información geométrica (mapa). En esta sección se describe cada uno de los componentes que forman parte de Piet y la interacción entre ellos.

4.1 Mapa y Data warehouse para ambiente Piet

La solución Piet no incluye herramientas (ni procedimientos) para la extracción de información de fuentes de datos externos y el posterior armado del data warehouse, de los metadatos OLAP, ni tampoco del mapa a utilizar, por lo que esta información deberá ser generada por medio de herramientas o procedimientos externos.

Dentro de un ambiente Piet puede no incluirse la parte de aplicación (data warehouse) de modo de utilizar únicamente un mapa (dimensión GIS) como repositorio de información de base. En este caso sólo se podrán ejecutar consultas de agregación geométrica y consultas geométricas, con medidas definidas sobre los componentes geométricos de la dimensión o sobre sus atributos. En este caso no podrán ejecutarse consultas OLAP puras o GIS-OLAP.
Los mapas GIS a utilizar en Piet pueden crearse con una herramienta para dibujo de mapas (p.ej., JUMP) u obtenerse de algún distribuidor de mapas (p.ej., vía Internet). Para que un mapa GIS pueda ser utilizado en un ambiente Piet, debe cumplir con las siguientes condiciones:

- Debe incluir un layer con un bounding box (que abarque a los componentes geométricos de los demás layers) rectangular, y con los bordes paralelos a las líneas de los ejes de coordenada.
- Todos los componentes geométricos deben tener un identificador único (sin importar el layer en que se encuentran).
- Cada layer puede tener un único tipo de geometría.
- El nombre de los hechos asignados a componentes geométricos de cada layer por medio de atributos debe comenzar con el prefijo "f_" (a excepción del área, que no requiere prefijo).
- Los hechos definidos en atributos de componentes geométricos sólo pueden ser de tipo "double".
- Componentes geométricos de un mismo layer deben tener definidos los mismos hechos.
- Los componentes geométricos soportados en esta implementación son polígonos, polilíneas o puntos (no se soportan "multipolígonos").
- Debe cumplir con el estándar OpenGIS [16] y estar almacenado en una base de datos PostGIS.

El data warehouse que contiene la información de aplicación de base que se usará en Piet debe estar configurado para funcionar con Mondrian, incluyendo las definiciones (metadatos) OLAP de dimensiones, jerarquías, cubos y medidas, ya que Piet no agrega ninguna condición adicional sobre estos.

Para que los usuarios finales puedan realizar consultas GIS-OLAP es necesario que algunos de los datos contenidos en el mapa y en el data warehouse tengan una relación lógica (en nuestro ejemplo, existen en el mapa geometrías que representan sucursales y también existe el concepto de sucursal del lado del data warehouse).

4.2 Integración OLAP-GIS

La integración GIS-OLAP permite realizar consultas con una parte de contenido geométrico (p.ej., sucursales en estados cruzados por ríos) y otra de información de aplicación (p.ej., cantidad de ventas en sucursales a lo largo de los últimos 4 años), dando la posibilidad al usuario de realizar consultas como: "Dar la cantidad de ventas durante los últimos 4 años, en sucursales pertenecientes a estados cruzados por un río".

Para lograr la integración de consultas GIS-OLAP se definió:

- Un modelo de esquema Piet (Piet-Schema), que especifica el formato de los metadatos que definen el esquema de dimensión GIS y permitirá, entre otras cosas, realizar la integración GIS-OLAP.
- Un lenguaje de consulta denominado GISOLAP-QL, en el que una consulta está conformada por una sección de información GIS (GIS-Query, con una sintaxis especialmente definida) y otra OLAP (OLAP-Query, con sintaxis MDX).

La implementación del soporte a consultas GIS-OLAP implicó el desarrolló de los siguientes componentes:

- Componente para la ejecución de consultas geométricas (sección GIS de una consulta GISOLAP-QL).
- Interfases para visualizar resultado de consulta geométrica.
- Componente para la integración de resultado de consulta geométrica con consulta OLAP.
- Componente para ejecución de consultas OLAP puras o GIS-OLAP integradas.
- Interfaz para visualización y navegación de resultados de consultas GIS-OLAP con herramientas OLAP (roll-up y drill-down).

A continuación se describen las definiciones creadas y los componentes desarrollados.

4.2.1 Estructura de Piet-Schema

Un Piet-Schema es un conjunto de definiciones (metadatos) que incluye la forma de almacenamiento de los componentes geométricos de la dimensión GIS y las medidas asociadas (tanto a hechos definidos en atributos de los componentes como a los componentes geométricos en sí mismo), las subgeometrías correspondientes a la subpoligonización de todos los layers que forman parte del mapa y la relación entre los componentes geométricos y la información de aplicación que será utilizada para integrar consultas espaciales con consultas OLAP.
Este tipo de documentos es utilizado por Piet para resolver consultas GIS-OLAP escritas utilizando el lenguaje que se describe en la Sección 4.2.2.

Los metadatos se almacenan en documentos XML que contienen tres tipos de tags: "Subpoligonization", "Layer" y "Measure".
En la Figura 11 se muestra el tag "Subpoligonization" de un Piet-Schema, donde se especifica información relacionada a la asociación de subgeometrías teniendo en cuenta la subpoligonización que usa todos los layers del mapa. En una especificación existirá un único segmento "Subpolygonization".

```
<Subpolygonization>
       <SubPLevel name="Polygon" table="gis_subp_polygon_4" primaryKey="id"
uniqueIdColumn="uniqueid" originalGeometryColumn="originalgeometryid"/>
       <SubPLevel name="Linestring" table="gis_subp_linestring_4" primaryKey="id"
uniqueIdColumn="uniqueid" originalGeometryColumn="originalgeometryid"/>
       <SubPLevel name="Point" table="gis_subp_point_4" primaryKey="id"
uniqueIdColumn="uniqueid" originalGeometryColumn="originalgeometryid"/>
</Subpolygonization>
```

Figura 11: Ejemplo de tag "Subpoligonization".

En este tag se incluye la ubicación de cada tipo de subgeometría (subnodo, subpolígono o sublínea) en el repositorio de datos (base PostGIS donde se almacena el mapa). En el tag se detalla el nombre de la tabla que contiene cada tipo de subgeometría, y los nombres de los campos primary key, de identificación única de la subgeometría y de identificación única del componente geométrico asociado.

En la Figura 12 se muestra un tag "Layer", que describe la información de cada layer del mapa y su relación con las subgeometrías y el data warehouse, definiendo la estructura de la dimensión GIS. En un Piet-Schema existirá una lista de "Layer", con un tag "Layer" por cada layer del mapa (representando una jerarquía dentro de la dimensión GIS).

```
<Layer name="usa_states" hasAll="true" table="usa_states" primaryKey="id"
geometry="geometry" descriptionField="name">
       <Properties>
              <Property name="Population" column="f_pop" type="Double" />
              <Property name="Total income" column="f_a13" type="Double" />
              <Property name="Total number of jobs" column="f_a34" type="Double" />
              <Property name="Male pop" column="f_male" type="Double" />
              <Property name="Population" column="f_female" type="Double" />
              <Property name="Population" column="f_under18" type="Double" />
              <Property name="Population" column="f_medage" type="Double" />
              <Property name="Population" column="f_perover65" type="Double" />
       </Properties>
       <SubpolygonizationLevels>
              <SubPUsedLevel name="Polygon" />
              <SubPUsedLevel name="Linestring" />
              <SubPUsedLevel name="Point" />
       </SubpolygonizationLevels>
       <OLAPRelation table="gis_olap_states" gisId="gisid" olapId="olapid"
olapDimensionName="Store" olapLevelName="Store State">
              <OlapTable name="store" id="state_id" hierarchyNameField="store_state"
hierarchyAll="[Store].[All Stores]" />
       </OLAPRelation>
</Layer>
```

Figura 12: Ejemplo de tag "Layer".

Un tag "Layer" contiene el nombre del layer en el mapa, el nombre de la tabla de almacenamiento y los nombres de los campos que contienen la primary key, la geometría y la descripción. En una lista de tags "Properties" se detallan los hechos asociados a los componentes geométricos de ese layer, incluyendo el nombre del hecho, el campo que lo contiene y el tipo de dato que lo representa. En un tag "SubpolygonizationLevel" se indica los niveles de subpoligonización utilizables para las geometrías de este layer (p.ej., si es un layer de ríos, sólo podrán usarse los niveles "point" y "line"). Finalmente, en caso de existir una relación entre este layer e información del data warehouse, la misma se define en un tag "OLAPRelation", que incluye la información de la tabla de relación (nombre y campos de identificación de geometría e identificación de objeto OLAP, nombre de la dimensión OLAP relacionada y el nivel en que se inserta la información espacial dentro de la jerarquía). Dentro de este tag se incluye otro ("OLAPTable") con información del nombre de la tabla OLAP, el campo de identificación a relacionar con la tabla de relación, el campo de donde se toma el nivel de la jerarquía OLAP y el segmento de MDX que se usará para insertar una nueva fila (dimensión) en el MDX original de una sentencia GISOLAP-QL. En la Figura 12, la asociación entre estados del mapa y estados del data warehouse se realiza por medio de la tabla gis_olap_states, que se relaciona con la tabla OLAP "store" del data warehouse, diferenciando cada state a través del campo "state_id" de dicha tabla y obteniendo el nombre de los states del campo "store_state". Además, el segmento de MDX para obtener el nivel All de la jerarquía "Store" (a la que pertenecen los estados) es "[Store].[All Stores]". En la Figura 13 se muestra un ejemplo con algunas columnas y filas de la tabla "store" del data warehouse asociado al ejemplo de la Figura 12.

La información del tag ("OLAPTable") es usada por Piet-Engine para integrar consultas geométricas con consultas OLAP, como se describe en la Sección 4.4.4.

R...	store_id...	store_name...	store_city ...	store_state...	store_countr...	city_id ...	state_id ...
1	13	Store 13	Lincoln	NE	USA	13	1
2	19	Store 19	Wichita	KA	USA	19	2
3	20	Store 20	Topeka	KA	USA	20	2
4	21	Store 21	Salem	OR	USA	21	3
5	5	Store 5	Minneapolis	MT	USA	5	4
6	10	Store 10	New Orlea...	LO	USA	10	5
7	2	Store 2	Cheyenne	WY	USA	2	6
8	16	Store 16	Casper	WY	USA	16	6
9	22	Store 22	Las Vegas	NV	USA	22	7

Figura 13: Ejemplo de porción de tabla "stores" de data warehouse.

La última parte de una definición Piet-Schema contiene una lista de tags "Measure", donde se especifica las medidas geométricas (asociadas a componentes geométricos de la dimensión GIS) disponibles en el ambiente Piet. En la Figura 14 se muestra dos medidas de ejemplo.

```
<Measure name="StoresQuantity" layer="usa_stores" aggregator="count"/>
<Measure name="RiverSegments" layer="usa_rivers" aggregator="count"/>
```

Figura 14: Ejemplo de lista de "Measure".

Un tag "Measure" tiene definido un nombre, el layer del mapa sobre el que aplica y el tipo de función[12] asociado.

4.2.2 Lenguaje de consulta: GISOLAP-QL

La sintaxis para el lenguaje GISOLAP-QL fue definida con el objetivo de expresar consultas geométricas integradas con consultas OLAP (consultas GIS-OLAP), evitando modificar la sintaxis MDX para lograrlo.
Una consulta GISOLAP-QL tiene la siguiente forma:

```
GIS-Query | OLAP-Query
```

La sección GIS-Query contiene la consulta aplicable a la parte espacial (geométrica), escrita con una sintaxis creada especialmente para la solución y la sección OLAP-Query contiene la consulta aplicable sobre la información de aplicación (data warehouse), escrita en MDX. El pipe ("|") sirve como separador.
El formato de la sección GIS-Query de una sentencia GISOLAP-QL está compuesto por tres secciones (SELECT, FROM y WHERE), cada una finalizada con ";", de la siguiente forma:

```
SELECT "lista de layers y/o medidas";FROM "pietSchema";WHERE "operaciones
geométricas";
```

La parte SELECT incluye una lista de layers y/o medidas a seleccionar que deberán estar definidas en el Piet-Schema correspondiente. Tanto los layers como las medidas se aplicarán sobre la información devuelta por operaciones geométricas de la consulta, expresadas en la sección WHERE. Los resultados "seleccionados" corresponderán a las geometrías que cumplan con las condiciones de la sección WHERE y estén incluidas en la lista de layers del SELECT o a las medidas aplicadas sobre ellas.
La sintaxis para especificar un layer es `layer."Nombre de layer"` mientras que para especificar una medida geométrica se utiliza el formato `measure."Nombre de measure"`. Por ejemplo, `layer.usa_rivers` y `measure.StoresQuantity` sería una posible utilización del layer usa_rivers y la medida geométrica StoresQuantity respectivamente.

El segmento FROM contiene únicamente el nombre del Piet-Schema a utilizar con la consulta (metadatos).

La sección WHERE de un GIS-Query contiene operaciones geométricas a aplicar sobre una lista de layers o miembros, y el tipo de subgeometrías a utilizar para resolver cada operación. Todos los elementos asociados al llamado de una operación geométrica (layers, miembros y tipo de subgeometría) deben estar separados por coma.
La sintaxis para una operación geométrica es:

```
"Nombre operación"("Lista layers/miembros","tipo subgeometría")
```

Donde:

- "Nombre operación" especifica el nombre de la operación geométrica a ejecutar.
- "Lista layers/miembros" puede ser una lista de layers sobre la que se aplicará la operación geométrica definidos con la sintaxis de la sección SELECT, o miembros particulares de un layer representados como `"Nombre layer"."Identificador miembro"`.
- "Tipo subgeometría" especifica el tipo de subgeometría a utilizar para resolver la operación. Los valores de este parámetro pueden ser: "Point", "LineString" o "Polygon", que corresponden a subnodo, sublínea y subpolígono respectivamente. El tipo de subgeometría especificado deberá ser coherente con la consulta a realizar[13].

[12] En esta implementación sólo se incluye la función de agregación "COUNT", que permite contar las geometrías asociadas a la medida (i.e., una medida de este tipo sobre el layer de ciudades permitiría realizar consultas como "Total de ciudades dentro del estado Nevada").

Las funciones geométricas soportadas en esta etapa de la implementación son "intersection" y "contains":

- La función "intersection" retorna los componentes geométricos de layers (o miembros) indicados que se intersecan entre sí, utilizando el nivel de subgeometría especificado para obtener la respuesta. Para ello consulta las subgeometrías del nivel especificado que son comunes a componentes geométricos de todos los layers.
- La función "contains" tiene dos parámetros (de tipo "layer"). La función busca los componentes geométricos del primer parámetro que contienen al menos un componente del segundo layer, retornando la identificación de cada uno de los componentes (repitiendo filas en caso de que el primer componente contenga más de un componente del segundo).

La sección WHERE puede incluir, adicionalmente, áreas de selección (en forma de polígonos) para marcar una zona dentro de la cual se ejecutarán las funciones[14].

Un ejemplo de llamado a la función contains para "obtener los estados que contienen ríos" utilizando sublíneas generadas y asociadas durante el proceso de precálculo sería:

```
Contains(layer.usa_states,layer.usa_rivers,subplevel.Linestring)
```

Piet resuelve las consultas GISOLAP-QL utilizando el repositorio de datos Piet introducido en la Sección 3.2, incluyendo la información del mapa, data warehouse, datos precalculados y metadatos (definición de la dimensión GIS y de las relaciones con información del data warehouse).

Consultas avanzadas

El lenguaje GISOLAP-QL permite relacionar la aplicación de dos o más funciones mediante el uso de **operadores booleanos** entre cada par de llamados a funciones. Cada operador booleano utilizado en una consulta tiene asociado un layer que se utiliza para relacionar la función geométrica a la izquierda del operador, con la de la derecha del mismo.

Un ejemplo del uso de un operador booleano para consultar la "descripción de ríos, ciudades y sucursales, para las sucursales dentro de una ciudad que se interseca con al menos un río" es el siguiente:

```
SELECT layer.usa_rivers,layer.usa_cities, layer.usa_stores;FROM PietSchema;WHERE
intersection(layer.usa_rivers,layer.usa_cities,subplevel.Linestring) and
(layer.usa_cities) contains(layer.usa_cities,layer.usa_stores,subplevel.Point);
```

Donde la selección de layers incluye los layers "usa_rivers", "usa_stores" y "usa_cities" y las condiciones son que los componentes del layer "usa_rivers" se intersequen con los de "usa_cities" con nivel sublínea, y que las ciudades que contengan por lo menos una sucursal. Para esta última operación (contains) se utiliza el nivel subnodo. Ambas funciones se relacionan por medio del operador "and" utilizando las ciudades que cumplan ambas condiciones. El resultado de la ejecución en Piet es el siguiente:

Usa_rivers	Usa_stores	Usa_cities
r1	Store 21	Salem
r1	Store 9	Salem
r1	HQ	Seattle
r1	Store 6	Seattle
r1	Store 7	Seattle

Utilizando una **medida geométrica** definida en un Piet-Schema se puede consultar por ejemplo el "total de sucursales por ciudad". La consulta sería:

```
SELECT layer.usa_cities,measure.StoresQuantity;FROM PietSchema;WHERE intersection
(layer.usa_cities,layer.usa_stores,subplevel.Point);
```

Donde la selección incluye las geometrías del layer "usa_cities" y la medida "StoresQuantity" bajo las condiciones de que se intersequen las geometrías del layer "usa_cities" con las de "usa_stores", con nivel

[13] En el caso de la intersección entre ríos ("Linestring") y sucursales ("Point"), no tendría sentido utilizar nivel "Polygon", por ejemplo.

[14] Esta opción fue pensada para una futura integración con el servicio de aplicación de funciones provisto por Piet-Engine, como se menciona en la Sección 7.

subnodo. En este caso, la medida "StoresQuantity" debe estar definida en el archivo Piet-Schema, como se muestra en la Figura 14, donde se asocia la función de agregación a utilizar para calcular la medida ("count", en el ejemplo).
El resultado de la ejecución sería:

Usa_stores_count	Usa_cities
1	Los Angeles
2	Salem
3	Seattle

Las consultas GISOLAP-QL también soportan la utilización de miembros para realizar una selección u operación. Por ejemplo, para consultar el "detalle de aeropuertos, ciudades y sucursales para el estado con id 6", la consulta sería:

```
SELECT layer.usa_cities,layer.usa_airports,layer.usa_stores;FROM PietSchema;WHERE
intersection(usa_states.6,layer.usa_cities,subplevel.Point) and(layer.usa_states)
intersection(usa_states.6,layer.usa_airports,subplevel.Point) and(layer.usa_states)
intersection(usa_states.6,layer.usa_stores,subplevel.Point);
```

Donde la selección incluye geometrías de los layers "usa_cities", "usa_airports" y "usa_stores" con tres condiciones de intersección relacionadas mediante la función "and" a través del layer usa_states. Cada operación de intersección relaciona el miembro 6 del layer "usa_states" con uno de los layers de la selección. En esta consulta no se utiliza ninguna medida ya que sólo se quiere obtener la descripción de los componentes geométricos que cumplen con las condiciones (se trata de una consulta geométrica).
El resultado de la ejecución de la consulta es:

Usa_airports	Usa_stores	Usa_cities
Eastern Oregon Regional At Pendleton	Store 21	Salem
Eastern Oregon Regional At Pendleton	Store 9	Salem
Eastern Oregon Regional At Pendleton	Store 21	c40
Eastern Oregon Regional At Pendleton	Store 9	c40

Ejecución de una consulta GISOLAP-QL
Utilizando el lenguaje GISOLAP-QL, un usuario Piet que desea consultar "cantidad de unidades vendidas, costos y total de venta para los diferentes productos y medios de promoción, para los diferentes estados del mapa", escribirá:

```
SELECT layer.usa_states;FROM PietSchema;WHERE intersection
(layer.usa_states,layer.usa_stores,subplevel.point);
            |
select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]}
ON columns,{([Promotion Media].[All Media], [Product].[All Products])} ON rows from
[Sales] where [Time].[1997]
```

La sección GIS-Query de esta consulta selecciona los estados de la dimensión GIS definida en el Piet-Schema que cumplan con la condición de intersecarse con sucursales (utilizando nivel subnodo para resolver esta operación), mientras que la sección OLAP-Query consulta las medidas OLAP para obtener cantidad de unidades vendidas ("Unit Sales"), los costos ("Store Cost") y el total de ventas ("Stores Sales"), sobre las dimensiones medios de promoción ("Promotion Media") y producto ("Product"), partiendo de nivel "All". Supongamos que la dimensión GIS "Store" está conformada por la jerarquía "All"->"País"->"Estado"->"Ciudad"->"Sucursal" definida en el documento de dimensión GIS (metadatos).

Al ejecutar esta consulta en un ambiente Piet se resuelve como primer paso la consulta GIS-Query. El resultado espacial sería el siguiente:

Gis_id	Descripción de estado en GIS
1	California
2	Orlando
3	Washington

El resultado es un conjunto de estados con su identificación de geometría. Este resultado no es devuelto a la interfaz, sino que el motor Piet-Engine continúa con el procesamiento: utilizando el tag "OLAPRelation" del Piet-Schema (Figura 12) obtiene el nombre de la tabla de asociación GIS-OLAP ("gis_olap_states" en el ejemplo) que utiliza para obtener los identificadores de los estados en la parte de aplicación. Con estos identificadores, con el nombre de la tabla OLAP asociada a los estados espaciales ("stores"), y con la información de los niveles de la dimensión "Store", se obtiene la descripción de los estados y de cada nivel de la jerarquía dentro de OLAP para dichos estados (en la Figura 13 se puede ver un ejemplo de una porción de la información de la tabla "stores"). La información manejada por Piet en este momento sería la siguiente:

Gis_id	Descripción de estado en GIS	OLAP_id	Descripción de estado en OLAP	Descripción de país en OLAP
1	California	32	CA	USA
2	Orlando	41	OR	USA
3	Washington	9	WA	USA

En este paso se tienen los identificadores gis y olap para cada uno de los tres estados, junto con la descripción utilizada en la parte geométrica ("Descripción de estado en GIS"), la descripción utilizada en la parte de aplicación ("Descripción de estado en OLAP") y la descripción del nivel país ("Descripción de país en OLAP") asociada a cada estado.
A continuación, Piet arma una subsentencia MDX para cada estado concatenando los diferentes elementos que forman parte de la jerarquía Store hasta llegar a nivel estado (en este caso "All"->"País"->"Estado"), obteniendo el nivel "All" del archivo Piet-Schema (atributo hierarchyAll="[Store].[All Stores]" del tag OLAPTable, en la Figura 12) y los valores de "País" y "Estado" a partir del resultado anterior. Finalmente concatena la directiva MDX Children[15] para que la subsentencia permita obtener los hijos de cada estado seleccionado (en la jerarquía "Store" serán las ciudades). El ejemplo de una de estas subsentencias MDX completa sería:

```
[Store].[All Stores].[USA].[CA].Children
```

Las subsentencias correspondientes a los tres estados resultantes de la consulta geométrica son relacionadas utilizando las sentencias Union[16] y Hierarchize[17] de MDX.
La porción de MDX final generada con la información espacial luego de todo este procesamiento, será:

```
Hierarchize(Union(Union({[Store].[All Stores].[USA].[CA].Children},{[Store].[All
Stores].[USA].[OR].Children}),{[Store].[All Stores].[USA].[WA].Children}))
```

Donde se puede observar que la consulta sobre la dimensión GIS buscará los hijos de los estados "CA", "OR" y "WA" resultantes de la consulta espacial. Según la jerarquía de la dimensión GIS "Store", debajo del nivel estado está el nivel ciudad, por lo que el resultado de esta dimensión comenzará a partir de dicho nivel.

Finalmente, la porción MDX generada se inserta en el MDX de la sección OLAP-Query de la consulta original como una dimensión adicional de esa consulta, dando origen a una nueva sentencia MDX válida que integra la información espacial con la de aplicación.

El MDX generado con la ejecución de la consulta GISOLAP-QL de ejemplo sería:

[15] La directiva Children de MDX [13] retorna el conjunto de hijos de un miembro especificado.
[16] La directiva Union de MDX [13] retorna el conjunto formado por la unión de dos conjuntos.
[17] La directiva Hierarchize de MDX [13] ordena los miembros de un conjunto según una jerarquía.

```
select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]}
ON columns, Crossjoin(Hierarchize(Union(Union({[Store].[All Stores].[USA].[CA].
Children},{[Store].[All Stores].[USA].[OR].Children}),{[Store].[All Stores].[USA].
[WA].Children})),{([Promotion Media].[All Media], [Product].[All Products])} ) ON
rows  from [Sales] where [Time].[1997]
```

Se puede observar que la nueva dimensión, producto de la unión de las diferentes ciudades dentro de los estados es cruzada (utilizando la operación crossjoin[18] de MDX) con las dimensiones previamente definidas en la consulta OLAP original, como se ve en la sección "rows". Las mediadas no son modificadas, como se ve en la sección "columns".
Piet permite la ejecución de la consulta MDX final en el motor Mondrian, devolviendo al usuario el resultado final de la ejecución de la consulta GISOLAP-QL como sigue:

			Measures		
Store	Promotion Media	Product	Unit Sales	Store Cost	Store Sales
+Los Angeles	+All Media	+All Products			
+Salem	+All Media	+All Products			
+Seattle	+All Media	+All Products	46,996	40,037.98	100,295.52

El resultado está compuesto por tres dimensiones: "Store", "Promotion Media" y "Product". La dimensión "Store" corresponde a la dimensión GIS incluida por el usuario mediante la sección espacial de la consulta GISOLAP-QL, y muestra la jerarquía "Store" en el nivel "City" (nivel inferior a "States", como se indicó en la consulta). Las otras dimensiones derivan de la sección OLAP-Query de la consulta original. Además se incluyen las medidas de la consulta OLAP: "Unit sales", "Store cost" y "Store sales".
El usuario Piet puede utilizar herramientas OLAP para navegar el resultado (drill-down o roll-up en las dimensiones). A continuación se muestra el drill-down sobre "Seattle", que despliega las sucursales ("Stores") dentro de esa ciudad.

			Measures		
Store	Promotion Media	Product	Unit Sales	Store Cost	Store Sales
+Los Angeles	+All Media	+All Products			
+Salem	+All Media	+All Products			
-Seattle	+All Media	+All Products	46,996	40,037.98	100,295.52
HQ	+All Media	+All Products			
Store 6	-All Media	+All Products	21,333	18,266.44	45,750.24
	Bulk Mail	-All Products	1,512	1,323.63	3,295.13
		+Drink	139	126.06	297.48
		+Food	1,105	963.90	2,410.03
		+Non-Consumable	268	233.67	587.62
	Cash Register Handout	+All Products	514	452.56	1,132.35
	Daily Paper	+All Products	1,279	1,098.90	2,735.05
	Daily Paper, Radio	+All Products	511	429.27	1,103.81
	Daily Paper, Radio, TV	+All Products	899	760.03	1,906.56
	In-Store Coupon	+All Products			
	No Media	+All Products	14,051	12,035.61	30,154.93
	Product Attachment	+All Products	1,013	866.55	2,184.40
	Radio	+All Products			
	Street Handout	+All Products	258	211.20	528.76
	Sunday Paper	+All Products	853	721.97	1,785.55
	Sunday Paper, Radio	+All Products			
	Sunday Paper, Radio, TV	+All Products			
	TV	+All Products	443	366.73	923.70
Store 7	+All Media	+All Products	25,663	21,771.54	54,545.28

Mediante GISOLAP-QL, los usuarios Piet pueden realizar además consultas espaciales puras completando únicamente la sección GIS-Query de una consulta GISOLAP-QL, o consultas OLAP puras, completando la sección OLAP-Query de la consulta Piet.
En esta implementación las consultas GIS-OLAP soportadas por GISOLAP-QL sólo pueden utilizar medidas definidas en la parte de aplicación. Para realizar consultas que utilicen medidas sobre hechos definidos como

[18] La directiva Crossjoin de MDX [13] retorna el producto cruzado de uno o más conjuntos.

atributos de componentes geométricos debe utilizarse la interfaz gráfica JUMP, como se describe en la Sección 4.4.1.

4.3 Motor de precálculo y ejecución: Piet-Engine

El componente Piet-Engine constituye la parte central de la solución Piet, ya que provee a las diferentes interfases desarrolladas los servicios de precálculo de información, ejecución de consultas de agregación geométrica y ejecución parcial/total de sentencias GISOLAP-QL, que permiten llevar adelante las operaciones planteadas como objetivo de la solución (precálculo de información y resolución de consultas geométricas, de agregación geométrica, OLAP y GIS-OLAP). Un esquema general se muestra en la Figura 15, indicando para cada servicio el tipo de consulta o tarea que permite realizar.

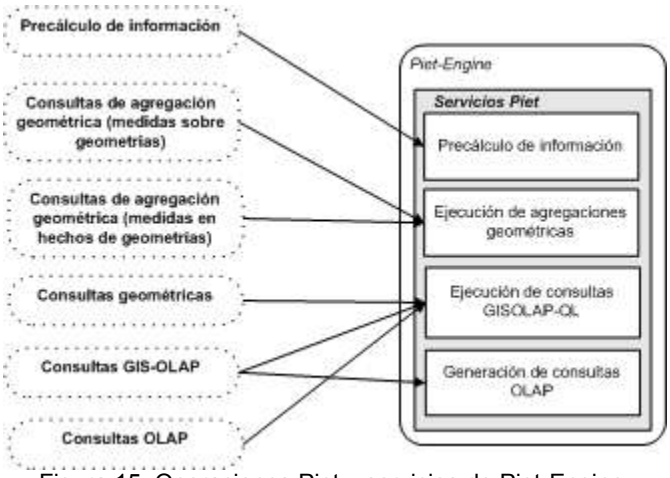


Figura 15: Operaciones Piet y servicios de Piet-Engine.

En la figura puede verse que el proceso de precálculo de información es llevado adelante por el servicio con igual nombre.
El servicio de ejecución de agregaciones geométricas permite resolver consultas geométricas y de agregación geométrica con medidas sobre atributos geométricos (p.ej., población de cierto sector de Buenos Aires). Estas consultas de agregación geométrica corresponden a las consultas sumables definidas en la Sección 2.
El servicio de ejecución de consultas GISOLAP-QL permite resolver consultas geométricas, incluyendo consultas con medidas geométricas (p.ej., cantidad de ríos que cruzan Buenos Aires) y consultas OLAP directamente sobre la parte de aplicación.
Por último, el servicio de generación de consultas OLAP permite transformar consultas GIS-OLAP en consultas OLAP (siguiendo los pasos descritos en la Sección 4.2.2).

Todos los servicios provistos por Piet-Engine basan su funcionamiento en un documento de configuración (Piet-Configuration) en formato XML. Este documento contiene un tag principal ("EngineExecutionConf"), que a su vez contiene un tag con los datos de configuración[19] general ("ExecutionConfig"), incluyendo el nombre del layer que contiene el bounding box, el path del archivo de log como atributos y los siguientes tags (obligatorios) con la configuración para el repositorio de datos Piet:

- Tag "GISDBConfig": con la configuración de la base PostGIS (nombre del server, puerto, nombre de la base, usuario, password, driver PostGIS, método de grabado de los objetos geométricos, nombre del campo de identificación de las geometrías del mapa y tamaño del pool de la base de datos).
- Tag "OLAPDBConfig": con la configuración del data warehouse (path de la configuración OLAP, driver, usuario, password y URL de la base de datos, y path del archivo con dimensiones OLAP).

El siguiente es un ejemplo de un documento de configuración de Piet-Engine que incluye los tags mencionados.

[19] Este tipo de archivos es utilizado en la interfaz Piet-File para definir ejecuciones del ambiente Piet, agregando otros tags dentro del tag "EngineExecutionConf", como se muestra en la Sección 4.4.3.

```
<EngineExecutionConf xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="EngineExecution.xsd">
  <ExecutionConfig boundingBox="usa_bb" log4j="C:\\PreOverlay\\log.properties">
   <GISDBConfig serverName="localhost" portNumber="5432" dbName="usa_m_c2" userName="cepi"
userPassword="noembocasuna" driver="org.postgresql.Driver" saveMethod="savemethod"
idFieldPostGis="id" poolSize="3" />
   <OLAPDBConfig gisCatalog="C:\\usa_coast_map\\olap\\FoodMart.piet.xml"
jdbcDrivers="org.postgresql.Driver" jdbcPassword="aa"      jdbcUser="bb"
jdbc="jdbc:postgresql://localhost/usa_m_c2"
mondrianDimensions="C:\\usa_coast_map\\olap\\FoodMart.dimensions.xml" />
  </ExecutionConfig>
</EngineExecutionConf>
```

En las secciones que siguen a continuación se describen los servicios provistos por el módulo Piet-Engine incluidos en la Figura 15.

4.3.1 Servicio de precálculo de información

Este servicio genera información que será utilizada por los demás servicios para la resolución de consultas. El servicio recibe como entrada un bounding box y un conjunto de layers y a partir de esta información genera todas las combinaciones posibles de los layers y realiza las siguientes tareas para cada combinación (almacenando en el repositorio de datos Piet la información generada):

- Genera el carrier set asociado a cada uno de los layers de la combinación (proceso de generación de carrier sets).
- Calcula la subpoligonización de los carrier sets de los layers de la combinación, generando las subgeometrías correspondientes (proceso de subpoligonización).
- Asocia las subgeometrías generadas con las geometrías originales contenidas en los layers de la combinación (proceso de asociación de subgeometrías).
- Para cada subgeometría asociada a un componente geométrico, calcula y asigna los valores proporcionales de los hechos del componente a la parte ocupada por la subgeometría (proceso de propagación de hechos).
- Calcular el overlay de los componentes geométricos para esa combinación de layers utilizando las subgeometrías (proceso de precálculo de overlay).

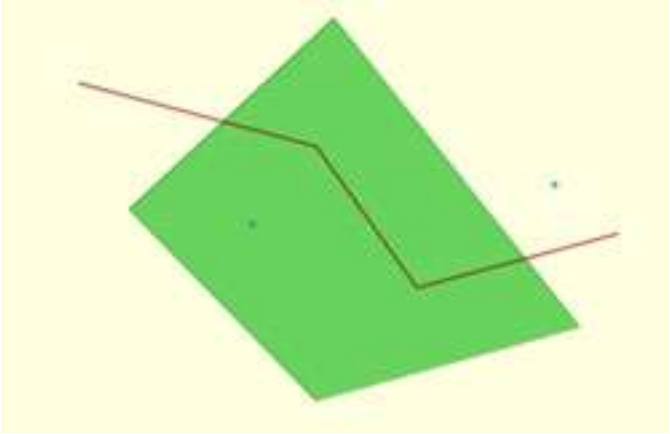


Figura 16: Mapa de ejemplo para precálculo de información.

A continuación se describe cada uno de los puntos mencionados, para una combinación de layers en particular. El proceso se repite para cada nueva combinación de layers generada, independientemente de la anterior. Para cada etapa del proceso de precálculo se ejemplifica la acción correspondiente sobre el mapa que se muestra en la Figura 16.
El mapa utilizado como ejemplo está compuesto por cuatro layers: uno conteniendo un polígono (verde), otro conteniendo una polilínea (rojo), un tercero con dos puntos (azul) y el último con el bounding box rectangular (amarillo).

Generación de carrier sets
Para cada layer se genera el carrier set correspondiente utilizando funciones de JTS y Piet-Utils, almacenando las carrier lines resultantes en una tabla en la base de datos PostGIS. Llamaremos carrier set a este conjunto de carrier lines.

La Figura 17 muestra las carrier lines generadas en esta etapa que componen el carrier set del mapa de ejemplo presentado en la Figura 16.

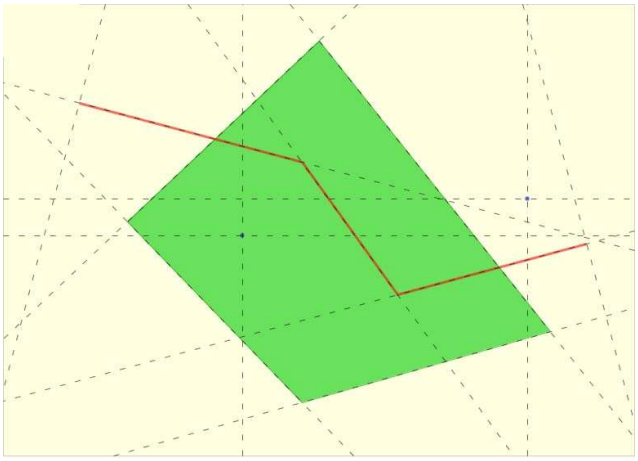


Figura 17: carrier set de mapa de ejemplo.

En el ejemplo se puede ver que las carrier lines generadas cumplen con la Definición 4 incluida en la Sección 2. En el repositorio Piet se creará una tabla tmp_carrirset_[id combinación] donde se almacenarán las carrier lines correspondientes a la combinación actual (con identificación "id combinación"), como se menciona en la Sección 3.4.

Subpoligonización
A partir del carrier set generado en el primer paso se calcula la subpoligonización, obteniendo las subgeometrías asociadas siguiendo estos pasos:

- Se interseca cada carrier line del carrier set con las demás carrier lines, obteniendo el conjunto de puntos de intersección, que corresponde a los subnodos de la subpoligonización.
- Utilizando los puntos de intersección de cada carrier line se obtienen los segmentos de línea que constituyen las sublíneas.
- Finalmente, se poligonizan las sublíneas generadas en el paso anterior, creando los subpolígonos que forman parte de la subpoligonización de este conjunto de layers.

Generadas todas las subgeometrías para esta combinación, se almacena la información en la base PostGIS del repositorio de datos Piet, utilizando una tabla para cada tipo de subgeometría, para cada combinación de layer. Debido a la gran cantidad de recursos que demanda esta etapa del proceso de precálculo de información (particularmente un gran uso de memoria), se implementaron diferentes alternativas y mejoras en la ejecución que se detallan en la Sección 5.2.2.

La Figura 18 muestra el mapa de la Figura 16 con los subpolígonos (gris) generados a partir de la subpoligonización de sus layers.

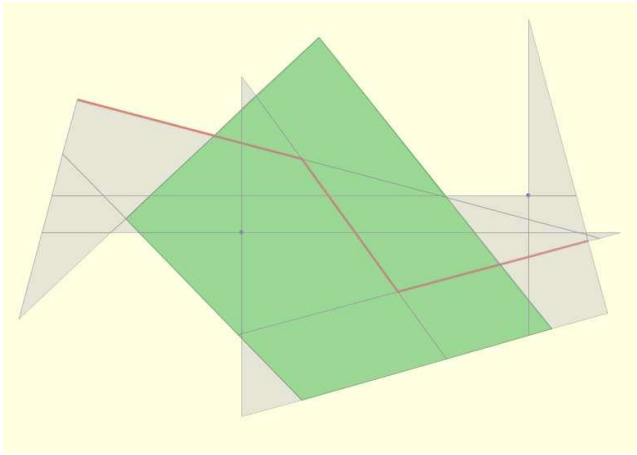


Figura 18: Subpoligonización de mapa de ejemplo.

Los subnodos, sublíneas y subpolígonos se almacenan en las tablas temporales tmp_point_[id combinación], tmp_linestring_[id combinación], tmp_polygon_[id combinación] respectivamente, como se desarrolla en la Sección 3.4.

Asociación de subgeometrías
Este proceso consiste en asociar a cada una de las geometrías de los layers seleccionados, las subgeometrías generadas en el paso anterior que se le intersecan. Es el más costoso en cuanto a tiempos de ejecución, por lo que se implementaron varias alternativas que se discuten en la Sección 5.2.3.
El proceso más simple es el secuencial, que itera los componentes geométricos de cada layer y calcula la intersección con las subgeometrías generadas en el paso anterior almacenadas en las tablas temporales.
La información de asociación obtenida se almacena en forma permanente en la base PostGIS del repositorio Piet.
Cada tipo de subgeometría asociada se almacena, según el caso, en las siguientes tablas: subp_point_[id combinación], subp_linestring_[id combinación], subp_polygon_[id combinación], como se menciona en la Sección 3.4.

Propagación de valores de hechos
Durante el proceso de asociación de subgeometrías se realiza en paralelo el proceso de "propagación de valores de hechos" que consiste en calcular, para cada subgeometría a asociar con un componente geométrico, los valores proporcionales de las funciones (hechos) asociadas a dicha geometría que corresponden a la porción ocupada por la subgeometría.

La forma en que se realiza la propagación depende del tipo de subgeometría (subnodo, subline o subpolígono) y sólo se propagan valores de funciones de geometrías del mismo tipo (p.ej., para un polígono, sólo se propagarán los valores de función a subpolígonos):

- Para los subpolígonos se utiliza el área del subpolígono comparado con el área de la geometría original.
- Para las sublíneas se utiliza el largo de la geometría original en relación al largo de la sublínea.
- En el caso de los subnodos, la propagación es constante (se asigna el mismo valor de la funciones del punto original a los subnodos asociados).

El resultado proporcional de cada función es almacenado en la relación de asociación definida en el paso anterior.

Si tuviéramos un mapa que tiene un layer con un estado (un cuadrado con "área" 4km2) y otro con una ciudad (punto), la subpoligonización de este generaría cuatro subpolígonos que serían asociados al estado. Suponiendo un hecho "población" asociado al estado con un valor de 4000, al hecho "población" de cada subpolígono asociado al estado se le propagará un valor de 1000, para la asociación con ese estado.

Precálculo del overlay
Este es el último paso del proceso de precálculo de información y consiste en precalcular el overlay de los componentes geométricos de la parte GIS del ambiente Piet utilizando las subgeometrías generadas en la subpoligonización. Para realizar esta tarea, se toma por definición que dos geometrías se superponen si tienen por lo menos una subgeometría asociada en común.

Para cada tipo de subgeometría se genera una tabla con información de las geometrías que se superponen con ese nivel de overlay, como: pre_point_[id combinación], pre_linestring_[id combinación], pre_polygon_[id combinación], como se describe en la Sección 3.4.

4.3.2 Servicio de ejecución de agregaciones geométricas

El servicio de ejecución de agregaciones geométricas permite realizar consultas geométricas y de agregación geométrica con medidas definidas sobre hechos asignados a componentes geométricos. La solución incluye una serie de operaciones denominadas Piet-Functions, que definen el tipo de resultado buscado con la consulta (p.ej., total de la medida sobre cada layer, sobre los componentes de cada layer, sobre los componentes superpuestos de cada layer, etc.). El servicio brinda también la posibilidad de asignar condiciones a las medidas utilizadas, permitiendo de esta forma expresar consultas como "Dar las descripciones de las ciudades que tengan más de 100000 habitantes y además estén cruzadas por un río".

Los datos de entrada de este servicio son: un conjunto de layers, un área de selección, un layer indicado como principal (nivel de la dimensión GIS donde se aplicará la agregación), el nombre de la medida a aplicar y el nombre de la Piet-Function. Con esta información se aplica la Piet-Function sobre el área seleccionada del conjunto de layers, utilizando la medida indicada. Los datos de precálculo usados para resolver una ejecución pueden ser definidos por el usuario o, de no ser así, se utilizará la combinación considerada óptima (la que contiene todos los layers incluidos en la lista de entrada).
El servicio genera como salida un documento con el detalle del tiempo de ejecución y los resultados según la Piet-Function seleccionada.

La ejecución de consultas de agregación geométrica se resuelve intersecando las subgeometrías que forman parte de los componentes geométricos de la lista de layers con el área de selección, sumarizando los valores de los hechos asignados a las subgeometrías para resolver el valor de la medida indicada por el usuario. Dependiendo de la Piet-Function seleccionada, también puede utilizarse la información de overlay precalculada para definir los hechos que formarán parte de los resultados finales de la consulta.

En todos los casos, los resultados de las medidas consultadas son APROXIMADOS, ya que las subgeometrías utilizadas para los cálculos pueden no adaptarse en forma exacta al área de selección. Además, la aproximación al resultado real puede variar según la combinación de subpoligonización utilizada en cada medida, como se desarrolla en la Sección 5.

La Figura 19 muestra un ejemplo donde se observa claramente que las subgeometrías que se utilizan para los cálculos (en amarillo) pueden no adaptarse a la zona de selección (verde):

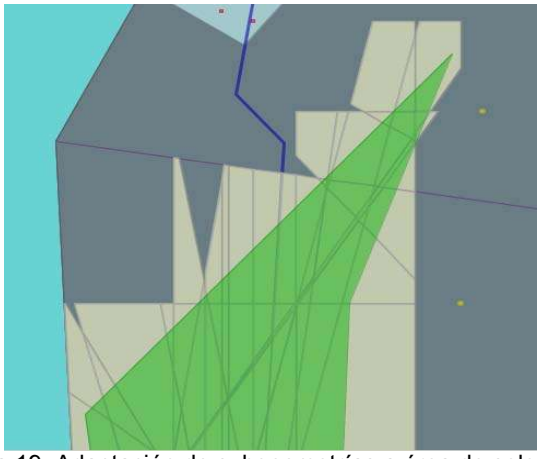


Figura 19: Adaptación de subgeometrías a área de selección.

A continuación se describe cada una de las Piet-Functions soportadas. Las primeras dos Piet-Functions calculan resultados sobre todos los layers incluidos en la lista de layers (para los layers que tienen definida la medida), mientras que las otras cuatro aplican las medidas sólo sobre el layer principal, utilizando los otros layers para resolver la parte geométrica de la consulta.

Figura 20: Mapa de ejemplo para Piet-Functions.

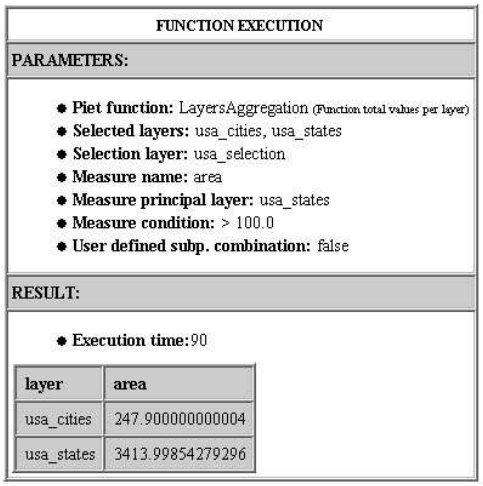
En todas las Piet-Functions se incluye un ejemplo basado en el mapa de la Figura 20, compuesto por cinco layers: un layer gris con la bounding box del mapa, un layer lila (usa_states) con dos estados, un layer rojo (usa_cities) con tres ciudades, un layer azul (usa_rivers) con un río y, finalmente, un layer verde con un área de selección que se usará para limitar el área de aplicación de las consultas.
Los layers usa_states y usa_cities tienen definidos las medidas "Área" y "Cantidad de población"[20].

Agregación total por layer: Piet-Function "LayersAggregation"
Esta operación obtiene el total de la medida seleccionada sobre cada layer de la lista tomando en cuenta, por cada layer, las partes de los componentes geométricos que se encuentran dentro del área de selección indicada

[20] Los valores asignados a los hechos son datos inventados.

al llamar al servicio. Además, los valores de las medidas deberán cumplir con las condiciones establecidas por el usuario del servicio (p.ej., "Área" mayor a 5000).
El resultado generado para cada layer incluido en la consulta contiene el valor de la medida seleccionada aplicada a cada layer. En caso de no estar definida la medida en algún layer seleccionado, este no se mostrará en el resultado.
Para resolver esta consulta Piet calcula la intersección de las subgeometrías que componen a las geometrías de cada layer (usando la dimensión GIS) con el área de selección, sumarizando por cada layer todos los valores de los hechos asignados a las subgeometrías que cumplan con las condiciones.
En esta función no se utiliza la información precalculada de overlay, ya que se busca el resultado por layer, sin relacionar uno con otro.
La combinación por defecto de layers subpoligonizados de la que se tomará las subgeometrías es la que incluye todos los layers que forman parte de la lista de layers, aunque puede ser definida por el usuario.

El resultado de la aplicación de esta operación sobre el mapa de la Figura 20, utilizando la medida "Área" con la condición "> 100", sobre los layers "usa_states" y "usa_cities" dentro del área seleccionada, sería el siguiente.



FUNCTION EXECUTION

PARAMETERS:

- **Piet function:** LayersAggregation (Function total values per layer)
- **Selected layers:** usa_cities, usa_states
- **Selection layer:** usa_selection
- **Measure name:** area
- **Measure principal layer:** usa_states
- **Measure condition:** > 100.0
- **User defined subp. combination:** false

RESULT:

- **Execution time:** 90

layer	area
usa_cities	247.900000000004
usa_states	3413.99854279296

El resultado generado muestra los parámetros de configuración de la consulta y debajo el resultado, detallando el área total (dentro de la zona seleccionada) de los layers "usa_states" y "usa_cities", debido a que ambos layers tienen definida la medida. Si el total para alguno de los layers hubiera sido menor a 100, no se incluiría en el resultado.

Agregación detallada por geometría, por layer: Piet-Function "GeometriesAggregation"
Esta operación computa el total de la medida indicada, incluyendo el detalle sobre cada componente geométrico dentro del área de la selección. Los componentes son agrupados según el layer al que corresponden. Además, los valores de las medidas deberán cumplir con las condiciones establecidas por el usuario del servicio. En caso de no estar definida la medida en algún layer seleccionado, este no será tenido en cuenta en el resultado.

La resolución de la consulta se logra en forma semejante a la consulta anterior ("LayersAggregation"), con la diferencia de que se sumariza por componente geométrico y no por layer. Aquí tampoco se utiliza el precálculo de overlay para obtener el resultado, aunque sí se usa el resto de la información precalculada.
La combinación por defecto de layers subpoligonizados de la que se tomará las subgeometrías es la que incluye todos los layers que forman parte de la lista de layers, aunque puede ser definida por el usuario.

El resultado de la aplicación de esta operación sobre el mapa de la Figura 20, utilizando la medida "Área" con la condición "> 50", sobre los layers "usa_states" y "usa_cities" dentro del área seleccionada, sería el siguiente.

FUNCTION EXECUTION

PARAMETERS:

- **Piet function:** GeometriesAggregation (Function total values per geometry, per layer)
- **Selected layers:** usa_cities, usa_states
- **Selection layer:** usa_selection
- **Measure name:** area
- **Measure principal layer:** usa_states
- **Measure condition:** > 50.0
- **User defined subp. combination:** false

RESULT:

- **Execution time:** 80

layer	name	geometrytotal
usa_cities	Ciudad 2	87.2000000000005
usa_cities	Ciudad 3	160.700000000003
usa_states	Estado 1	871.519801276564
usa_states	Estado 2	2542.47874151639

El resultado muestra el detalle de las ciudades pertenecientes a usa_cities y los estados dentro de usa_states, que están dentro del área de selección y cumplen con las condiciones. El resultado es semejante a hacer un drill-down del resultado de la consulta anterior (si se suman los resultados por layer, el resultado total es el mismo que el de la consulta LayersAggregation).
En el resultado obtenido se puede ver que no figura la Ciudad 1, debido a que no se encuentra dentro del área de selección.

Agregación total sobre layer principal aplicando overlay de layers seleccionados: Piet-Function "OverlayedLayerAggregation"
Esta Piet-Function devuelve el total de una medida aplicada sobre el layer principal, tomando en cuenta las partes de los componentes geométricos de este layer que se encuentran dentro del área de selección, sólo para los componentes que se intersecan con ALGUNA geometría de otro layer de la lista dentro de ese área. Además, los valores de las medidas deberán cumplir con las condiciones establecidas por el usuario del servicio.
Para resolver esta operación se ejecuta una consulta de overlay (sobre la información de overlay precalculada), para las combinaciones del layer principal con cada uno de los demás layers de la lista, obteniendo las subgeometrías comunes a componentes del layer principal con cada uno de los layers de la lista, que además cumplan con la condición de encontrarse dentro del área de selección.
Los componentes geométricos del layer principal que tienen asociados al menos una subgeometría devuelta por la consulta de overlay son utilizados para calcular el valor final de la medida para el layer.

Piet calcula la intersección de las subgeometrías que componen a las geometrías de cada layer (usando la dimensión GIS) con el área de selección, sumarizando por cada layer todos los valores de los hechos asignados a las subgeometrías que cumplan con las condiciones.
Para ejecutar la operación se utilizan las subgeometrías correspondientes a la subpoligonización de la lista de layers seleccionados ya que es la única que asegura que, en caso de que exista overlay, se encontrará un resultado[21].

El resultado de la aplicación de esta función sobre el mapa de la Figura 20, utilizando la medida "Cantidad de población" con la condición "> 5000", sobre los layers "usa_states" y "usa_cities" dentro del área seleccionada, sería el siguiente.

[21] Por ejemplo, si el usuario selecciona una combinación de overlay que contenga estados, ríos y sucursales, probablemente no exista información de overlay para esa combinación, ya que el overlay entre sucursales y ríos anulará el resto de los overlay.

FUNCTION EXECUTION

PARAMETERS:

- **Piet function:** OverlayedLayersAggregation (Function total for selected layer overlaped with geometries from other layers)
- **Selected layers:** usa_cities, usa_states
- **Selection layer:** usa_selection
- **Measure name:** f_pop
- **Measure principal layer:** usa_states
- **Measure condition:** > 5000.0
- **User defined subp. combination:** false

RESULT:

- **Execution time:** 671

layer	f_pop
usa_states	2604425.328719

El resultado muestra el total de la medida aplicada a sobre usa_states (layer principal). Para calcular este valor se tuvo en cuenta sólo el Estado 1, ya que el Estado 2 no se superpone con ninguna ciudad dentro del área de selección.

Agregación detallada por geometría, sobre layer principal aplicando overlay de layers seleccionados: Piet-Function "OverlayedGeometriesAggregation"

Esta Piet-Function es semejante a la anterior, pero incluye el detalle por componente geométrico dentro del layer principal.
La resolución de esta consulta se logra en forma semejante a la consulta anterior ("OverlayedLayerAggregation"), con la diferencia de que se sumariza por componente geométrico dentro del layer principal.

El resultado de la aplicación de esta operación sobre el mapa de la Figura 20, utilizando la medida "Cantidad de población" con la condición "> 5000", sobre los layers "usa_states" y "usa_cities" dentro del área seleccionada, sería el siguiente.

FUNCTION EXECUTION

PARAMETERS:

- **Piet function:** OverlayedGeometriesAggregation (Function total values per overlayed geometry for applied layer)
- **Selected layers:** usa_cities, usa_states
- **Selection layer:** usa_selection
- **Measure name:** f_pop
- **Measure principal layer:** usa_states
- **Measure condition:** > 5000.0
- **User defined subp. combination:** false

RESULT:

- **Execution time:** 80

usa_states	usa_cities	f_pop
Estado 1	Ciudad 2	2604425.328719
Estado 1	Ciudad 3	2604425.328719

El resultado es semejante al anterior, pero en este caso se incluye el detalle por cada componente geométrico perteneciente al layer "usa_states" (layer principal), incluyendo el total de la medida y los componentes

geométricos de los demás layers incluidos en la selección que se superponen. En el resultado se puede ver que no figura el Estado 2, debido a que no se superpone con ninguna ciudad dentro del área de selección.

Agregación sobre layer principal usando overlay exclusivo de componentes geométricos: Piet-Function "ExclusiveOverlayedGeometriesAggregation"
Esta Piet-Function devuelve la medida aplicada a cada componente geométrico del layer principal teniendo en cuenta sólo las partes que se intersecan con una geometría de uno de los otros layers seleccionados.
Para resolver esta operación se utiliza una lógica semejante a la de la consulta anterior, pero se sumariza únicamente las subgeometrías superpuestas con el layer principal.

El resultado de la aplicación de esta operación sobre el mapa de la Figura 20, utilizando la medida "Cantidad de población" con la condición "> 5000", sobre los layers "usa_states" y "usa_cities" dentro del área seleccionada, sería el siguiente.

FUNCTION EXECUTION

PARAMETERS:

- **Piet function:** ExclusiveOverlayedLayersAggregation (Function total values per 'exclusive' overlayed geometry)
- **Selected layers:** usa_cities, usa_states
- **Selection layer:** usa_selection
- **Measure name:** f_pop
- **Measure principal layer:** usa_states
- **Measure condition:** > 5000.0
- **User defined subp. combination:** false

RESULT:

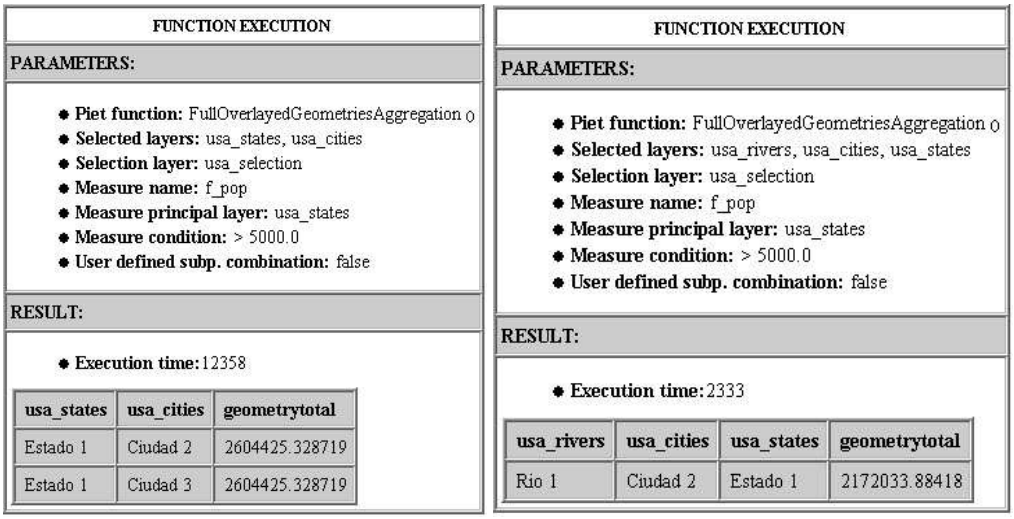
- **Execution time:** 2303

usa_cities	usa_states	geometrytotal
Ciudad 2	Estado 1	83040.392251
Ciudad 3	Estado 1	11721.159228

El resultado muestra la medida aplicada sobre las zonas ocupadas por las ciudades 1 y 2 dentro del layer usa_states. Este resultado probablemente sería diferente si se seleccionara usa_cities como layer principal, ya que en este caso se utilizaría la medida aplicada sobre este layer, y los valores de los hechos probablemente no concuerden con los calculados para las ciudades desde el layer de estados.

Agregación sobre layer principal teniendo en cuenta overlay de TODOS los layers seleccionados: Piet-Function "FullOverlayedGeometriesAggregation"
Esta Piet-Function calcula la medida por componente geométrico del layer principal, teniendo en cuenta sólo las partes de las geometrías dentro del área de selección, que además se intersecan con al menos una geometría de cada layer de la lista de layers. Además, los valores de las medidas deben cumplir con las condiciones definidas por el usuario.
Para resolver esta operación se ejecuta una consulta de overlay (sobre la información de overlay precalculada), para la combinación que contiene todos los layers seleccionados, obteniendo las subgeometrías comunes a componentes de todos los layers que además cumplan con la condición de encontrarse dentro del área de selección. Para calcular el valor de la medida para el layer principal se toman en cuenta sólo los componentes geométricos de este layer que contengan una subgeometría que cumpla con las condiciones de overlay.

A continuación se muestra los resultados de la aplicación de esta operación sobre el mapa de la Figura 20, utilizando la medida "Cantidad de población" con la condición "> 5000", primero sobre los layers "usa_states" y "usa_cities" dentro del área seleccionada, y luego agregando el layer "usa_rivers".



FUNCTION EXECUTION

PARAMETERS:

- **Piet function:** FullOverlayedGeometriesAggregation 0
- **Selected layers:** usa_states, usa_cities
- **Selection layer:** usa_selection
- **Measure name:** f_pop
- **Measure principal layer:** usa_states
- **Measure condition:** > 5000.0
- **User defined subp. combination:** false

RESULT:

- **Execution time:** 12358

usa_states	usa_cities	geometrytotal
Estado 1	Ciudad 2	2604425.328719
Estado 1	Ciudad 3	2604425.328719

FUNCTION EXECUTION

PARAMETERS:

- **Piet function:** FullOverlayedGeometriesAggregation 0
- **Selected layers:** usa_rivers, usa_cities, usa_states
- **Selection layer:** usa_selection
- **Measure name:** f_pop
- **Measure principal layer:** usa_states
- **Measure condition:** > 5000.0
- **User defined subp. combination:** false

RESULT:

- **Execution time:** 2333

usa_rivers	usa_cities	usa_states	geometrytotal
Rio 1	Ciudad 2	Estado 1	2172033.88418

En el primer resultado se muestra las ciudades que se superponen con algún componente de usa_states, dentro del área de selección. El total de la medida corresponde a su aplicación para cada estado (Estado 1 en el ejemplo) dentro del área que se superponga con una ciudad.
En el segundo resultado, al agregar el layer usa_rivers, el resultado muestra únicamente los estados que tienen alguna ciudad que además está cruzada por algún río. El resultado de la medida es nuevamente el total, dentro del área, sobre cada estado.
En estos resultados se puede observar que los valores de la medida son distintos, aunque en ambos casos se debería estar utilizando la medida sobre el Estado 1, dentro de la misma área de selección. La diferencia en el resultado se debe a la subpoligonización utilizada: en el primer caso se utiliza una subpoligonización que involucra sólo dos layers (usa_states y usa_cities), mientras que en el segundo se utiliza una combinación de tres layers (usa_rivers, usa_states y usa_cities).
Como se menciona al comienzo de la sección, la adaptación de las subgeometrías utilizadas para resolver las consultas generalmente no se adaptará al área de selección, y esto se ve aumentado en el caso de utilizar subgeometrías generadas por una combinación de pocos layers (como en el primer caso). El resultado se aproxima mejor en el segundo caso ya que al haber más subgeometrías (por utilizarse una combinación de tres layers), estás serán más pequeñas, adaptándose mejor al área de selección y mejorando la aproximación al resultado real.

4.3.3 Servicio de ejecución de sentencias GISOLAP-QL

Este servicio permite ejecutar las consultas GISOLAP-QL presentadas en la Sección 4.2.2. El servicio resuelve una consulta GISOLAP-QL recibida como entrada, devolviendo a la interfaz el resultado en un formato que dependerá del tipo de consulta realizado: MDX en el caso de consultas OLAP y GIS-OLAP, o ResultSet en el caso de consultas geométricas.
A continuación se describe en forma general la resolución general de cada una de las consultas soportadas por la sintaxis GISOLAP-QL a través de Piet-Engine.

Consultas OLAP (OLAP-Query)
Las consultas OLAP pueden escribirse completando únicamente la segunda sección de una consulta GISOLAP-QL. Una consulta de este tipo contiene únicamente texto que representa una sentencia MDX, por lo que Piet-Engine la resuelve ejecutándola directamente en el motor Mondrian. El resultado (cubo) será enviado a la interfaz Piet-Web (única interfaz que soporta este tipo de consultas) que se describe en la Sección 4.4.2, para ser presentado al usuario a través de JPivot.

Consultas geométricas (GIS-Query)
Las consultas geométricas puras pueden formularse completando únicamente la sección GIS de una consulta GISOLQP-QL.

Para resolver este tipo de consultas Piet-Engine utiliza la dimensión GIS y las definiciones del Piet-Schema para crear un SQL que consulte el repositorio de datos Piet en busca de la respuesta. El resultado se muestra en la interfaz Piet-Web.

Consultas GIS-OLAP
Las consultas GIS-OLAP se formulan por medio de una consulta GISOLQP-QL completa, con sus segmentos GIS-Query y OLAP-Query. Piet-Engine resuelve la consulta geométrica como un primer paso (de la forma en que se explica en el punto anterior). El resultado obtenido no es devuelto a la interfaz, sino que las geometrías resultantes se integran a la consulta OLAP utilizando el Piet-Schema para relacionarlas con la sentencia MDX. Como resultado se genera un MDX que se ejecuta en Mondrian y el resultado final se muestra con JPivot. Para más detalles, consultar la Sección 4.2.2.

El proceso de inserción de los resultados geométricos con la consulta OLAP-Query requiere que estén definidas asociaciones entre uno[22] de los layers resultantes de la consulta geométrica y la información del data warehouse. El proceso agrega una nueva fila (row) al MDX original, con la información de las geometrías. Debe existir una relación entre la información de esta nueva fila y el resto de la consulta MDX en los metadatos para Mondrian para que la consulta tenga sentido.

4.3.4 Servicio de generación de consultas OLAP

Este servicio permite relacionar información espacial con información de aplicación evitando la necesidad de armar una consulta GISOLAP-QL.

El servicio recibe como entrada un área de selección, un segmento MDX con la información que se desea navegar del data warehouse y un layer donde buscar las geometrías a integrar con la sentencia MDX. Durante su ejecución se comporta de forma similar a la descrita en la Sección 4.2.2: obtiene todas las geometrías del layer indicado que se encuentran dentro del área seleccionada (utilizando la función "contains" de PostGIS) y basándose en los metadatos de asociación GIS-OLAP (Piet-Schema), inserta en el segmento MDX las geometrías resultantes de la selección. La inserción de esta información genera como resultado un nuevo MDX integrado que será retornado a la interfaz[23].

Este servicio fue creado para utilizarse desde una interfaz gráfica. Por este motivo, si bien las utilidades que ofrece son limitadas en comparación con consultas GISOLAP-QL, la utilización de una interfaz gráfica facilita notablemente el armado de las consultas, dando importancia al servicio.

4.4 Interfases

En esta sección se presentan las interfases que forman parte de Piet y que permiten el acceso a los servicios provistos por Piet-Engine.

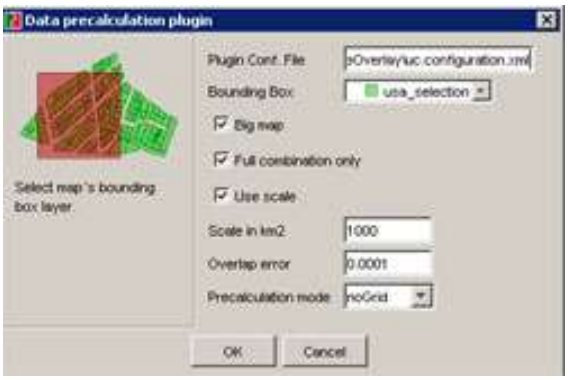
[22] La implementación actual permite la asociación de geometrías de un layer por consulta, únicamente.
[23] El usuario podrá ejecutar el MDX resultante desde la interfaz Piet-Web para obtener resultados navegables con herramientas OLAP.

4.4.1 Interfaz Piet-JUMP

La interfaz Piet-JUMP es una interfaz gráfica desarrollada en forma de plugins instalables sobre la plataforma JUMP, que permite el precálculo de información y la ejecución de consultas geométricas, de agregación geométrica (con medidas sobre atributos geométricos) y GIS-OLAP, utilizando los servicios de "precálculo de información", "ejecución de agregaciones geométricas" y "generación de consultas OLAP" provistos por Piet-Engine.

La ejecución de cada plugin genera como resultado un pequeño informe que incluye el detalle de los parámetros ingresados para la ejecución, el tiempo de ejecución y el resultado de la misma (que dependerá del tipo de plugin).

Plugin "Data precalculation"
Este plugin permite ejecutar el proceso de precálculo de información en el ambiente Piet, a través del servicio de precálculo de información de Piet-Engine.
La selección del plugin se realiza desde "Geometric Tools->Piet tools->Data precalculation", en una instalación JUMP con Piet-JUMP incorporado.
La ejecución del plugin muestra una pantalla como la siguiente.



Al ejecutar el plugin, el usuario deberá ingresar en la pantalla los siguientes datos:

- Layers seleccionados: layers que se usarán para precalcular el overlay.
- Plugin Configuration File: Archivo de configuración.
- Bounding Box: Layer con bounding box.
- Big map: indica si el mapa a procesar es "grande", como se indica en la Sección 5.
- Full combination only: indica si se quiere generar todas las combinaciones posibles de layers o sólo la combinación que incluye todos los layers.
- Use scale: indica si se utiliza una escala para el área de los polígonos.
- Scale in km2: indica cual es esa escala.
- Overlap error: indica el error de overlap que se tolerará para realizar los cálculos, como se indica en la Sección 5.
- Precalculation mode: indica el algoritmo a utilizar durante el proceso de cálculo de información. Los valores posibles son "No grid", "Partial Grid" y "Full grid". La descripción de cada alternativa corresponden a las mencionadas en la Sección 5.2.2.

La ejecución del plugin creará en el repositorio la información precalculada que será utilizada por los demás servicios que ofrece el motor, y mostrará en pantalla un reporte indicando los tiempos de ejecución y el resultado de la generación.

Plugin "Geometric Aggregation"
Este plugin permite ejecutar consultas de agregación geométrica sobre medidas aplicadas a atributos de componentes geométricos, utilizando el servicio de ejecución de funciones de Piet-Engine.
La selección del plugin se realiza desde "Geometric Tools->Piet tools->Geometric Aggregation", en una instalación JUMP con Piet-JUMP incorporado.

La ejecución del plugin muestra una pantalla como la siguiente.

Al ejecutar el plugin, el usuario deberá ingresar los siguientes datos:

- Layers seleccionados: layers a utilizar para aplicar la Piet-Function.
- Plugin Configuration File: Archivo de configuración.
- Selection Layer: Layer con área de selección.
- Fact Layer: dependiendo de la Piet-Function seleccionada, este layer será el utilizado para calcular el resultado según el hecho indicado.
- Custom subpolygonization combination: usar la combinación de layers subpoligonizada especificada por el usuario (de lo contrario se utilizará la que se considere mejor según la Piet-Function).
- Custom subpolygonization Id: Id de combinación seleccionada por el usuario.
- Show used subgeometries: indica si se quiere agregar un layer con las subgeometrías utilizadas para los cálculos de la Piet-Function seleccionada.
- Fact Name: Nombre del hecho a utilizar.
- Piet-Function: nombre de la Piet-Function a ejecutar.
- Condition symbol: símbolo utilizado para aplicar una condición a la consulta.
- Condition value: valor a aplicar junto con el símbolo anterior para limitar el resultado de la consulta.

La ejecución del plugin generará un reporte que incluye el tiempo de ejecución y el resultado de la Piet-Function. Adicionalmente, en caso de haber sido configurado, se crea un nuevo layer en el mapa con las subgeometrías utilizadas para realizar los cálculos.

Para el correcto funcionamiento de este plugin debe haberse generado y almacenado en el repositorio de datos Piet la información precalculada.

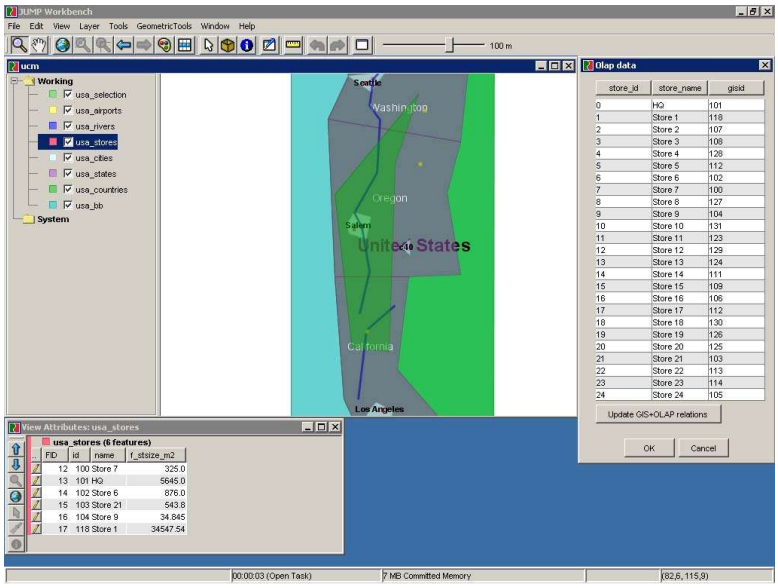
Plugin "GIS-OLAP association"

Este plugin es utilizado para asociar la información geométrica con información de aplicación. Las relaciones definidas serán utilizadas por el plugin "GISOLAP-QL partial execution" y por el módulo Piet-Web. Este es el único medio implementado a través del cual se pueden definir estas relaciones.
El plugin se ejecuta desde "Geometric Tools->Piet Tools->GIS-OLAP association", en una instalación JUMP con Piet-JUMP incorporado.
La ejecución del plugin muestra una pantalla como la siguiente.

Con la ejecución de este plugin se deben ingresar los siguientes datos:

- Plugin Configuration File: Archivo de configuración.
- OLAP table: nombre de la tabla OLAP a asociar.
- Relation Table: nombre de la tabla de relación a utilizar (en caso de no existir, se creará al almacenar las relaciones).
- ID Field: campo de identificación a usar tomado de la tabla OLAP.
- Description Field: campo de descripción tomado de la tabla OLAP.

Luego de la configuración del plugin, el usuario verá una pantalla como la siguiente.



Como se ve en la pantalla de ejemplo, el plugin despliega al usuario una tabla con el id y la descripción de la tabla OLAP (store_id y store_name en este caso) armada a partir del data warehouse, y un campo editable (gisid) donde el usuario deberá ingresar el identificador único de la geometría que corresponde a esa descripción.
Para completar la información es conveniente que el usuario despliegue el detalle de atributos del layer a relacionar (en este caso, atributos de usa_stores, a la izquierda en la figura anterior).

Al finalizar la carga de todas las relación para este conjunto de datos OLAP se debe presionar el botón de "Update GIS-OLAP relations" para actualizar la información. La información de relación generada es almacenada en la tabla definida por el usuario mediante una creación o actualización de información.

Plugin "GISOLAP-QL partial execution"
Este plugin permite utilizar el servicio de ejecución de consultas GISOLAP-QL de Piet-Engine para ejecución de consultas GIS-OLAP en forma parcial, generando un MDX que deberá ser ejecutado mediante otras utilidades (p.ej., Piet-Web).
La selección del plugin se realiza desde "Geometric Tools->Piet tools->GISOLAP-QL partial execution", en una instalación JUMP con Piet-JUMP incorporado.
La ejecución del plugin muestra una pantalla como la siguiente.

Al ejecutar el plugin, el usuario deberá ingresar los siguientes datos:

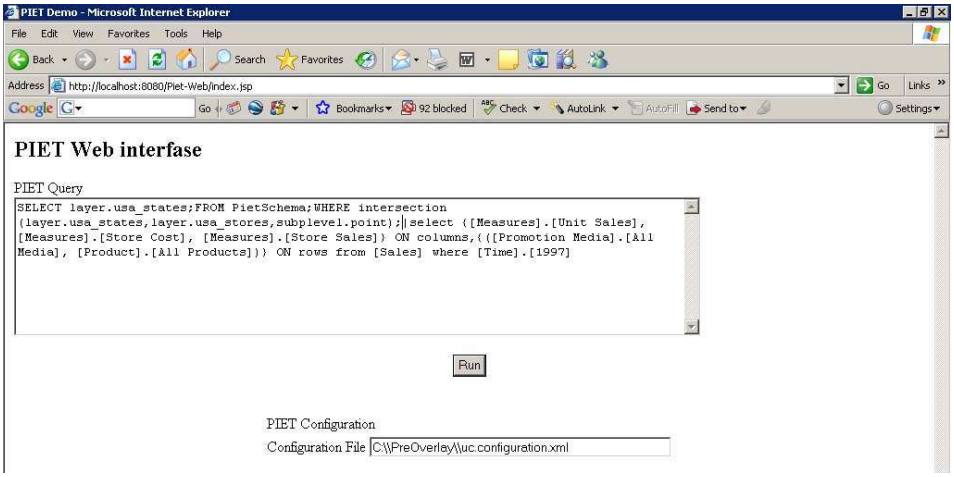
- Plugin Configuration File: Archivo de configuración.
- Selection layer: Layer con área seleccionada.
- Result layer: layer del que se obtendrán las geometrías dentro de la selección.
- MDX query: segmento de MDX que se usará para obtener el MDX integrado.

La ejecución del plugin devuelve un reporte con el detalle de los parámetros ingresados, el MDX generado integrando la información y los tiempos consumidos en el proceso de creación del resultado final.

4.4.2 Interfaz Piet-Web

Esta interfaz está implementada como una aplicación Java web que permite la ejecución de consultas OLAP y consultas GIS-OLAP por medio de sentencias GISOLAP-QL, a través del servicio de ejecución de sentencias GISOLAP-QL provisto por Piet-Engine. Además, permite utilizar en forma directa el motor PostGIS para resolver consultas sin usar Piet-Engine, utilizando SQL con las funciones geométricas provistas por PostGIS.

La ejecución de la interfaz en un explorador de Internet despliega una página como la siguiente.



El usuario debe ingresar:

- Consulta GISOLAP-QL o PostGIS[24].
- Path del archivo de configuración.

Luego de la ejecución de la consulta, el resultado se muestra en una nueva página cuyo formato dependerá de la consulta realizada:

- Consulta GIS: en este caso se muestra una tabla estática con la información del resultado.
- Consulta GIS-OLAP u OLAP: en este caso se muestra un cubo OLAP, navegable con todas las facilidades OLAP.

[24] En el caso de consultas Postgis, debe anteponerse el prefijo POSTGIS a la consulta.

- Consulta PostGIS[25]: en este caso se muestra una tabla estática con la información del resultado.

En la Sección 4.2 se muestra una variedad de pantallas de ejemplo con los resultados de las diferentes consultas realizadas a través de Piet-Web.

4.4.3 Interfaz Piet-File

Esta interfaz está implementada como una aplicación cliente Java incluida en el paquete Piet-Engine, y permite realizar el precálculo de información y ejecutar consultas geométricas, de agregación geométrica y GIS-OLAP a través de los servicios de Piet-Engine.
La configuración de una ejecución por medio de esta interfaz se realiza con documentos XML similares a los utilizados para configurar Piet-Engine, introducidos en la Sección 4.3.

La ejecución del cliente Java puede recibir los siguientes parámetros:
- Path del documento de configuración de la ejecución.
- Nombre del documento o "DIR" en caso de querer procesar todos los archivos del directorio anterior.
- Path del directorio de salida.
- Tipo de salida: "System" o "File".
- Formato de salida: "HTML" o "Text".

El resultado generado por medio de esta interfaz puede ser escrito en el Standard Output o en un archivo, en formato "texto plano" o "HTML", según sea configurado por el usuario. Además, la aplicación puede procesar un único documento de ejecución o un conjunto de documentos contenidos en un directorio.

En un documento de configuración de ejecución Piet se incluye una serie de definiciones adicionales dentro del tag "EngineExecutionConf", según el/los servicios que se quiere ejecutar.
Los tags soportados son "PreoverlayFunction", "FunctionExecution" o "PietQuery", que corresponden al precálculo de información, la ejecución de una consulta de agregación geométrica (con medidas sobre atributos de componentes geométricos) o una consulta GISOLAP-QL, respectivamente. Cada tag debe incluir información adicional similar a la ingresada por medio de los plugins de JUMP descritos en la Sección 4.4.1, "Interfaz Piet-JUMP".
Además, en la sección "ExecutionConfig" debe incluirse la lista general de tipo de hechos asociados a los componentes geométricos del mapa, incluyendo el nombre del hecho y el nivel de la jerarquía GIS en el que se aplica.

El siguiente es un ejemplo de un documento XML de ejecución de la interfaz que incluye los tres tipos de ejecuciones soportados.

[25] Consultas SQL con funciones Postgis para ser resueltas directamente por el motor PostgreSQL. Esta facilidad se incluyó especialmente para poder facilitar la comparación entre Piet y Postgis durante la experimentación.

```
<EngineExecutionConf xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="EngineExecution.xsd">
 <ExecutionConfig boundingBox="usa_bb" log4j="C:\\PreOverlay\\log.properties">
  <FunctionLevel name="area" subpLevel="polygon" />
  <FunctionLevel name="f_pop" subpLevel="polygon" />
  <GISDBConfig serverName="localhost" portNumber="5432" dbName="usa_m_c" userName="user"
userPassword="pass" driver="org.postgresql.Driver" saveMethod="savemethod"
idFieldPostGis="id" poolSize="3" />
  <OLAPDBConfig gisCatalog="C:\\ej\\FoodMart.piet.xml" jdbcDrivers="org.postgresql.Driver"
jdbcPassword="pass" jdbcUser="user" jdbc="jdbc:postgresql://localhost/usa_m_c"
mondrianDimensions="C:\\ej\\FoodMart.dimensions.xml" />
 </ExecutionConfig>

 <PreoverlayFunction fullCombination="false" mode="default" overlapError="0.0001"
scaleKm2="10000" useScale="false" bigMap="false" selectionLayer="usa_bb">
  <POSelectedLayer layerName="usa_stores" />
  <POSelectedLayer layerName="usa_states" />
  <POSelectedLayer layerName="usa_rivers" />
</PreoverlayFunction>

 <FunctionExecution functionLayer="usa_states" functionName="f_a13" subpId="4"
useFullSubp="true" pietFunction="layersAggregation" selectionLayer="usa_selection">
  <FESelectedLayer layerName="usa_stores" />
  <FESelectedLayer layerName="usa_states" />
</FunctionExecution>

 <PietQuery>
  <GisPart gisQueryValue = "POSTGIS SELECT distinct s.name as usa_stores, p.name as
usa_states, c.name as usa_cities from usa_states as p, usa_stores as s, usa_cities as c
where intersects(p.geometry, c.geometry) and intersects(c.geometry, s.geometry);" />
 </PietQuery>

</EngineExecutionConf>
```

El objetivo de esta interfaz es permitir la ejecución de servicios provistos por Piet-Engine en forma no interactiva (batch), utilizando la información definida en un documento de configuración de ejecución. La interfaz fue creada para facilitar la ejecución repetitiva de tareas sobre el motor Piet-Engine, evitando tener que recurrir al uso de interfases gráficas para esto (acelerando los tiempos necesarios para realizar la experimentación).
En el ejemplo, el objetivo del documento es llevar adelante la ejecución sucesiva de los servicios de precálculo de información (configurado en el tag "PreoverlayFunction"), posteriormente (al finalizar la ejecución del precálculo) ejecutar una consulta de agregación geométrica (configurado en el tag "FunctionExecution") y finalmente ejecutar a través de Piet-Engine una consulta PostGIS (configurada en el tag "Piet-Query").

5 Algoritmos y modelos de representación

En esta sección se presentan los algoritmos más importantes que forman parte de la solución, y los modelos de representación creados especialmente para simplificar el diseño y ejecución de estos algoritmos.

5.1 Representación de la información

Para representar geometrías en el ambiente Piet se utiliza la representación vectorial a través de objetos de tipo "Geometry" incluidos en la librería JTS. Algunos ejemplos de la representación vectorial para puntos, polilíneas y polígonos en formato de texto son:

```
POINT (378 145)
LINESTRING (191 300, 280 319, 350 272, 367 300)
POLYGON (83 215, 298 213, 204 74, 120 113, 83 215)
```

Cada componente geométrico incluye el tipo de geometría (POINT, LINESTRING y POLYGON) y una lista con los vértices que la representan, en forma de valores para las coordenadas X e Y.
Los objetos de tipo "Geometry" de JTS tienen además la capacidad de realizar algunas operaciones espaciales, como intersección, superposición y contiene, procesadas contra otro objeto del mismo tipo.

Uno de los principales problemas que surgen al trabajar con algoritmos geométricos es la pérdida de robustez, consecuencia del modelo de representación numérica finito. Los problemas de robustez aparecen cuando un cálculo numérico produce un resultado inexacto debido a errores de redondeo. En algoritmos geométricos este tipo de problemas puede ser muy grave, ya que tiende a propagarse y terminan haciendo fallar al algoritmo. En la librería JTS no se incluyen todas las operaciones necesarias para realizar el precálculo de información planteado en este trabajo, y en muchas de los algoritmos provistas por la librería se sufren problemas de robustez. Además, en algunos casos, las operaciones de JTS permiten obtener resultados luego de tiempos de ejecución muy elevados, inaceptables para el proceso de precálculo de información. Por estos motivos se diseñaron e implementaron operaciones y estructuras adicionales para:

- Facilitar operaciones con geometrías de tipo "línea".
- Representar un bounding box.
- Representación de carrier lines y ordenamiento de vértices en polilíneas.
- Facilitar la interacción con los repositorios de datos Piet (PostgreSQL y PostGIS).
- Identificar en forma unívoca objetos geométricos.
- Representar una grilla sobre un mapa.

A continuación se describe cada uno de los puntos mencionados.

Facilitar operaciones con geometrías de tipo "línea"
Las operaciones incluidas en JTS no incluyen funciones para:

- Extender un segmento de línea más allá de los definidos por el vector.
- Limitar un segmento de línea a los bordes de un bounding box.
- Generar un segmento de línea perpendicular a otro segmento dado.

Teniendo en cuenta estas limitaciones de JTS y que la representación vectorial para líneas no facilita los cálculos a realizar durante la subpoligonización (en particular, los cálculos para la generación de los carrier sets del mapa), se creó un modelo de representación de líneas utilizando ecuaciones paramétricas, definido en la clase "ParametricEcuation". Este formato fue elegido porque permite definir una línea sin imponer límites fijos (comienzo y final), facilitando de esta forma los procesos de extensión de líneas para generar las carrier lines de un polígono o una polilínea. La clase "ParametricEcuation" contiene los siguientes atributos y operaciones:

- Un punto de inicio: coordenada del extremo del vector más cercano al origen.
- Un punto de fin: coordenada del extremo del vector más alejado del origen.
- Un punto sobre la línea: coordenada sobre el vector.
- Constructor para crear una ecuación paramétrica a partir de la representación vectorial.
- Operación para obtener el punto de intersección con otro objeto de ese tipo.

- Operación para saber si una coordenada se encuentra sobre la línea.
- Operación para, dada una coordenada cercana a la línea que representa, calcular el punto exacto sobre ella. Esto permite evitar posibles errores consecuencia de realizar cálculos con representaciones finitas.
- Operación para convertir la ecuación paramétrica a representación vectorial.

Representar un bounding box
Se creo un tipo "BoundingBox" para representar el rectángulo definido por el usuario antes de procesar un mapa, que cubre todos los componentes geométricos del mismo. La particularidad de esta representación es que utiliza ecuaciones paramétricas para representar los límites del rectángulo, facilitando los cálculos con las líneas (carrier lines) que utilizan esta representación.

Una bounding box está compuesto por:
- La ecuación paramétrica de borde "Norte" del rectángulo.
- La ecuación paramétrica de borde "Sur" del rectángulo.
- La ecuación paramétrica de borde "Este" del rectángulo.
- La ecuación paramétrica de borde "Oeste" del rectángulo.

Además de los límites del bounding box, un objeto de este tipo tiene una operación que, dado un segmento de línea representado por una ecuación paramétrica (como una carrier line), retorna un nuevo objeto con los extremos del segmento de entrada extendidos hasta los límites del bounding box. Esta operación funciona calculando la intersección de la ecuación paramétrica del segmento ingresado con las ecuaciones paramétricas que representan los bordes del rectángulo del bounding box.

Representación de carrier lines y ordenamiento de vértices en polilíneas
Durante el proceso de precálculo de información el principal problema por la representación numérica finita aparece durante el cálculo de los puntos de intersección entre las carrier lines del carrier set de todos los layers, que dará origen a los subnodos, como se muestra en el algoritmo principal de precálculo de información (Sección 5.2).
Los puntos que se generan por la intersección de carrier lines pueden crearse por:
1- La intersección de un único par de carrier lines en ese punto.
2- La intersección de más de 2 carrier lines en el punto.

En el caso 1, dadas dos carrier lines L1 y L2, puede ocurrir que el punto P1 resultante de la intersección de L1 con L2 difiera (aunque esta diferencia sea muy pequeña) del punto P2, resultante de la intersección de L2 con L1. Este problema se debe a la falta de robustez de la operación de intersección provista por JTS.
Para resolver el problema se creo una clase especial llamada "CarrierLine", para representar las carrier lines. Los objetos de este tipo contienen el vector que representa la carrier line y los puntos de intersección (cortes) con las demás carrier lines creadas a partir de ese mapa. Utilizando esta representación para las carrier lines y evaluando la intersección entre carrier lines siguiendo un orden específico como el que se utiliza en el algoritmo de precálculo en la Sección 5.2, se logra solucionar el problema.
Para las carrier lines L1 y L2 del ejemplo anterior, al calcular la intersección de L1 con L2 se crearía el punto de corte P1, que se almacenaría en la lista de cortes de L1 y L2. Con esta información almacenada no será necesario calcular la intersección de L2 con L1, ya que ese corte fue almacenado.
Utilizando un orden para iterar las carrier lines tomadas como pivote para el cálculo de intersecciones, una carrier line como L1, luego de ser intersecada con todas las demás carrier lines, no volverá a ser utilizada por las carrier lines siguientes en dicho orden. Por ejemplo, al utilizar L2 como base para buscar intersecciones, esta no se evaluará contra L1.
Con esta solución se redujo el cálculo de intersección a una única operación por cada par de líneas, minimizando los tiempos de cálculo y evitando el problema de robustez de la función de intersección.

En el caso 2 puede p ocurrir que las líneas L1, L2 y L3 se intersequen en puntos P1, P2 y P3 muy cercanos entre sí pero que por problemas de representación numérica o por la definición del mapa no son exactamente iguales. En este caso, problemas de robustez en el algoritmo de poligonización de JTS hacen que la utilización de estos puntos (extremadamente cercanos) impida la creación de los polígonos que deberían crearse a partir de dichos puntos. Para solucionar este problema, la clase que representa carrier lines permite agregar cortes con otras carrier lines utilizando una función que verifica si actualmente no existe en la lista de cortes un punto "semejante" al que queremos agregar, donde con "semejante" me refiero a que esté extremadamente cerca de

alguno de los puntos de corte ya asociados con esa carrier line. En caso de que exista un corte semejante ya calculado, este es utilizado en la carrier line que generó el nuevo corte.
El pseudocódigo del método para agregar un nuevo punto de corte a la lista asociada a una carrier line sería:

```
   Lista AgregarPuntoDeCorte(Punto p, Lista listaDePuntos) {
        if (noEstaEnLaLista(p, listaDePuntos)) {
            posición = dondeInsertarPuntoEnOrden(p, listaDePuntos);
            insertarPuntoEnLista(p, listaDePuntos, posición);
        }
        return listaDePuntos;
    }
```

Donde "noEstaEnLaLista" devuelve verdadero cuando no existe ningún punto en "listaDePuntos" similar a "p". Para verificar la similitud entre un par de puntos se utiliza un margen de error, como se muestra en el siguiente pseudocódigo:

```
    boolean esPuntoSimilar(Punto p1, Punto p2) {
        return result = ((-1.0) * ERROR < p1.getX() - p2.getX() && p1.getX() - p2.getX() <
ERROR && (-1.0) * ERROR < p1.getY() - p2.getY() && p1.getY() - p2.getY() < ERROR);
    }
```

La variable estática "ERROR" corresponde al margen de error tenido en cuenta para tomar un par de puntos como similares. Un valor normal para "ERROR" sería del orden de 0.0000000001.

Por ejemplo, supongamos que tenemos las carrier lines L1, L2 y L3 cada una con el corte asociado P1 que corresponde a la intersección de L1 con L2 y L1 con L3. Además, supongamos que el algoritmo para generación de subnodos comienza a utilizar L2 como pivote (L1 ya fue utilizadas y L3 aún no). Al calcular la intersección de L2 con L3 se genera un punto P3 distinto de P1 (por problemas con la representación numérica finita), pero muy cercano a este: la operación de agregado de nuevo corte en la carrier line que representa a L2 verificaría que P3 es similar a P1 y entonces no lo agregaría en la lista de cortes (puesto que ese corte ya estaría cubierto por P1). Además, devolvería P1 como resultado del agregado, indicando de esta forma que el corte P3 debe ser reemplazado por P1 en las carrier lines involucradas. Al intentar agregar P1 a los cortes de L3, el algoritmo de agregado verificaría que el punto ya existe y daría la tarea por finalizada. De esta forma se evita la generación de cortes muy similares para distintas carrier lines, solucionando el problema mencionado.

Para facilitar la búsqueda de cortes semejantes en la lista de cortes y la siguiente etapa del proceso de subpoligonización (generación de sublíneas) los puntos son almacenados en orden (según su distancia al origen). Con esta idea, la búsqueda de cortes semejantes es más rápida ya que una vez que se encuentra en la lista un corte distinto al corte a agregar y menor a este, se puede dar por terminada la búsqueda. Además, el armado de los segmentos de línea que representarán las sublíneas se puede realizar en forma directa, tomando los cortes en orden, de a pares.

Siguiendo con el pseudocódigo de "AgregarPuntoEnLínea", "InsertarPuntoEnLista" devuelve la posición dentro de la lista donde debería insertarse el punto "p", utilizando el siguiente pseudocódigo:

```
    int dondeInsertarPuntoEnOrden(Punto p1, Lista listaDePuntos) {
        fin = false;
        posición = -1;
        for(cada puntoActual en listaDePuntos, mientras fin == false) {
                if (p1.X < (puntoActual.X + ERROR) && p1.X > (puntoActual.X - ERROR)) {
                    if (p1.Y < puntoActual.Y + ERROR) {
                        fin = true;
                        posición = i;
                    }
                } else {
                    if (p1.X < puntoActual.X + ERROR) {
                        fin = true;
                        posición = i;
                    }
                }
            }
        return posición;
      }
```

Como se puede ver en el pseudocódigo de "dondeInsertarPuntoEnOrden", la posición de "p1" (el punto de entrada) se define comparando las coordenadas X e Y con los puntos de "listaDePuntos", teniendo en cuenta que los puntos de la lista están ordenados sobre la recta según su distancia al origen.

Facilitar la interacción con los repositorios de datos Piet (PostgreSQL y PostGIS).
Para facilitar la interacción con repositorios Piet se crearon las clases "PostgisDBManager" y "PostgresDBManager". Los objetos de estos tipos son instanciados a partir del archivo de configuración del ambiente Piet.

La clase "PostgisDBManager" permite trabajar con bases de datos PostGIS, facilitando la creación de conexiones, carga y almacenamiento de geometrías y features, y asignación de identificadores únicos a las geometrías. Algunos de los métodos más importantes provistos por esta clase son:

- "identificarGeometría(Geometry g)": asigna un identificador único al objeto g.
- "obtenerFeature(String SQL)": ejecuta la sentencia SQL en la base PostGIS y convierte el resultado en una lista de features utilizando el plugin de PostGIS para JUMP.
- "obtenerGeometrias(String SQL)": ejecuta las sentencias SQL en la base PostGIS y convierte el resultado en una lista de geometrías.
- "almacenarFeatures(Lista features, String nombreTabla)": almacena los features de la lista en la tabla "nombreTabla", utilizando métodos provistos en el plugin de PostGIS para JUMP.
- "almacenarGeometrías(Lista geometrías, String nombreTabla)": almacena las geometrías de la lista en la tabla "nombreTabla", armando el insert correspondiente para cada elemento de la lista.

El método para almacenar features provisto por el plugin de PostGIS para JUMP es un método general que soporta cualquier tipo de atributos asociados a los Features. Teniendo en cuenta que los atributos de las subgeometrías generadas en la subpoligonización se limitan a un atributo para la geometría y un número acotado de atributos de tipo "double" para almacenar los valores de los hechos asociados, en este caso los tiempos de ejecución del método del plugin resultan inaceptables. Para mejorar esto se creó un método especial que genera la sentencia "insert" para cada feature de la lista sin utilizar el método general, teniendo en cuenta la estructura fija de los features que representan subgeometrías.
La solución permite obtener tiempos de respuesta aceptables, muy superiores a los que se logran usando el plugin. La desventaja de esta solución es la pérdida de flexibilidad al limitar los valores de los atributos a tipos double o geometry.

La clase "PostgresDBManager" facilita la creación y reutilización de conexiones a bases PostgreSQL, la ejecución de consultas y la verificación de existencia de las tablas. Los métodos más importantes provistos por la clase son:

- "obtenerConexión()": devuelve una conexión a la base PostgreSQL configurada.
- "liberarConexión(Conexión c)": libera la conexión c.
- "ejecutarConsulta(String consulta)": ejecuta la consulta SQL contra la base PostgreSQL configurada y devuelve el ResultSet correspondiente.
- "ejecutarActualización(String actualización)": ejecuta la actualización contra la base PostgreSQL configurada.
- "existeTabla(String nombreTabla)": verifica la existencia de "nombreTabla" en la base PostgreSQL y retorna un resultado booleano indicando la respuesta.

Para el manejo de conexiones se creo una clase llamada "PoolDeConexiones" que permite crear, liberar y reutilizar conexiones a los repositorios de datos.
Utilizando objetos de este tipo, cuando el usuario libera una conexión la misma se almacena en una lista de conexiones libres. Cuando el usuario solicita una nueva conexión al pool, este busca una conexión libre en su lista de conexiones y la devuelve. Sólo se crean conexiones nuevas en caso de que no exista ninguna conexión libre. De esta forma se logra reducir al mínimo la cantidad de conexiones abiertas contra la base de datos sin degradar la performance.

Finalmente, la clase con utilidades generales "GISOLAPTools" incluye métodos como:

- "crearGeometriesDeFeatures(Lista features)": convierte los features de entrada en una lista de objetos Geometry (eliminando la información de atributos). Este método es utilizado al obtener subgeometrías de la base PostGIS, en los casos en que no importan los valores de los atributos, sólo la geometría.
- "crearFeaturesDeGeometries(Lista geometries)": convierte la lista de geometrías de entrada en una lista de features. Este método es utilizado para poder almacenar geometrías en la base PostGIS utilizando los métodos provistos por el plugin de PostGIS para JUMP, que sólo permite almacenar features y no geometries.
- "crearGeometría(int id, String geometry)": crea un objeto "Geometry" con el id indicado y las coordenadas incluidas en geometry.
- "obtenerTipoDeSubgeometríaComún(Lista layers)": dada una lista de layers, obtiene el mayor[26] tipo de subgeometría común a todos los layers. Por ejemplo, dados una lista con los layers L1 con polígonos, L2 con polilíneas y L3 con polígonos, la función retornaría "sublínea" como mayor tipo de subgeometría común a todos los layers.

Identificar en forma unívoca objetos geométricos.
La clase Geometry incluida en JTS fue modificada para permitir la identificación unívoca de distintas instancias de ese tipo. Para ello se agregó un atributo "geometryId" de tipo "integer".

Representar una grilla sobre un mapa.
Para resolver los métodos de subpoligonización y asociación que utilizan una grilla (como se desarrolla en la Sección 5.2.2) fue necesario definir una estructura adecuada para realizar las operaciones. Para ello se creo el tipo "Grilla", compuesto principalmente por una lista de objetos de tipo "Cuadrante", y con métodos para almacenar y buscar cuadrantes en el repositorio Piet.
El tipo "Cuadrante" esta formado por tres listas con los subpolígonos, sublíneas y subnodos correspondientes a la porción del mapa que ocupa.

5.2 Algoritmos para precálculo de información

5.2.1 Algoritmo principal

El algoritmo principal para el precálculo de información es el que se encarga de realizar todos los pasos descritos en la Sección 4.3.1, calculando entre otras cosas la subpoligonización de los layers del mapa, generando subgeometrías, asociando las subgeometrías a componentes geométricos del mapa, propagando los valores de hechos de estas geometrías a las subgeometrías asociadas y precalculando el overlay entre los componentes geométricos.
El algoritmo general para el precálculo de información es el siguiente:

```
precalcularInformación(Lista layersSeleccionados) {
        componentesGeométricos = obtenerComponentesGeometricos(layersSeleccionados);

        carrierSet = generarCarrierSet(componentesGeométricos);

        subNodos = generarSubNodos(carrierSet);

        subLineas = generarSubLíneas(carrierSet);

        subPolígonos = generatarSubpolígonos(subLíneas);

        subGeometrías = crearListaDeSubgeometrías(subNodos, subLíneas, subPolígonos);

        Para cada geometría en "componentesGeométricos" {
                asociarSubgeometríasYHechos(geometría, subGeometrias);
        }

        precalcularOverlay(componentesGeométricos);
}
```

[26] Mayor se refiere en este caso a la subgeometría de mayor complejidad. Subpolígono, sublínea y subnodo sería el orden de mayor a menor.

Los puntos más importantes de este algoritmo se describen a continuación.

Generación de carrier set
Esta etapa del proceso de generación de información crea una lista con las carrier lines que componen el carrier set de los componentes geométricos contenidos en los layers seleccionados, utilizando la representación de carrier lines presentada en la Sección 5.1 y la Definición 7.

Si se siguiera estrictamente la definición de carrier lines y su generación, podría suceder que se generaran carrier lines repetidas (iguales o muy similares) a partir de componentes geométricos de layers diferentes. Por ejemplo, si un río sirve de delimitador de dos estados, al superponer los carrier sets de los layers "ríos" y "estados" los componentes geométricos que representan el río y los estados mencionados darán origen a carrier lines repetidas.
La utilización de carrier lines repetidas perjudica el proceso de precálculo ya que impide el correcto funcionamiento del algoritmo de poligonización de JTS.
Para solucionar este problema se ideó un método que se ejecuta al finalizar la creación de las carrier lines de todos los carrier sets del mapa, y que se encarga de eliminar las líneas repetidas de la lista de carrier lines. El método itera la lista de carrier lines tomando las carrier lines de a una, como pivote, y comparando sus extremos con el de las demás, utilizando la función "esPuntoSimilar"[27] que retorna "verdadero" si los puntos consultados se encuentran a una distancia menor que cierta cota. En caso de encontrar carrier lines "similares" al pivote, se eliminan de la lista. De esta forma, la lista final no contiene carrier lines semejantes.

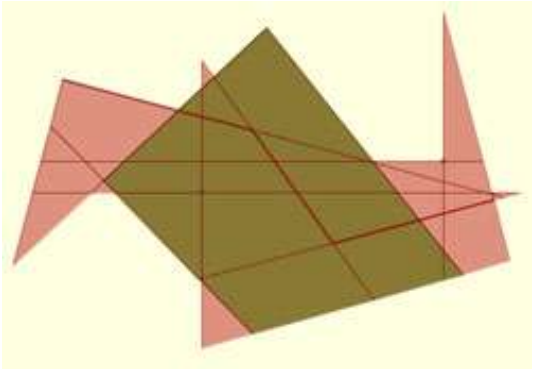
Subpoligonización y generación de subgeometrías
La generación de subnodos de la subpoligonización se realiza calculando la intersección entre las diferentes carrier lines, almacenando los cortes para cada una utilizando la representación presentada en la Sección 5.1.

La creación de sublíneas se realiza utilizando directamente los cortes almacenados en cada carrier line, tomándolos de a pares para dar origen a los segmentos que representarán las sublíneas.

Los subpolígonos se generan utilizando la clase "Polygonizer" provista en la librería JTS, con las sublíneas generadas en el paso anterior como entrada. La forma en que se generan las sublíneas de entrada asegura que se cumplan con los requisitos para el correcto funcionamiento de un objeto "Polygonizer": no existen líneas repetidas o extremadamente similares, ni tampoco existen líneas "sueltas" (todas las líneas incluidas en la lista de sublíneas de entrada formarán parte de algún polígono).

Una optimización realizada para reducir los cálculos de la generación de subpolígonos, es que en las sublíneas enviadas al "Poligonizer" para generar los subpolígonos no se incluyen las líneas que componen el bounding box, debido a que estas líneas no se relacionan con elementos del mapa.

La siguiente figura es un ejemplo de los subpolígonos generados sobre el mapa de la Figura 16.



[27] El pseudocódigo de la función se incluye en la Sección 5.1.

En el ejemplo se puede ver que los bordes del bounding box, en combinación con sublíneas generadas a partir de los carrier sets, darían origen a subpolígonos en el área amarilla (bounding box) de la figura, con lo cual dichas subpolígonos serían "externos" (no se superpondrán con componentes geométricos, y por lo tanto no serán asociados) a los componentes geométricos que forman el mapa. La generación de estos subpolígonos "externos" supondría un mayor tiempo de cálculo y agregaría una carga extra a los algoritmos de asociación, ya que tendrían que verificar subpolígonos que de antemano se sabe no formarán parte de ninguna geometría.

Asociación de subgeometrías a componentes geométricos

El paso siguiente a la creación de subgeometrías usando la subpoligonización es la asociación de las subgeometrías creadas con los componentes geométricos, reorganizando la dimensión GIS.
En este punto nuevamente, por limitación en la robustez de las funciones JTS [24] para obtener superposiciones e intersecciones, debieron implementarse operaciones especiales para calcular las asociaciones de cada componente geométrico con los diferentes tipos de subgeometrías.

En el caso de los subpolígonos, la asociación está definida según el siguiente pseudocódigo:

```
obtenerSubpolígonosRelacionados(Feature f) {
        for(cada potencial subpolígono a asociar) {
                if (sonPolígonosSuperpuestos(subpolígonoActual, f.geometría())) {
                        agregar(subpolígonoActual, listaSubpRelacionados)
                }
        }
}
```

Los potenciales subpolígonos a evaluar en el bloque "for" dependerán del método elegido para generar la subpoligonización, como se menciona en la Sección 5.2.2.

Por limitaciones consecuencia de la representación finita, puede suceder que subpolígonos adyacentes (que están pegados uno con otro pero que no se superponen), se intersequen. Esta situación puede hacer que la función de intersección provista por JTS devuelva verdadero para un componente geométrico y el grupo de subpolígonos adyacentes al mismo, cuando en realidad no debería ser así. Para solucionar esto, la operación "sonPolígonosSuperpuestos(Geometry subPolígonoActual, Geometry componenteGeométricoOriginal)" reemplaza la utilización directa de la función de intersección provista por JTS por el siguiente pseudocódigo:

```
boolean sonPolígonosSuperpuestos(Geometry p1, Geometry p2) {
        double áreaDeSuperposición = obtenerAreaDeSuperposición(p1, p2);
        return (áreaDeSuperposición > ERRORSUPERPOSICIÓN);
}
```

En esta función, "obtenerAreaDeSuperposición(Geometry p1, Geometry p2)" retorna el área de superposición entre las geometrías p1 y p2 calculada con JTS. La falta de robustez del algoritmo de intersección hace que, en caso de excepciones, se deba aplicar un buffer[28] sobre p1 para calcular la intersección, permitiendo continuar con el procesamiento.
El área resultante se compara con un error de superposición que indica el valor a partir del cual un par de geometrías se considerar realmente superpuesta. En el caso de los mapas con un nivel de detalle extremadamente alto, el valor de la variable ERRORSUPERPOSICIÓN" se define como 0. Igualmente, Piet está preparado para que los administradores modifiquen este valor en caso de que los resultados obtenidos no sean los esperados (a través de los parámetros de configuración mencionados en la Sección 4.4).
Utilizando este método, los tiempos para calcular las asociaciones de subpolígonos son mayores en comparación con la utilización directa de la función de intersección, aunque de esta forma se evitan errores en los resultados.
En el caso de la asociación de sublíneas y subnodos, la operación de asociación se realiza utilizando un buffer sobre el componente geométrico original, para evitar problemas similares a los de los subpolígonos. Debido a la simplicidad de estos tipos de geometría, los tiempos no se ven sensiblemente afectados al utilizar este método.

[28] La función buffer de JTS aplicada a una geometría g y una distancia d, retorna una geometría g' que representa todos los puntos cuya distancia a g es menor o igual a d.

Propagación de hechos a subgeometrías
Durante la asociación de subgeometrías se realiza también la propagación de valores de hechos asociados a los componentes geométricos. Esta propagación se realiza sólo para aquellas subgeometrías que son del mismo tipo que el componente geométrico. La propagación para cada tipo de componente geométrico es la siguiente:

- Polígono: los hechos sólo se propagarán a los subpolígonos asociados (no a las sublíneas y subnodos, ya que esto no tendría sentido). El valor a propagar para cada hecho, para cada subpolígono, se define a partir del área del componente original y el del subpolígono evaluado.
- Polilínea: los hechos sólo se propagarán a las sublíneas asociadas. En este caso, el valor a propagar se define utilizando la longitud del componente original y la de cada sublínea asociada.
- Punto: los hechos sólo se propagarán al subnodo asociado. El valor a propagar es constante, asignando el valor del hecho original (del componente geométrico) directamente al subnodo asociado.

El valor del área de un polígono puede definirse en un atributo "área" de tipo "double", que se interpretará como el área en km2 de la geometría y se utilizará para la propagación de hechos correspondientes a dicha geometría. En el caso de que no esté definido ese atributo, la implementación tomará el área de la geometría del mapa para realizar la propagación de hechos. Opcionalmente, el administrador Piet podrá definir una escala que se multiplicará por el área de la geometría en el mapa para obtener el área real de la geometría, como se menciona en la Sección 4.4.

La propagación de hechos para subpolígonos se realiza utilizando la siguiente fórmula:

```
Resultado = (ARS * VTF) / ARG;
```

Donde:

```
ARS = Área real del subpolígono.
VTF = Valor total del hecho en la geometría original.
ARG = Área real de la geometría.
```

Precálculo de overlay
Este algoritmo precalcula la superposición entre los diferentes componentes geométricos que forman parte del mapa. Para realizar los cálculos se basa en la subpoligonización, lo que permite resolver la superposición y además calcular fácilmente las geometrías que representan el área de intersección y las subgeometrías incluidas en esta área.
Dados los layers Y1 e Y2 de un mapa y basados en la estructura del repositorio de datos que se desarrolla en la sección 4.3, el algoritmo funciona de la siguiente forma:

- Utilizando sentencias SQL se busca en las tablas de asociación de subgeometrías, los identificadores únicos de las subgeometrías asociadas a más de un componente geométrico, para los componentes de los layers Y1 e Y2, para cada tipo de subgeometría. La consulta retornará identificadores de subgeometrías de ese tipo asociadas únicamente a pares de componentes geométricos de los layers que se superponen.
- El resultado de la consulta es almacenado en una tabla de precálculo de overlay para los layers Y1 e Y2 con el nivel de subgeometría utilizado, incluyendo los identificadores de los componentes geométricos que se superponen y las subgeometrías que forman parte de esa superposición.

El proceso de precálculo se repite de la misma manera para todas las combinaciones posibles de los layers, para cada tipo de subgeometría.

Al finalizar el precálculo el repositorio de datos Piet tendrá una tabla de precálculo por cada combinación de layers posible, por cada tipo de subgeometría, como se explica en la Sección 3.4. Una cantidad importante de las tablas creadas no contendrán información, ya que al calcular el overlay simultáneo entre los layers muchas de las combinaciones no presentarán superposición alguna.

En el mapa de ejemplo de la Figura 16, suponiendo que los layers se llaman "ej_puntos", "ej_lineas" y "ej_poligonos", y que el polígono tiene identificación 1, la polilínea identificación 5, y los puntos identificación 3 y 4, la situación final en el repositorio de datos respecto del precálculo del overlay para la combinación del layer "ej_poligonos" con el layer "ej_lineas" (combinación con identificación 3) usando las subgeometrías generadas para esa combinación, sería la siguiente:

pre_point_3

uniqueid int4	ej_lineas int4	ej_poligonos int4
440	5	1
444	5	1
446	5	1
450	5	1

pre_linestring_3

uniqueid int4	ej_lineas int4	ej_poligonos int4
412	5	1
417	5	1
420	5	1

pre_polygon_3

uniqueid int4	ej_lineas int4	ej_poligonos int4

La información de overlay de geometrías para la combinación 3 utilizando subnodos se almacena en la tabla "pre_point_3". En este caso se superponen la geometría 5 (polilínea que se encuentra en el layer "ej_lineas") y la 1 (polígono que se encuentra en "ej_poligonos"), sobre 4 subnodos en común (superpuestos con las geometrías 5 y 1), con identificación única 440, 444, 446 y 450, respectivamente.
En la tabla "pre_linestring_3" se incluye el precálculo del overlay utilizando sublíneas: se superponen las mismas geometrías que en el caso anterior, pero en este caso coincidiendo en 3 sublíneas identificadas como 412, 417 y 420.
Finalmente, en la tabla "pre_polygon_3" se incluye la superposición de ambos layers a nivel subpolígono. El resultado es vacío debido a que en la comparación se utilizan polilíneas, que no pueden tener subpolígonos asociados. El resultado en estos casos siempre será un conjunto vacío, sin importar el tipo de geometrías con el que se busca la superposición.

En el ejemplo planteado, el precálculo de overlay sobre la combinación que incluye los tres layers da como resultado tres tablas vacías (una por cada tipo de subgeometría). Esta situación se debe a que:

- En el caso de buscar superposición por subnodos no se intersecan "simultáneamente" las cuatro geometrías del mapa en ningún punto.
- En el caso de buscar superposición por sublínea y subpolígono, no se generará información de overlay, ya que en la combinación se están incluyendo geometrías de tipo punto, que no pueden tener nunca asociados subpolígonos ni sublíneas. Esto hace que el resultado también sea vacío en estos casos.

5.2.2 Alternativas para subpoligonización

La subpoligonización de un mapa requiere una gran cantidad de memoria y espacio en disco, por lo que se evaluaron e implementaron diferentes alternativas para llevarla adelante. En nuestra implementación, la selección del método a utilizar es realizada por el administrador Piet durante la configuración de la ejecución.

A continuación se describen las diferentes alternativas incluidas en Piet.

Subpoligonización utilizando funciones de JTS
Este método está basado en una utilidad provista por JTS que permite armar en forma dinámica una "nodedList" (según la terminología JTS), que es una geometría compuesta por un conjunto de líneas y sus puntos de corte. Para realizar todos los cálculos de este método se utilizan funciones provistas por JTS.

El proceso comienza armando una "nodedList" a partir de un par de carrier lines. Por la definición de "nodedList", esta nueva geometría contendrá segmentos de línea y nodos que corresponderían a sublíneas y subnodos, si sólo formaran parte del mapa el par de carrier lines utilizados al comienzo. A partir de ahí, se van agregando a la "nodedList" una por una todas las carrier lines, aumentando a cada paso el tamaño de la misma. Finalizado el agregado de todas las carrier lines, se ejecuta la poligonización de JTS con la "nodedList" como parámetro para crear los subnodos correspondientes. Al terminar la ejecución, se extraen los nodos y segmentos de línea definidos en la "nodedList", para dar origen a los subnodos y sublíneas de la subpoligonización. En este punto se ejecutan procedimientos para eliminar elementos repetidos que pueden existir en la "nodedList".

Esta alternativa tiene serios problemas de memoria en mapas grandes, ya que se procesa el mapa completo para crear las subgeometrías. Sin embargo, el método funciona muy bien en mapas medianos y chicos, por lo que es el que se ejecuta por defecto (en caso de que el administrador Piet no seleccione un método alternativo).

Subpoligonización utilizando grilla
Esta alternativa se basa en la creación de una grilla para dividir el mapa en sectores más pequeños, dividiendo la tarea de subpoligonizar el mapa completo en tareas más pequeñas (divide&conquer), subpoligonizando cada cuadrante de la grilla por separado. Con esta alternativa se busca reducir los problemas de memoria que aparecen en la alternativa anterior.

La grilla es generada durante la creación de las carrier lines, tomando como delimitadores de cada cuadrante las carrier lines que se crean a partir de puntos del mapa (por definición, carrier lines horizontales y verticales). Luego de la creación de la grilla, se asocia a cada cuadrante los componentes geométricos relacionados (superpuestos con él).
Existen dos variantes para generar la subpoligonización utilizando la grilla. Las alternativas utilizan la grilla para:

- Generar todas las subgeometrías del mapa: se asignan a cada cuadrante las porciones de carrier lines que están dentro de él, y a continuación se subpoligoniza cada cuadrante como si fuera un pequeño mapa, utilizando los segmentos de carrier lines como carrier lines completas. Para realizar esta subpoligonización se puede utilizar el método anterior ("Subpoligonización utilizando funciones JTS").
- Generar únicamente los subpolígonos: los subnodos y sublíneas se generan utilizando el algoritmo por defecto (funciones JTS), mientras que para generar los subpolígonos se utilizan los cuadrantes, por ser el proceso que trae problemas de memoria. A medida que se generan los subnodos y sublíneas, estos son asociados con cada cuadrante. Al finalizar este proceso, se ejecuta la poligonización de cada cuadrante como si fuera un pequeño mapa independiente.

En la Figura 21 se muestra un ejemplo del mapa de la Figura 16 con la grilla (gris) generada sobre él. Los cuadrantes generados están delimitados por líneas grises (correspondientes a los carrier sets de los puntos del mapa original).

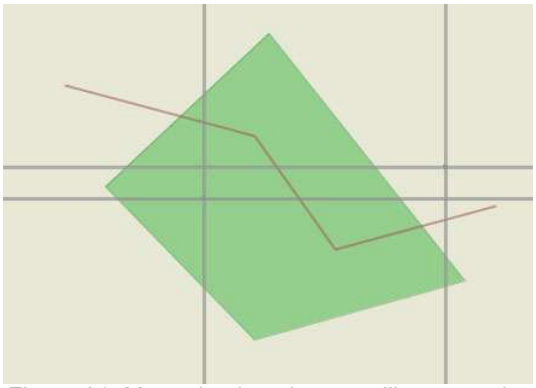


Figura 21: Mapa de ejemplo con grilla generada.

La implementación de la primera de las alternativas no fue incluida finalmente en Piet, ya que pueden surgir serios problemas en los cálculos (particularmente en los bordes de los cuadrantes) que hacen que el algoritmo de poligonización de JTS no genere todos los polígonos que debiera, produciendo la falla del proceso de subpoligonización del mapa completo. No se consideró conveniente mejorar esta alternativa ya que la segunda de las alternativas propuestas resuelve la subpoligonización en un tiempo aceptable.
Sin embargo, sería conveniente avanzar con esta alternativa en caso de agregar soporte de procesamiento paralelo, ya que esto permitiría procesar cada cuadrante en un procesador separado, obteniendo en teoría una mejora sustancial respecto a las demás alternativas de subpoligonización.

Las subgeometrías generadas utilizando los métodos con grilla son almacenadas en el repositorio de datos Piet, incluyendo la relación con los diferentes cuadrantes. Esta información será utilizada durante la etapa de asociación de subgeometrías, como se menciona en la Sección 5.2.1.

Subpoligonización utilizando algoritmo tipo "quicksort"
Esta alternativa intenta mejorar la performance del armado de la "nodedList" de la alternativa por defecto ("Subpoligonización utilizando funciones JTS"), basándose en la idea de un algoritmo tipo "quicksort". El

método consiste en partir a la mitad la lista de carrier lines y llamar en forma recursiva al algoritmo de creación de la "nodedList" con cada una de las mitades. El caso base crea una "nodedList" con dos carrier lines. Debido a las llamadas recursivas, esta alternativa produce mayores problemas de memoria que la alternativa por defecto, por lo que no fue incluida en la implementación Piet final.

5.2.3 Alternativas para asociación de subgeometrías

La asociación de subgeometrías es la etapa que mayor cantidad de memoria y tiempo requiere. Por este motivo se implementaron diferentes alternativas para poder ejecutarla en forma eficiente.

La definición de la alternativa a utilizar en una ejecución se realiza en forma automática, a partir del método de subpoligonización seleccionado por el usuario administrador Piet.

A continuación se describen las alternativas implementadas.

Asociación secuencial
Este método consiste en buscar en forma secuencial las subgeometrías a asociar a cada componente geométrico. Por cada componente se verifica la relación con todas las subgeometrías creadas en la subpoligonización, utilizando las operaciones JTS de intersección y superposición con las modificaciones presentadas en la Sección 5.1.
Este método de asociación se ejecuta automáticamente cuando se selecciona como proceso de subpoligonización la "subpoligonización utilizando funciones JTS".

Las ventajas de esta solución son:
- Es un método muy simple, que para mapas medianos y chicos no presenta problemas de memoria durante la ejecución.
- A diferencia de los métodos con grilla (que se detallan a continuación), no requiere trabajo adicional durante la generación de subgeometrías, por lo que los tiempos totales de procesamiento suelen ser muy buenos.

Sus desventajas aparecen al procesar mapas de gran tamaño:
- Para mapas grandes se transforma en el método que más tiempo demora, porque busca asociaciones para cada componente geométrico revisando una por una todas las subgeometrías del mapa, por cada componente.
- El método trabaja con las subgeometrías en memoria, lo que hace que cuando se utilizan mapas grandes la memoria física sea insuficiente para soportar esta carga, haciendo que la ejecución termine abruptamente.

Asociación utilizando grilla
Esta alternativa se utiliza cuando el método de subpoligonización elegido es "Subpoligonización utilizando grilla".
La grilla creada durante la subpoligonización es utilizada para reducir el campo de búsqueda para la generación de asociaciones de subgeometrías a componentes geométricos.
El método funciona basándose en que cada cuadrante de la grilla tiene asociado las subgeometrías y componentes geométricos que se superponen con él. Utilizando esta información, para cada componente geométrico se buscan asociaciones con subgeometrías únicamente de los cuadrantes que se superponen con él, reduciendo sensiblemente el campo de búsqueda, con la consecuente reducción en la utilización de memoria y grandes mejoras en los tiempos que toma el procesamiento.

La ventaja de esta alternativa es su velocidad en comparación con la anterior.

Las desventajas de este método son:
- Se utiliza un mayor espacio de almacenamiento: se almacenan los cuadrantes de la grilla y, para cada cuadrante, sus relaciones con subgeometrías y componentes geométricos del mapa.

- Aumento considerable de tiempos para procesamiento previo a la asociación: el tiempo requerido para llevar adelante los pasos anteriores a la etapa de asociación de subgeometrías se incrementan en forma importante, ya que se debe calcular la superposición de subgeometrías y componentes geométricos con los cuadrantes, y almacenar toda esta información en el repositorio Piet. Esta desventaja pierde importancia al procesar mapas grandes, en los que este tiempo de overhead puede ser de menor magnitud al tiempo ganado para realizar la asociación de subgeometrías a componentes geométricos utilizando sólo la información de los cuadrantes.

6 Experimentación

A efectos de comprobar el comportamiento de nuestra solución, ejecutamos una serie de experimentos, con distintos objetivos puntuales. Estas pruebas fueron realizadas sobre mapas de diferentes tamaños y complejidad, utilizando diferentes configuraciones y alternativas. Para conformar la parte de aplicación del ambiente Piet utilizado para las pruebas se usó el data warehouse "FoodMart" incluido en el proyecto Mondrian. En la presente se describen las pruebas realizadas y los resultados obtenidos.

6.1 Objetivos de la experimentación

Los principales objetivos de los experimentos fueron:

- Verificar que los tiempos de precálculo y armado del ambiente Piet sean compatibles con un uso razonable de la solución en un marco realista.
- Verificar que el espacio ocupado por la información precalculada sea razonable.
- Comparar los tiempos de respuesta obtenidos a partir de nuestra propuesta con los obtenidos mediante el uso de sistemas GIS existentes, que utilicen otros métodos de procesamiento de consultas (es decir, comparar básicamente R-Trees con el mecanismo de precómputo del overlay).
- Mostrar que la solución permite ejecutar consultas de agregación geométrica con medidas definidas sobre atributos geométricos sin necesidad de realizar ninguna programación adicional, manteniendo tiempos de respuesta aceptables.
- Probar que la solución permite ejecutar consultas que integren información de la parte geométrica del modelo con información de la parte de aplicación. Además, que los tiempos de respuesta para resolver este tipo de consultas sean razonables y que los resultados puedan navegarse con herramientas OLAP.
- Probar que la solución permite ejecutar consultas OLAP sobre la parte de aplicación del modelo.

6.2 Ambiente para experimentación: hardware y mapas

La máquina utilizada para realizar la experimentación tiene las siguientes características:

- 4 procesadores Intel(R) Xeon(TM) CPU 3.60GHz.
- 26 Gb. de disco libre para realizar las pruebas.
- 8 Gb. de memoria.

Las pruebas se realizaron sobre cuatro mapas diferentes: tres fueron dibujados en forma manual (los llamaremos mapas USA), mientras que el cuarto es un mapa real (mapa Nevada). Los mapas dibujados representan partes del territorio de Estados Unidos y varían significativamente en tamaño (cantidad de geometrías y complejidad de las mismas).

Los siguientes son los tres mapas USA: el mapa chico (costa USA), mapa mediano (oeste USA) y mapa grande (USA completo), y se muestran, de izquierda a derecha, en la Figura 22**.**

Figura 22: mapas USA.

Los mapas están compuestos por siete layers, y las geometrías de cada uno tienen asociadas una serie de hechos en sus atributos. La descripción de los layers y hechos es:

- Bounding box (celeste), sin hechos definidos.
- Países (verde), sin hechos definidos ya que sirve sólo a modo de referencia.
- Estados (gris), con los hechos:
  - area: área del estado.
  - f_pop: población del estado.
  - f_a13: promedio de ingresos de la población del condado.
  - f_a34: promedio de trabajos generados en el condado.
  - f_male: población masculina del estado.
  - f_female: población femenina del estado.
  - f_under18: población menor de 18 años.
  - f_medage: población entre 18 y 65 años.
  - f_perover65: población mayor a 65.
- Ciudades (celeste), con los hechos:
  - f_edu_level: nivel de educación promedio en la ciudad.
  - f_pop_km2: población por kilómetro cuadrado.
  - area: área de la ciudad.
- Ríos (azul), con los hechos:
  - f_avgdisc_m3s: metros cúbicos de agua por segundo.
- Sucursales (rojo), con los hechos:
  - f_stsize_m2: tamaño de la sucursal en metros cuadrados.
- Aeropuertos (amarillo), con los hechos:
  - f_tot_enp: promedio anual de vuelos que salen del aeropuerto.

Las pruebas sobre un mapa real se realizaron sobre un mapa[29] representando el Noroeste del estado de Nevada de los Estados Unidos. El mapa tiene un alto nivel de detalle y una serie de hechos reales asociados a través de atributos.

La Figura 23 muestra una imagen de este mapa.

[29] La información para armar el mapa fue obtenida del sitio de Internet con URL: http://www.nationalatlas.gov/.

Figura 23: Mapa Nevada.

El mapa está compuesto por cuatro layers, y las geometrías de cada uno tienen asociadas una serie de hechos en sus atributos. La descripción de los layers y hechos es:

- Condados (verde), con hechos:
  - area: área del condado.
  - f_pop: población del condado.
  - f_a13: promedio de ingresos de la población del condado.
  - f_a34: promedio de trabajos generados en el condado.
  - f_male: población masculina del condado.
  - f_female: población femenina del condado
  - f_under18: población menor de 18 años.
  - f_medage: población entre 18 y 65 años.
  - f_perover65: población mayor a 65.
- Ferrocarriles (rojo), con los hechos:
  - f_length: largo de las vías del ferrocarril.
- Sucursales (amarillo), con los hechos:
  - f_stsize_m2: tamaño de la sucursal en metros cuadrados.
- Aeropuertos (negro), con los hechos:
  - f_tot_enp: promedio anual de vuelos que salen del aeropuerto.

En la Figura 24 se muestra una comparación, para cada mapa, de la cantidad de geometrías por tipo de componente geométrico (puntos, polilíneas y polígonos) teniendo en cuenta todos los layers.

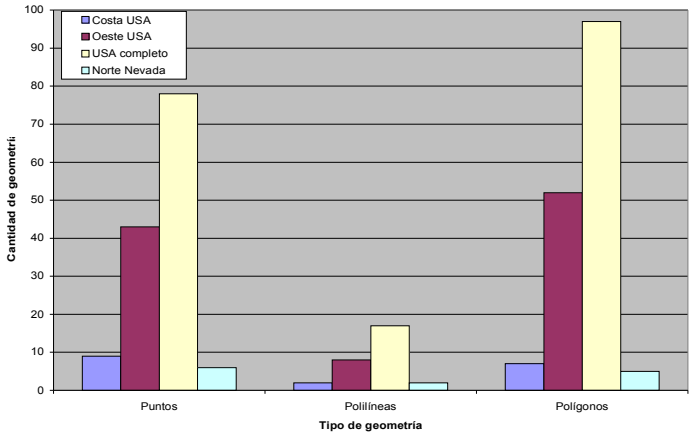


Figura 24: Cantidad de geometrías de cada tipo, por mapa.

Observando el resultado vemos que la cantidad de geometrías para el mapa "USA completo" duplica la cantidad de geometrías por componente geométrico del mapa anterior mediano. Respecto del mapa de Nevada, se puede ver que tiene una cantidad muy pequeña de geometrías para cada tipo de componente geométrico (incluso menor que las de "Costa USA"). Esto último hace pensar que la cantidad de geometrías contenidas en un mapa no es un indicador que refleje realmente la complejidad de dicho mapa para el tipo de cálculo que se realizarán en un ambiente Piet. El total de vértices por tipo de geometrías parece ser un indicador más significativo, teniendo en cuenta que está íntimamente relacionado con la generación de la subpoligonización (a través del uso de los carrier sets de los layers del mapa).

En la Figura 25 se compara el total de vértices por tipo de geometría, para cada mapa.

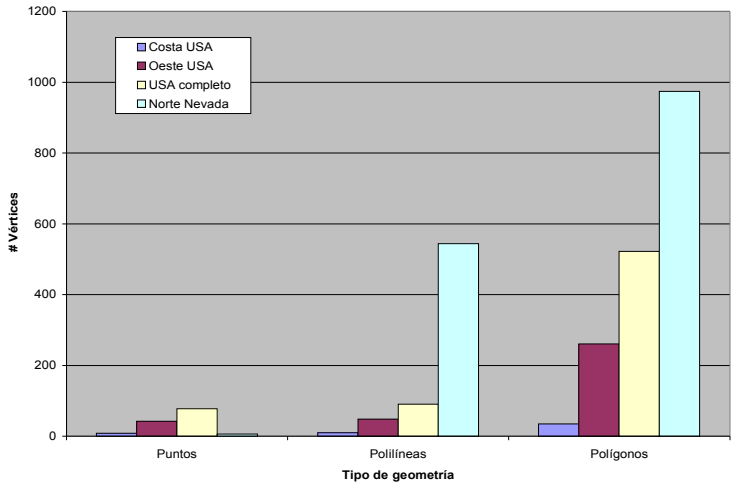


Figura 25: Total de vértices por cada tipo, por mapa.

Este resultado muestra una situación totalmente diferente a la Figura 24: la cantidad de vértices en el mapa de Nevada para polilíneas y polígonos supera en más del doble a las del mapa "USA completo". Para los tres mapas "USA" la relación sigue siendo similar a la obtenida al comparar cantidad de geometrías. Como la cantidad de carrier lines generadas a partir de polígonos y polilíneas es mucho mayor a las generadas por puntos, y con una mayor cantidad de carrier lines se generarán más subgeometrías, el orden según el nivel de complejidad de los mapas dentro del ambiente Piet sería el siguiente, de menor a mayor: "Costa USA", "Oeste USA", "USA completo" y finalmente "Norte Nevada".

La complejidad del mapa de Nevada puede ser excesiva, ya que contiene un nivel de detalle muy alto. Se supone que los mapas reales utilizados en el ambiente Piet deberían tener menor nivel de detalle que este mapa real, con lo que la experimentación sobre este mapa correspondería a lo que podríamos denominar el "peor caso".

6.3 Resultados de experimentación

Agrupamos los resultados de nuestros experimentos según el aspecto de la solución a verificar: (a) proceso de precálculo de información; (b) evaluación de consultas geométricas; (c) evaluación de consultas de agregación geométrica; (d) evaluación de consultas GIS-OLAP. A continuación se muestran los resultados y su análisis.

6.3.1 Resultados del precálculo de información

Para el proceso de precálculo de información se verificó el espacio ocupado por la información generada y los tiempos de ejecución para los diferentes mapas, teniendo en cuenta la cantidad de combinaciones de layer creadas durante el proceso.

Combinaciones para precálculo
La cantidad de combinaciones generadas durante el proceso de precálculo depende directamente de la cantidad de layers tenidos en cuenta para generar la subpoligonización.
Para el procesamiento de mapas USA, que contienen cinco layers utilizables (los layers de países y bounding box no se incluyen dentro del precálculo), se generan 27 combinaciones de layers. En el caso del mapa de Nevada, que incluye cuatro layers, se generan 11 combinaciones.

Espacio ocupado por precálculo
A partir de la experimentación con el proceso de precálculo de información, en la Figura 26 se compara el espacio en Megabytes utilizado en la base de datos por cada uno de los mapas para almacenar la información de componentes geométricos y medidas originales que corresponden a cada mapa previa realización de cualquier procesamiento (color azul), la información auxiliar (temporal) generada durante el proceso de precálculo (bordó) y la información precalculada que se incluye en forma definitiva en el repositorio de datos Piet.

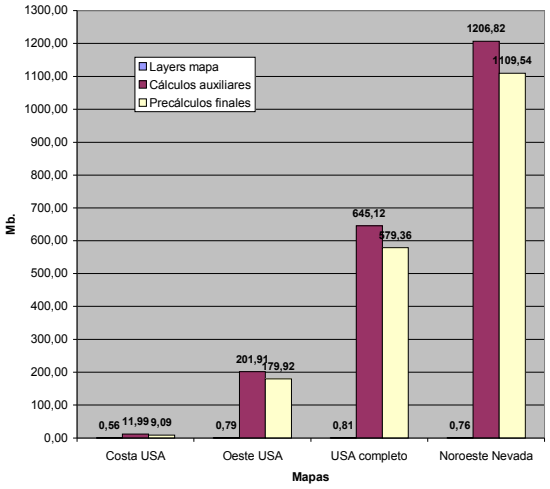


Figura 26: Espacio en repositorio.

En el resultado anterior se puede ver que para los ambientes Piet de cada mapa el espacio ocupado por el mapa original es mínimo en relación a la información auxiliar y de precálculo. Además, podemos ver que el espacio ocupado por los cálculos auxiliares es mayor que el de precálculo final, y esta diferencia crece a medida que aumenta la complejidad del mapa. Esto parece lógico ya que a mayor complejidad del mapa, se crea una mayor cantidad de subgeometrías.

En la Figura 27 se muestra, para el mapa del Noroeste de Nevada, el porcentaje de espacio ocupado y el espacio en Megabytes según el "tipo de información/combinación de layers" (mapa original, datos auxiliares y datos de precálculo).

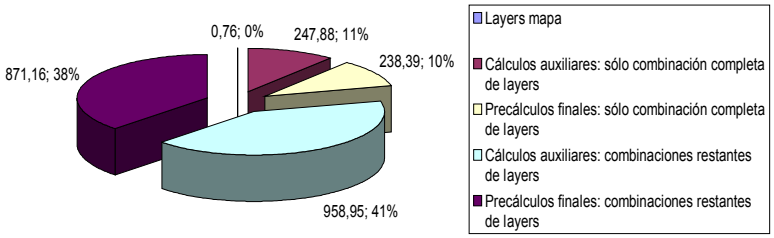


Figura 27: Porcentajes de tipo de información "Noroeste Nevada".

Teniendo en cuenta la cantidad de combinaciones para el mapa de Nevada, en la Figura 27 podemos observar que el porcentaje de espacio ocupado para el almacenamiento final y auxiliar de la combinación completa corresponde a una parte importante del espacio ocupado (21%), aunque sólo sea una combinación. Las restantes 10 combinaciones ocupan sólo un 79%. Esto nos da una idea de que el espacio ocupado por la combinación que utiliza todos los layers es, proporcionalmente, mucho mayor al resto de las combinaciones, llegando incluso a representar un quinto del total del espacio ocupado.

Tiempos para generación de precálculo

El análisis de tiempo de generación de información de precálculo para cada mapa se muestra en la Figura 28, e incluye la comparación de los dos métodos implementados (uso o no de grilla) para el precálculo de todas las combinaciones posibles de layers por cada mapa, incluyendo la cantidad de geometrías y vértices por cada uno. En el mapa "Noroeste Nevada" sólo se puede aplicar el método que utiliza grilla, ya que el otro método (sin grilla) tiene problemas de memoria con mapas grandes, como se describe en la Sección 5.2.2. Los tiempos utilizando el método directo con JTS se muestran en 0 por este motivo.

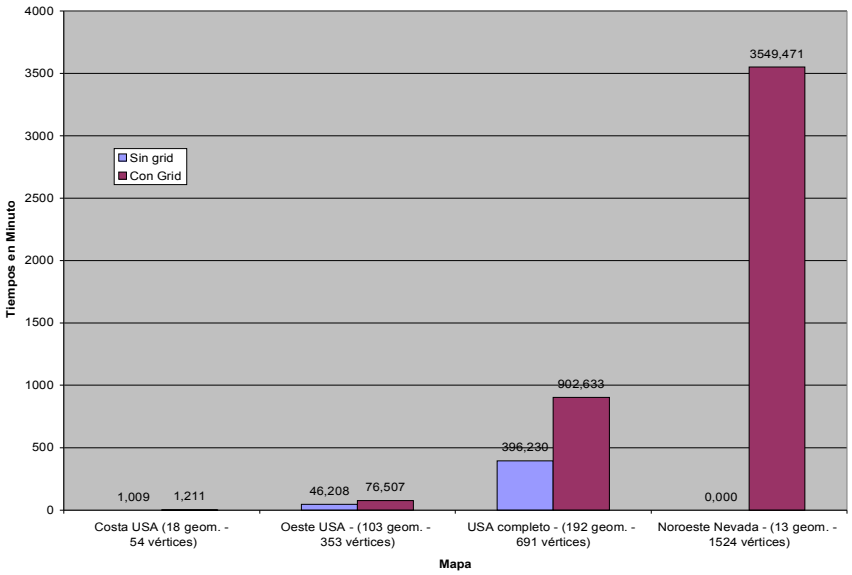


Figura 28: Tiempos de precálculo según método.

Analizando este resultado, corroboramos que la cantidad de geometrías no es un indicativo de la complejidad del mapa, ya que por ejemplo en el caso del mapa "Noroeste Nevada" el tiempo de precálculo es mayor al de los otros mapas, pero sin embargo es el que menor cantidad de geometrías posee (13). El resultado permite ver que la cantidad de vértices sí es un indicador más acorde. Por ejemplo, "Noroeste Nevada" tiene 1524 vértices contra 691 del que le sigue en cantidad de vértices ("USA completo"), y los tiempos de procesamiento son considerablemente mayores para el primer caso en comparación con el segundo.

Observando la ejecución de precálculo con grilla, vemos que el tiempo requerido aumenta considerablemente de acuerdo a la complejidad del mapa (medida por la cantidad de vértices). En esta caso es importante notar que la cantidad de cuadrantes generados es menor en el mapa de Nevada (49 cuadrantes) que en el resto (100, 1936, 6241 para costa USA, oeste USA y USA completo, respectivamente), debido a la cantidad de puntos contenidos en el mapa. Este hecho hace que el rango de búsqueda de asociaciones para cada componente geométrico sea muy grande para este mapa, en comparación con los otros mapas.

Si tomamos en cuenta los tiempos de los dos métodos para los 3 primeros mapas, vemos que el tiempo requerido por el método con grilla crece considerablemente de acuerdo al aumento en la cantidad de vértices del mapa, con la siguiente relación:

- Mapa 54 vértices: 20% más de tiempo utilizando grilla.
- Mapa 353 vértices: 60% más de tiempo utilizando grilla.
- Mapa 691 vértices: 140% más de tiempo utilizando grilla.

Estos datos podrían justificarse por el importante aumento de overhead por parte del algoritmo con grilla, consecuencia de una interacción constante con el repositorio Piet para almacenar y recuperar subgeometrías de acuerdo a los cuadrantes requeridos.

6.3.2 Resultados de consultas geométricas

La experimentación con consultas geométricas se realizó comparando el rendimiento de la ejecución en Piet con consultas semejantes realizadas directamente sobre PostGIS.
La elección de PostGIS [20] como sistema GIS con el cual comparar el rendimiento se debe a que es una herramienta libre, lleva mucho tiempo en el mercado, implementa algoritmos complejos de indexación (R-Trees y GiST[30]) y además tiene buena crítica de especialistas. A pesar de ello, es sabido que no es la más eficiente de las implementaciones de R-Trees, pero es suficientemente representativa de la eficiencia del método como para ser utilizada como comparación.
La utilización de indexación en bases PostGIS es configurable, al igual que en PostgreSQL: los índices pueden crearse y eliminarse para cada tabla en particular. Llamaremos "PostGIS sin indexación" al hecho de no generar ningún tipo de índice sobre las tablas a consultar y "PostGIS con indexación" a los momentos en que se generan índices previa ejecución de la consulta en PostGIS. La creación de índices en PostGIS sólo afecta las consultas realizadas utilizando PostGIS directamente, ya que las consultas Piet se resuelven en general accediendo a información precalculada (sin utilizar los índices sobre geometrías).

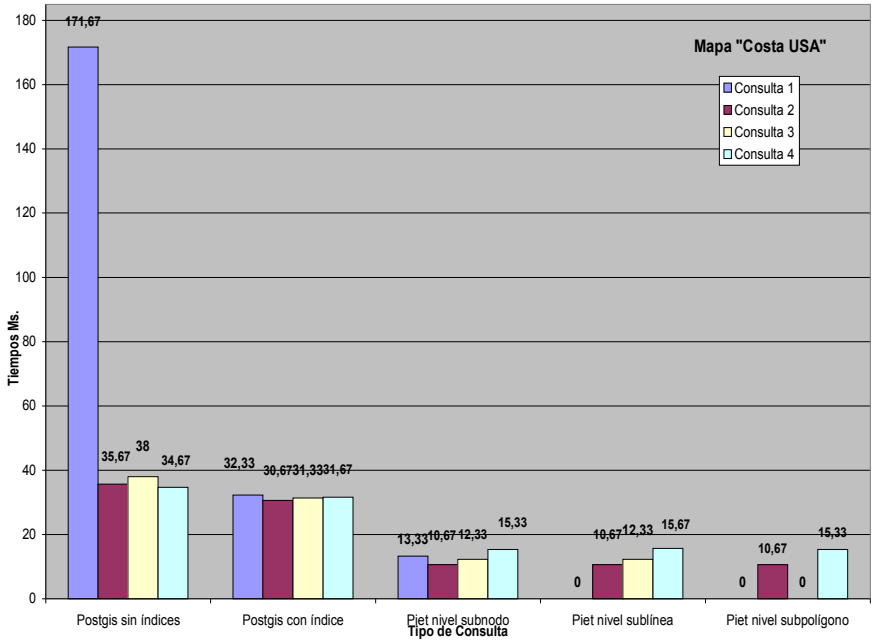
Para la comparación de consultas geométricas definimos cuatro consultas para los mapas "USA" y tres para el mapa "Noroeste Nevada". La ejecución de cada consulta se realizó utilizando PostGIS sin indexación, PostGIS con indexación (GiST), Piet a nivel subnodo[31], Piet a nivel sublínea y Piet a nivel subpolígono (estos últimos en las consultas en que fue posible). El tiempo de ejecución está medido en milisegundos, tomando el promedio de 3 ejecuciones como resultado final.

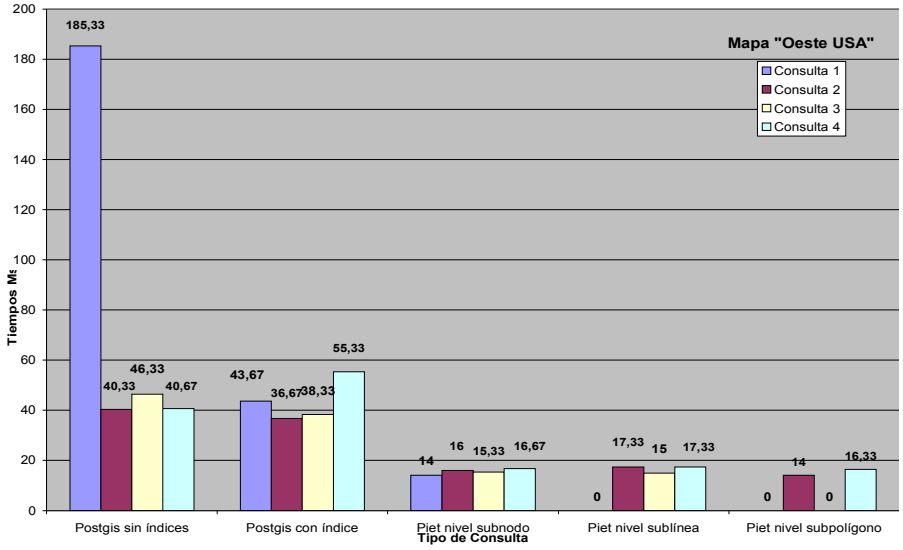
Las consultas ejecutadas en los mapas "Costa USA", "Oeste USA" y "USA completo" fueron las siguientes:
- Consulta 1: "Estados con ciudades que contienen sucursales".
- Consulta 2: "Estados con sus ciudades".
- Consulta 3: "Ciudades y sus sucursales, para ciudades cruzadas por algún río".
- Consulta 4: "Ciudades con aeropuertos y sucursales, dentro del estado identificador 6".

La consulta 1 corresponde a una consulta espacial básica, ya que puede resolverse con el overlay de los 3 layers involucrados. Esta consulta permite el uso de un único nivel de subgeometrías (subnodos) que está dado por el uso de sucursales (representadas con nodos).
La consulta 2 también es una consulta espacial básica, pero a diferencia de la anterior nos permite utilizar en Piet todos los niveles de subgeometrías (subnodos, sublíneas y subpolígonos), ya que involucra 2 layers con geometrías de tipo polígono.
La consulta 3 corresponde a una consulta espacial de mayor complejidad, ya que su resolución obliga a realizar el overlay de 2 layers en forma separada, y luego asociar los resultados.
Finalmente, la consulta 4 selecciona una geometría particular para realizar la consulta.

[30] Los índices GiST (por Generalized Search Trees) dividen la información en "partes a un lado", "partes que se superponen" y "partes que están dentro", y pueden ser usados en un gran rango de tipos de datos. PostGIS usa un índice R-Tree implementado sobre GiST para indexar la información GIS.

[31] Con nivel utilizado en las consultas Piet nos referimos al tipo de subgeometría a utilizar para resolver una operación geométrica, como se describe en la Sección 4.2.2.


Mapa "Costa USA"
Consulta 1
Consulta 2
Consulta 3
Consulta 4
Tiempos Ms.
180
160
140
120
100
80
60
40
20
0
171,67
35,67
38
34,67
32,33
30,67
31,33
31,67
13,33
10,67
12,33
15,33
0
10,67
12,33
15,67
0
10,67
0
15,33
Postgis sin índices
Postgis con índice
Piet nivel subnodo
Piet nivel sublínea
Piet nivel subpolígono
Tipo de Consulta


Mapa "Oeste USA"
Consulta 1
Consulta 2
Consulta 3
Consulta 4
Tiempos Ms.
200
180
160
140
120
100
80
60
40
20
0
185,33
40,33
46,33
40,67
43,67
36,67
38,33
55,33
14
16
15,33
16,67
0
17,33
15
17,33
0
14
0
16,33
Postgis sin índices
Postgis con índice
Piet nivel subnodo
Piet nivel sublínea
Piet nivel subpolígono
Tipo de Consulta

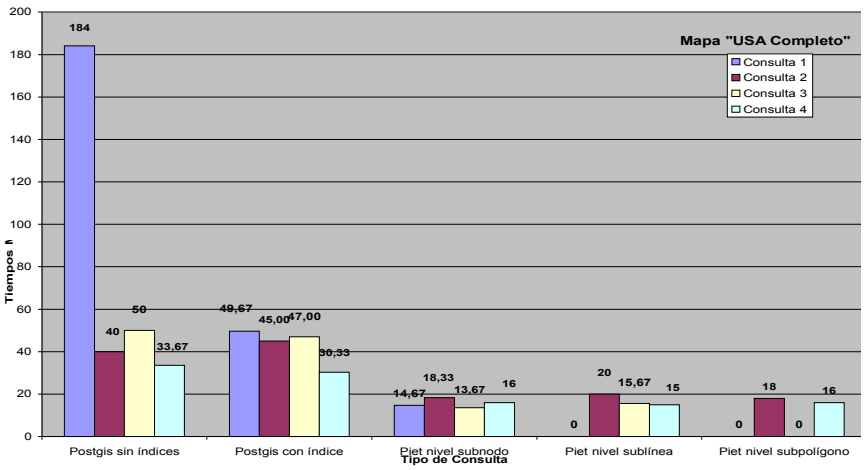


Figura 29: Ejecución consultas geométricas mapas "USA".

Analizando la Figura 29, donde se muestran los resultados obtenidos para los mapas USA para las diferentes consultas, se pueden observar que la consulta 1 en PostGIS sin índices es muy costosa, probablemente debido a que no existe ningún tipo de indexación ni de información precalculada que permita acelerar los cálculos, y esto se ve potenciado en esta consulta. No ocurre lo mismo con el resto de las consultas en los diferentes mapas en PostGIS sin índices.
Para esta consulta (1), la utilización de índices mejora considerablemente los tiempos de ejecución en comparación a la utilización de PostGIS sin índices. En las consultas restantes no se observa una variación importante entre las ejecuciones sobre PostGIS con y sin índices.
Al evaluar la utilización de Piet, en todas las consultas y mapas se observa una mejora importante en la performance entregada por PostGIS, lo que inicialmente demuestra que la utilización de información precalculada para acelerar la resolución de consultas geométricas es una técnica con potencial.
Dentro de la utilización de Piet, en principio no se puede observar un patrón para la utilización de algún tipo de nivel de subgeometría para la resolución de un tipo de consulta particular, aunque a priori se suponía que la utilización de un menor número de subgeometrías en la consulta debería mejorar los tiempos de respuesta (p.ej., utilizando de ser posible el nivel sublínea, la resolución debería ser más rápida que utilizando subnodo, debido a que existen menor cantidad de subgeometrías del primer tipo que del segundo en una tabla de precálculo de overlay). Es probable que esto último no se cumpla porque la diferencia en la cantidad de subgeometrías por tipo es mínima para estos mapas.

Las consultas para el mapa "Noroeste Nevada" fueron:

- Consulta 1: "Condados con aeropuertos y sucursales".
- Consulta 2: "Condados con sucursales, para los condados cruzados por trenes".
- Consulta 3: "Aeropuertos, trenes y sucursales para el condado con identificador 4".
-

Estas 3 consultas se corresponden con las consultas 1, 3 y 4 de realizadas sobre los mapas USA. En la Figura 30 se muestra la comparación de la ejecución de las diferentes consultas sobre el mapa "Noroeste Nevada".

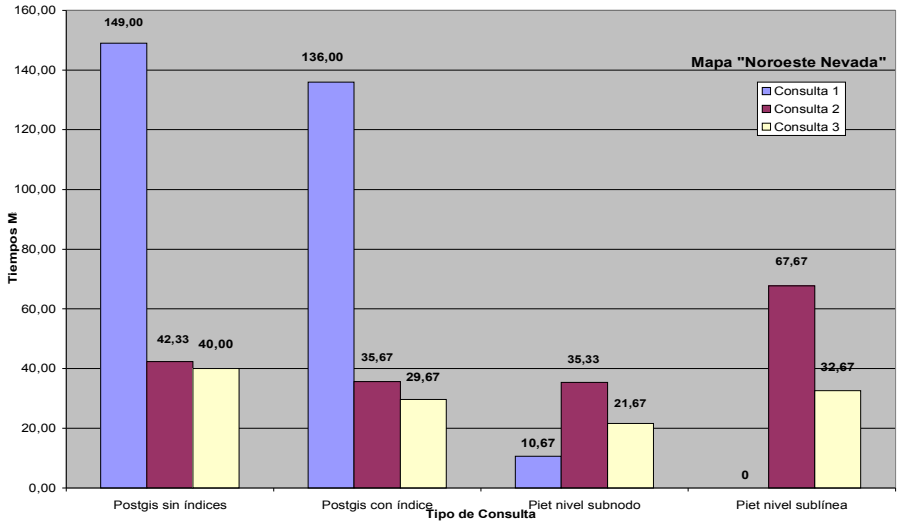


Figura 30: Ejecución consultas geométricas mapa "Noroeste Nevada".

En esta figura se puede observar que, a diferencia de las consultas sobre mapas USA, la resolución de la consulta 1 tarda un tiempo importante utilizando PostGIS con o sin índices (aunque las consultas con índices siguen siendo más eficientes). Esto probablemente se deba a que el motor PostGIS no puede reducir la alta complejidad de las geometrías involucradas en la consulta 1 (los condados están definidos con polígonos complejos), mientras que sí lo logra en las otras dos consultas.
Al utilizar Piet con este mapa a nivel subnodo, la resolución sigue siendo en general más rápida que con PostGIS, para todas las consultas. La ejecución Piet con nivel sublínea arrojó peores resultados que PostGIS con índice, debido probablemente a que se genera una gran cantidad de sublíneas que perjudican la performance de las consultas SQL generadas a partir de Piet.

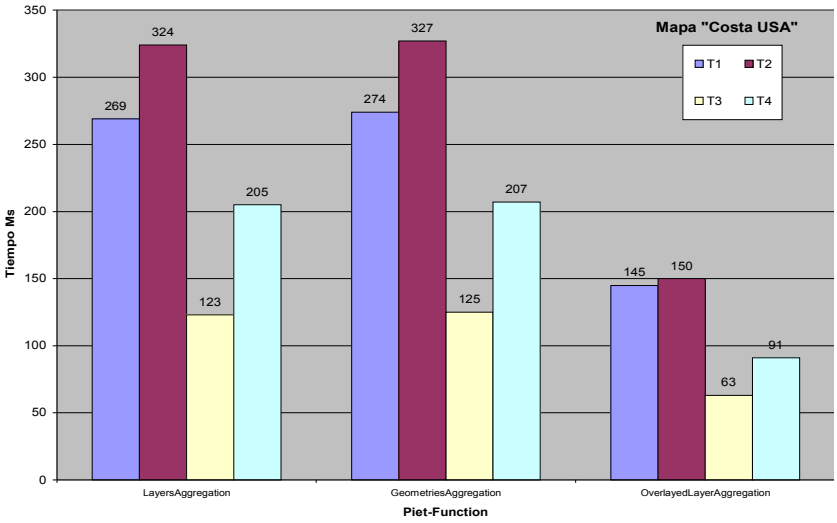
6.3.3 Resultados de consultas de agregación geométrica

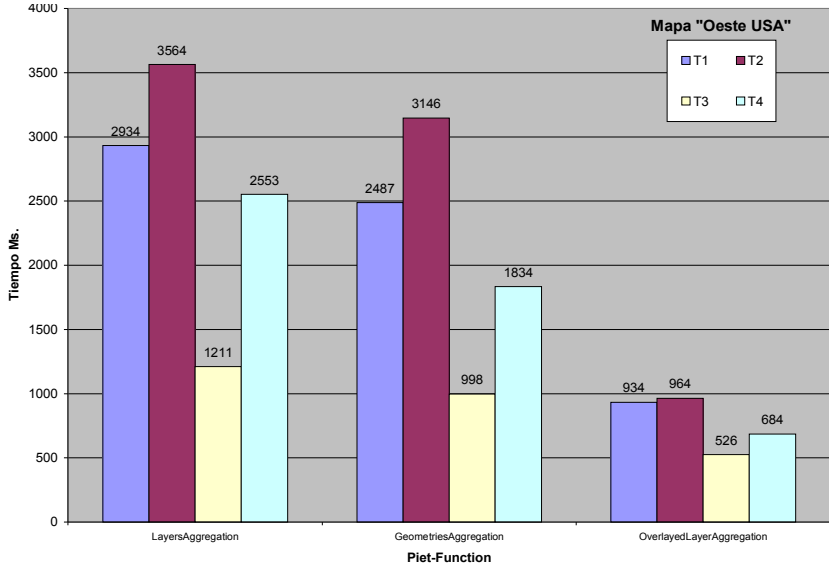
Para verificar el comportamiento de las consultas de agregación geométrica sobre medidas definidas en atributos (hechos) de componentes geométricos[32], se utilizaron los 3 mapas USA y 3 Piet-Functions: "LayersAggregation", "GeometriesAggregation" y "OverlayedLayerAggregation". Los tests ejecutados en esta parte de la experimentación fueron:

- T1: Total de ingresos por estado, para los estados con aeropuertos, ciudades y sucursales.
- T2: Total de población de los estados con sucursales, ciudades, aeropuertos y ríos.
- T3: Cantidad total de vuelos para aeropuertos dentro de estados.
- T4: Área de las ciudades cruzadas por ríos y que contienen sucursales.

En la Figura 31 se incluye la ejecución de las tres Piet-Functions sobre una zona particular seleccionada dentro de cada mapa, para cada uno de los tests mencionados.

[32] Sólo ejecutables desde las interfases Piet-JUMP y Piet-Web


Mapa "Costa USA"
T1
T2
T3
T4
Tiempo Ms
350
300
250
200
150
100
50
0
269
324
123
205
274
327
125
207
145
150
63
91
LayersAggregation
GeometriesAggregation
OverlayedLayerAggregation
Piet-Function


Mapa "Oeste USA"
T1
T2
T3
T4
Tiempo Ms.
4000
3500
3000
2500
2000
1500
1000
500
0
2934
3564
1211
2553
2487
3146
998
1834
934
964
526
684
LayersAggregation
GeometriesAggregation
OverlayedLayerAggregation
Piet-Function

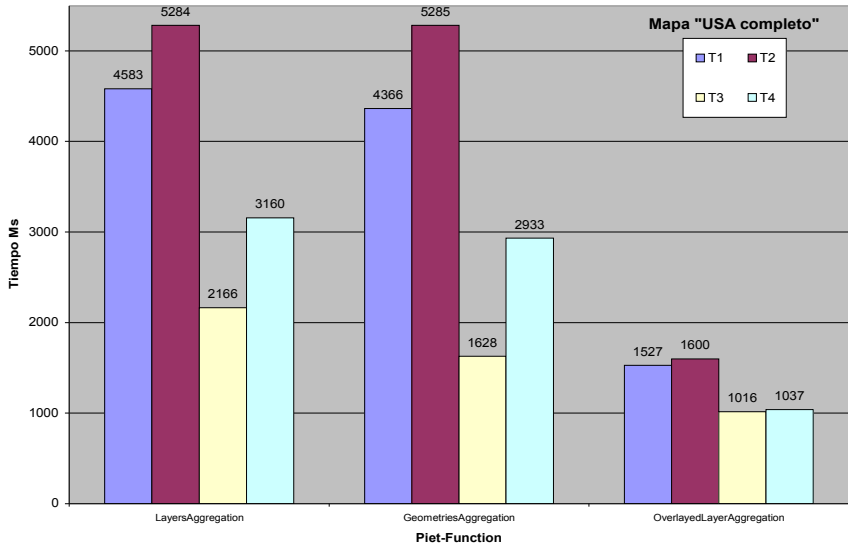
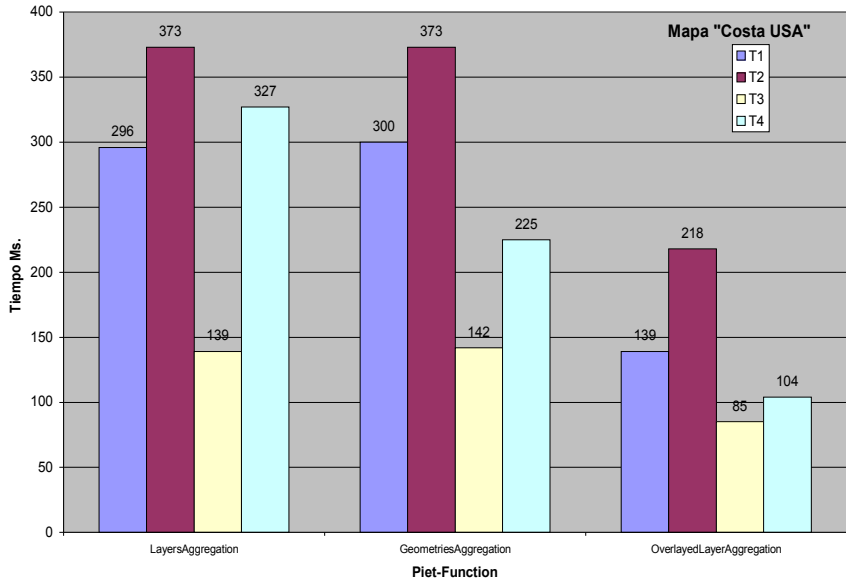


Figura 31: Tiempos Piet-Functions sobre zona seleccionadas para mapas “USA”.

En la Figura 32 se incluyen las mismas consultas pero realizadas sobre los mapas completos.



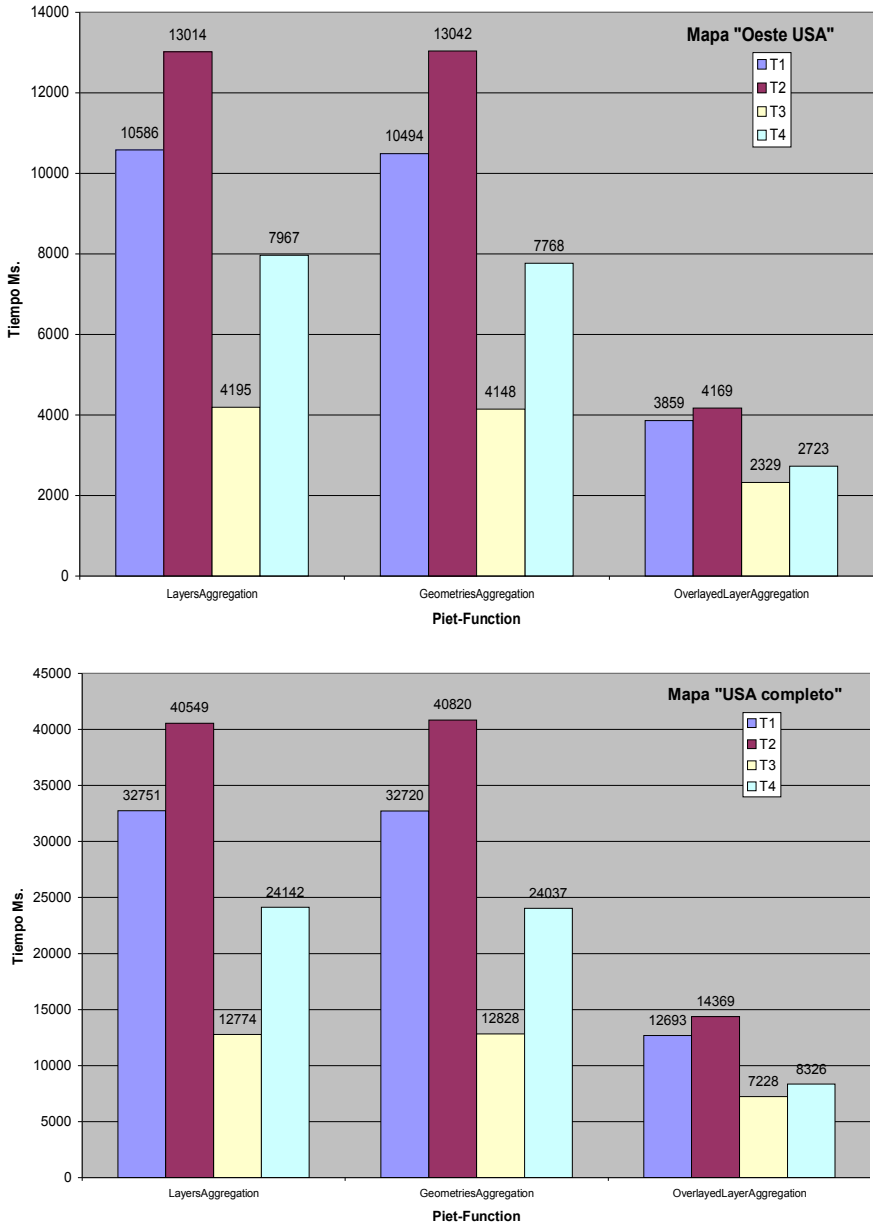


Figura 32: Piet-Functions sobre todo el mapa, para mapas USA.

El análisis de las Figuras 31 y 32 muestra que a mayor complejidad en la cantidad de layers a superponer para resolver la consulta, aumentan los tiempos de resolución. Además, se puede observar que al ejecutar las consultas sobre los mapas completos, los tiempos de respuesta son considerablemente mayores. Se puede observar que los tiempos aumentan considerablemente según aumenta la complejidad del mapa. Sin embargo, los tiempos para resolver cada tipo de consulta (T1 a T4) mantienen la misma relación sin importar el tamaño del área donde se ejecuta (comparación general de resultados de las Figuras 31 y 32).

En los resultados que siguen a continuación (Figura 33) se compara la utilización de las subgeometrías creadas con la subpoligonización de todos los layers de cada mapa USA con la subpoligonización que incluye sólo los layers involucrados en la consulta ("true" indica subpoligonización completa, "false" lo contrario), para la Piet-Function LayersAggregation (sumarización total por layer), para cada consulta, para cada mapa USA. El primer resultado se logró aplicando las consultas sobre un área de selección, mientras que el segundo se produjo con

consultas sobre los mapas completos. Las cuatro consultas se diferencian por color, diferenciando el uso o no de la combinación completa por la tonalidad del color.

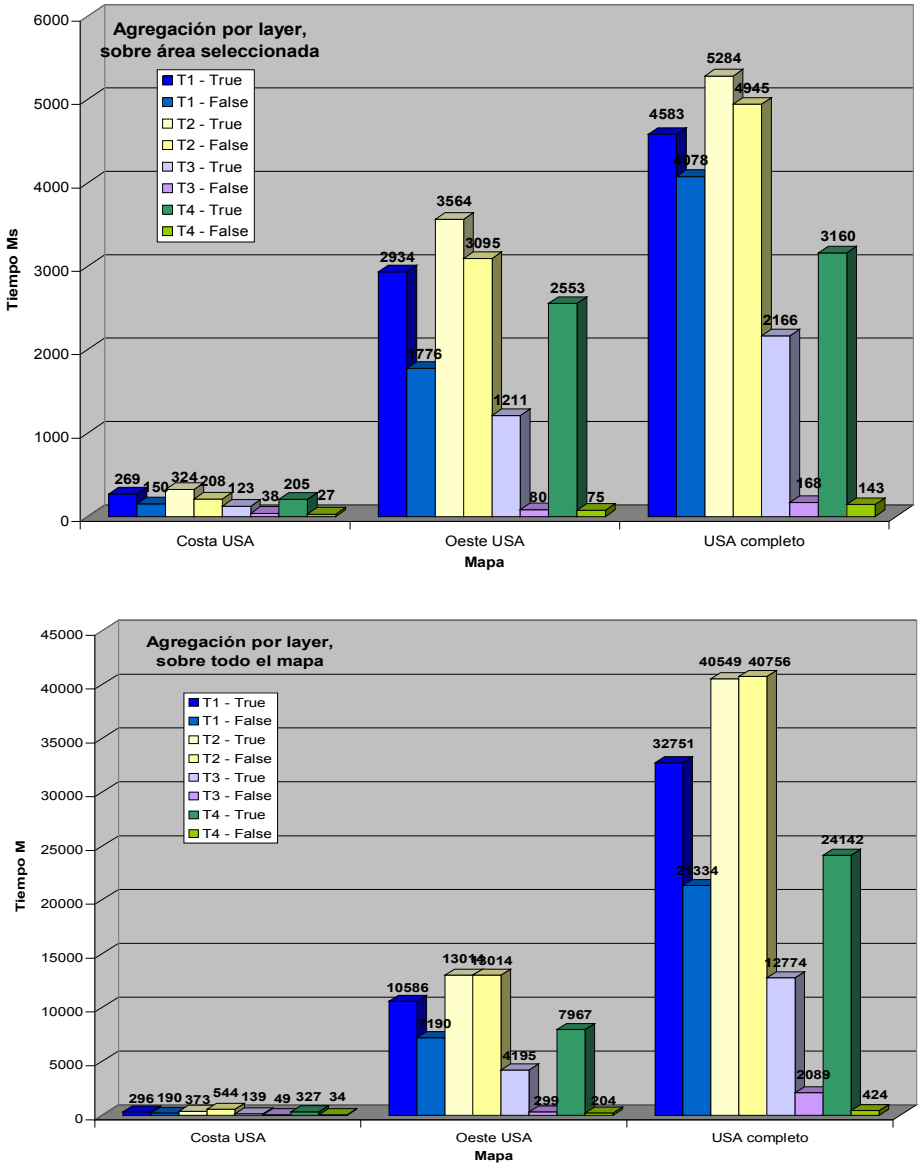


Figura 33: Función de agregación por layer (sumarización) comparando uso de subpoligonización completa en zona seleccionada y mapas completos.

En los resultados de la Figura 33 se puede observar que existe una gran diferencia entre la utilización de la subpoligonización completa de layers y la utilización de la combinación con los layers involucrados en la consulta únicamente llegando, en algunos casos, a tardar 10 veces más. También podemos notar, comparando ambos resultados, que la relación entre la aplicación en una zona seleccionada y en todo el mapa no es constante. Esto se puede deber a que la cantidad de subgeometrías aplicadas a la zona de selección puede involucrar una menor cantidad dependiendo de la consulta en relación a la cantidad de subgeometrías utilizadas para todo el mapa.

Para definir entre el uso de la subpoligonización que utiliza la combinación de todos los layers del mapa o la combinación que incluye sólo los layers involucrados en una consulta, es necesario observar no sólo el tiempo de resolución (Figura 31, 32 y 33), sino también la diferencia en los resultados según la combinación. En la

Figura 34 se incluye, para el mapa oeste USA, la diferencia entre los valores devueltos por las consultas T1 a T4 utilizando la combinación completa de layers o la combinación con los layers involucrados en la consulta únicamente, en una zona de selección y en el mapa completo, para la Piet-Function "LayersAggregation".

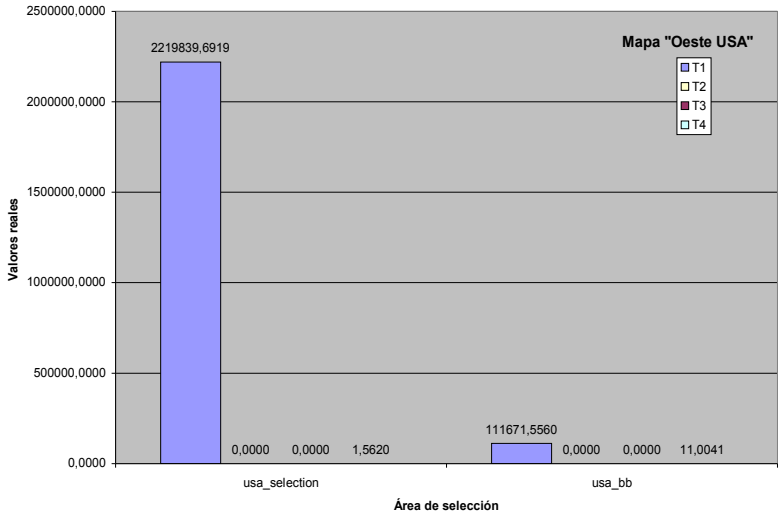


Figura 34: Diferencia "LayersAggregation" según combinación de layers de subpoligonización para mapa oeste USA.

En el resultado de la Figura 34 se puede ver que no existe diferencia en el resultado para las consultas T2 y T3. Esto se debe a que los componentes geométricos involucrados en esas consultas están completamente dentro del área de selección, por lo que las subgeometrías utilizadas para resolver la agregación cubren en forma exacta a los componentes geométricos obteniendo el mismo resultado sin importar la combinación utilizada. Respecto a los resultados para las consultas T1 y T4, estos presentan diferencias tanto con un área de selección como por la selección en el mapa completo. Las diferencias en el mapa completo se deben a errores producidos por las limitaciones en la representación finita, como se desarrolla en la Sección 5.
En el caso de la utilización de un área de selección, se puede observar que la diferencia en el resultado varía según el tipo de combinación de subpoligonización:

- En el caso de utilizar la subpoligonización completa de todos los layers la aproximación es cercana al resultado real, ya que las subgeometrías utilizadas se adaptarán mejor al área seleccionada, al ser más pequeñas.
- Si se utiliza la subpoligonización correspondiente a los layers seleccionados, los valores obtenidos estarán más alejados del resultado real ya que hay un menor número de subgeometrías de mayor tamaño (que cubren los mismos componentes geométricos), que se adaptarán peor a la zona seleccionada.

De los resultados anteriores se desprende que pueden reducirse los errores de aproximación para resolver las consultas a costa de un mayor esfuerzo (mayor tiempo de cálculo): utilizando la subpoligonización de todos los layers combinados implica utilizar la mayor cantidad de subgeometrías (pequeñas) posibles consiguiendo una mejor aproximación, pero a la vez esto hace que el proceso de cálculo requiera más tiempo, ya que el volumen de datos utilizados para obtener el resultado final aumenta. Por otro lado, en los casos en que los componentes geométricos están dentro de la zona de selección, sería conveniente utilizar la combinación exclusiva de los layers involucrados ya que tiene una mejor performance de tiempo y el resultado probablemente sea exacto. En el caso de requerirse un resultado relativamente cerca del real y no poder asegurar que los componentes queden completamente dentro del área de selección, es conveniente utilizar las subpoligonización completa de layers para resolver la agregación, ya que garantiza este tipo de resultados.

6.3.4 Resultados de consultas GIS-OLAP y OLAP

La experimentación con una consulta GIS-OLAP se realizó utilizando el mapa "costa oeste USA", con la siguiente consulta expresada en GISOLAP-QL:

```
SELECT layer.usa_states;FROM PietSchema;WHERE intersection
(layer.usa_states,layer.usa_stores,subplevel.point);
                |
select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]}
ON columns,{([Promotion Media].[All Media], [Product].[All Products])} ON rows from
[Sales] where [Time].[1997]
```

Esta consulta tiene como objetivo obtener la "cantidad de unidades vendidas, costos y total de venta para los diferentes productos y medios de promoción, para las diferentes ciudades dentro de estados definidos en el mapa".
El resultado de la consulta se muestra en la Figura 35.

			Measures		
Store	Promotion Media	Product	Unit Sales	Store Cost	Store Sales
+Los Angeles	+All Media	+All Products			
+Salem	+All Media	+All Products			
-Seattle	+All Media	+All Products	46,996	40,037.98	100,295.52
HQ	+All Media	+All Products			
Store 6	-All Media	+All Products	21,333	18,266.44	45,750.24
	Bulk Mail	-All Products	1,512	1,323.63	3,295.13
		+Drink	139	126.06	297.48
		+Food	1,105	963.90	2,410.03
		+Non-Consumable	268	233.67	587.62
	Cash Register Handout	+All Products	514	452.56	1,132.35
	Daily Paper	+All Products	1,279	1,098.90	2,735.05
	Daily Paper, Radio	+All Products	511	429.27	1,103.81
	Daily Paper, Radio, TV	+All Products	899	760.03	1,906.56
	In-Store Coupon	+All Products			
	No Media	+All Products	14,051	12,035.61	30,154.93
	Product Attachment	+All Products	1,013	866.55	2,184.40
	Radio	+All Products			
	Street Handout	+All Products	258	211.20	528.76
	Sunday Paper	+All Products	853	721.97	1,785.55
	Sunday Paper, Radio	+All Products			
	Sunday Paper, Radio, TV	+All Products			
	TV	+All Products	443	366.73	923.70
Store 7	+All Media	+All Products	25,663	21,771.54	54,545.28

Figura 35: Resultado consulta GIS-OLAP.

La consulta anterior se ejecutó tres veces en forma consecutiva, con los siguientes tiempos medidos en milisegundos.

Número de ejecución	Tiempo de armado de MDX (Piet)	Tiempo de ejecución de MDX (Mondrian)	Tiempo total
1	4166	831	4997
2	1335	170	1505
3	1090	81	1171

Los tiempos requeridos para la resolución de las consultas se componen de los tiempos requeridos por Piet para generar la consulta MDX y los tiempos que tarda Mondrian en ejecutar estas sentencias.
El promedio de tiempo de las tres ejecuciones fue de 2557 milisegundos, de los cuales 2197 fueron utilizados para el armado de la consulta MDX final por parte de Piet, mientras que los 360 restantes fueron utilizados por Mondrian para generar y desplegar el cubo. Es interesante notar como los tiempos requeridos por Mondrian mejoran notablemente con las sucesivas ejecuciones. Esto se puede deber a que Mondrian está cacheando las consultas realizadas por los usuarios para responder con velocidad a una consulta repetida.

Finalmente, para verificar que Piet soporta la ejecución de sentencias OLAP puras se ejecutó la siguiente sentencia MDX:

```
MDX select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store
Sales]} ON columns, Crossjoin({[Store].[All Stores].[USA]}, {([Promotion Media].[All
Media], [Product].[All Products])}) ON rows
from [Sales]
where [Time].[1997]
```

Esta sentencia es semejante a la anterior, con la diferencia de que utiliza únicamente la parte de aplicación (sin integrarla con GIS), agregando la dimensión "Store" de OLAP a nivel "Country".
El resultado obtenido al ejecutar la consulta se muestra en la Figura 36.

			Measures		
Store	Promotion Media	Product	Unit Sales	Store Cost	Store Sales
-USA	+All Media	+All Products	266,773	225,627.23	565,238.13
+CA	+All Media	+All Products			
+ID	+All Media	+All Products	25,635	21,713.53	54,431.14
+KA	+All Media	+All Products			
+LO	+All Media	+All Products			
+MS	+All Media	+All Products			
+MT	+All Media	+All Products			
+NE	+All Media	+All Products	67,659	56,772.50	142,277.07
+NV	+All Media	+All Products	48,951	41,553.43	103,879.16
+OR	+All Media	+All Products			
+TX	+All Media	+All Products			
+UT	+All Media	+All Products	2,117	1,778.92	4,441.18
-WA	+All Media	+All Products	46,996	40,037.98	100,295.52
+Seattle	+All Media	+All Products	46,996	40,037.98	100,295.52
+WY	+All Media	+All Products	75,415	63,770.87	159,914.06

Figura 36: Resultado consultas OLAP en Piet.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

El modelo implementado, junto con la utilización de la subpoligonización y el precálculo del overlay permite expresar y resolver, exitosamente y en tiempos aceptables, consultas de agregación geométrica y consultas GIS-OLAP en forma flexible, manteniendo las principales prestaciones de los sistemas GIS y OLAP actuales (ejecución de consultas GIS y consultas OLAP). La experimentación realizada demostró que el precálculo del overlay es una alternativa competitiva contra otros métodos de indexación, para resolver consultas geométricas en forma rápida.

El cálculo de medidas sobre hechos (o funciones) asignados a componentes geométricos no puede realizarse directamente utilizando librerías geométricas (JTS o PostGIS, por ejemplo), por lo cual sería necesario programar una lógica adicional para resolver los valores de las medidas en ciertos sectores del mapa. Por el contrario, una de las características de Piet es que, a través de la implementación del modelo OLAP espacial, permite expresar consultas de agregación sobre atributos de la parte geométrica sin necesidad de programar funciones adicionales. Además, la utilización de la subpoligonización permite reducir el esfuerzo para calcular la medida usada en una consulta de agregación geométrica de tipo sumable (las soportadas por Piet), brindando tiempos de respuesta competitivos.

Del análisis de la solución propuesta en este documento se desprende que la utilización de la subpoligonización resulta ser una herramienta útil para soportar consultas de agregación geométrica. Adicionalmente, nuestra implementación permite optimizar el precálculo del overlay de los componentes geométricos.

Piet incluye una amplia variedad de configuraciones (p.ej., posibilidad de selección de la combinación de layers de subpoligonización, variantes en los métodos de precálculo, etc.) que pueden mejorar los tiempos de respuesta y la exactitud de los resultados obtenidos.

La solución implementada cumple con el objetivo inicial de resolver consultas básicas de todos los tipos planteados utilizando el modelo OLAP espacial y los conceptos de subpoligonización y precálculo de overlay. Sin embargo, posee ciertas limitaciones que se detallan a continuación (nótese que estas limitaciones no son inherentes al modelo utilizado, sino que son consecuencia de un acotamiento del alcance en esta implementación en particular):

- Las consultas GIS-OLAP soportadas sólo pueden utilizar medidas definidas en la parte de aplicación (no se pueden utilizar medidas sobre hechos definidos en la parte geométrica o hechos implícitos de la dimensión GIS).
- No permite resolver consultas GIS-OLAP integrando medidas sobre hechos de la parte geométrica con medidas asociadas a hechos de la parte de aplicación, ni tampoco la integración entre medidas geométricas y medidas de la parte de aplicación.
- Los tipos de funciones para definir medidas geométricas son limitados.
- Sólo soporta funciones constantes (hechos) asociadas a atributos.

Para ejemplificar las limitaciones de la actual implementación, supongamos el siguiente escenario Piet:

- Existe un mapa con layers "sucursal", "ciudad" y "estado", y una dimensión GIS "sucursal" definida como All->estado->ciudad->sucursal.
- Existe una tabla de hechos, en la parte de aplicación, con el detalle de ventas por sucursal.
- Existen asociaciones definidas entre las sucursales de la parte de aplicación y los correspondientes componentes geométricos.
- El hecho "población" está definido en cada componente geométrico "ciudad" y "estado".
- Existen las medidas "Total de población", "Cantidad de sucursales" y "Total de ventas", definidas como medidas sobre atributos de componentes geométricos, sobre componentes geométricos y sobre la parte de aplicación, respectivamente.

Bajo el escenario anteriormente descrito, las limitaciones de esta implementación de Piet impiden realizar las siguientes consultas:

- "total de población y total de ventas por ciudad", utilizando la dimensión "sucursal" y las medidas "Total de población" y "Total de ventas".
- "total de sucursales y total de ventas por estado", utilizando la dimensión "sucursal" y las medidas "Cantidad de sucursales" y "Total de ventas".

A pesar de estas limitaciones, la solución implementada cumple ampliamente el objetivo inicial de resolver consultas básicas de agregación geométrica, GIS, OLAP y de asociación GIS-OLAP, utilizando el modelo OLAP espacial y los conceptos de subpoligonización y precálculo de overlay. . En la siguiente sección se mencionan las posibles extensiones que pueden realizarse para mejorar la flexibilidad de Piet.

7.2 Trabajo futuro

A partir de la experimentación surgen una serie de extensiones y mejoras importantes que podrían aumentar las prestaciones y la flexibilidad de Piet. A continuación se listan las más importantes:

- Mejorar el proceso de precálculo de información: generar sólo las combinaciones de layers que tienen sentido (p.ej., dado un mapa con un conjunto de layers que incluye un layer de ríos y otro de sucursales, actualmente se está calculando entre todas las combinaciones, la subpoligonización de la combinación exclusiva de ríos y sucursales, lo cual carece de utilidad ya que muy probablemente no existan sucursales que se superpongan con un río, y un usuario nunca consultaría algo así) y además reutilizar de ser posible los cálculos de una combinación de layers para otra combinación que incluya a los mismos layers (p.ej., si se procesó la combinación de ríos con ciudades, reutilizar los cálculos al calcular la subpoligonización de la combinación de ríos, ciudades y estados).

- Agregar al modelo OLAP espacial soporte para un modelo topológico, incorporando a Piet soporte para este tipo de consultas, por ejemplo, total de ventas en las provincias vecinas a Mendoza.

- Modificar el proceso de precálculo para soportar modificaciones en la parte geométrica sin necesidad de recalcular todos los datos (subgeometrías, asociaciones de subgeometrías y valores de hechos propagados, overlay, etc.).

- Automatizar la asociación de componentes geométricos (GIS) con elementos de la parte de aplicación, reduciendo el trabajo manual por parte del administrador Piet.

- Agregar soporte para la aplicación de funciones complejas a utilizar en las consultas de agregación geométrica, complementando los hechos fijos (funciones constantes) soportados en la actualidad. Para esto será necesario, probablemente, utilizar la parte algebraica del modelo, definida en la Sección 2.

- Modificar la sintaxis MDX (gramática) para incorporar consultas Piet-Queries directamente en las sentencias MDX, a diferencia de la versión actual que utiliza un lenguaje externo a MDX (GISOLAP-QL).
  La idea sería aumentar la gramática que define MDX (incluida dentro de Mondrian) agregando las funcionalidades (por medio de nuevas reglas gramaticales) provistas por GISOLAP-QL, modificando también el parser MDX de Mondrian para que resuelva las secciones correspondientes a Piet utilizando el Piet-Engine y el resto (MDX original) utilizando el motor Mondrian como en la actualidad. De esta forma se crearía una gramática GISOLAP-QL integrada (que contendría a MDX) y permitiría expresar las consultas de forma más "elegante" que en la versión actual, además de que facilitaría las futuras extensiones del lenguaje.

- Aumentar las prestaciones de las consultas Piet-Queries:
  - Implementar el soporte de una mayor cantidad de funciones geométricas (que sean resueltas a partir de la subpoligonización y el precálculo del overlay). Deberían implementarse funciones como "touches", "crosses", etc.

    - Implementar nuevas medidas para componentes geométricos para complementar a "count", como "distance", "length" y "area".
    - Extender la expresividad de Piet-Queries para que permitan utilizar todos los tipos de medidas soportados por esta implementación.
    - Implementar nuevas funciones booleanas para conectar operaciones geométricas en una sentencia GISOLAP-QL, como "not", "or" y "xor".
    - Agregar expresividad para escribir consultas con Piet-Functions y consultas geométricas.

- Definir e implementar métodos alternativos para la generación de grillas (utilizadas en algunos métodos de subpoligonización y asociación). Los nuevos métodos deberían definir los cuadrantes de la grilla de acuerdo a la estructura y complejidad del mapa, y no a un dato aislado (en la implementación actual se usan las carrier lines generadas a partir de los nodos, cuya cantidad y distribución es independiente de la complejidad del mapa).

8 Apéndice

8.1 Referencias

[1] Castor software. http://www.castor.org/

[2] Display tags. http://displaytag.sourceforge.net/11/

[3] A. Gutman. R-trees: A dynamic index structure for spatial searching. In Proceedings of SIGMOD'84, páginas 47-57, 1984.

[4] Haesevoets, S. and Kuijpers, B. and Vaisman, A., "Spatial Aggregation: Data Model and Implementation". Submitted for review, 2006.

[5] J. Han, N. Stefanovic, and K. Koperski. Selective materialization: An efficient method for spatial data cube construction. In Proceedings of PAKDD'98, páginas 144-158, 1998.

[6] V. Harinarayan, A. Rajaraman, and J. Ullman. Implementing data cubes efficiently. In Proceedings of the ACM-SIGMOD Conference, páginas 205-216, Montreal, Canada, 1996.

[7] C. Hurtado, A.O. Mendelzon, and A. Vaisman. Updating OLAP dimensions. In Proceedings of ACM DOLAP'99, páginas 60-66, Kansas City, USA, 1999.

[8] Java driver para PostGIS (Posgisdriver). http://postgis.refractions.net/documentation/

[9] Java driver para PostgreSQL (Postgresql.jdbc3). http://jdbc.postgresql.org/

[10] JPivot. http://jpivot.sourceforge.net/

[11] G. Kuper and M. Scholl. Geographic information systems. In J. Paredaens, G. Kuper, and L. Libkin, editors, Constraint databases, chapter 12, páginas 175 a 198. Springer-Verlag, 2000.

[12] Log4j library. http://logging.apache.org/log4j/docs/

[13] Manual de referencia MDX. http://msdn2.microsoft.com/en-us/library/ms145506.aspx

[14] M. Miquel, Y. B_edard, A. Brisebois, J. Pouliot, P. Marchand, and J. Brodeur. Modeling multidimensional spatio-temporal data warehouses in a context of evolving specifications. In Symposium on Geospatial Theory, Processing and Applications, Ottawa, Canada, 2002.

[15] Mondrian. http://mondrian.sourceforge.net/.

[16] OpenGIS. http://www.opengeospatial.org/standards

[17] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. OLAP operations in spatial data warehouses. In Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases, páginas 443 a 459, 2001.

[18] D. Papadias, Y. Tao, P. Kalnis, and J. Zhang. Indexing spatio-temporal data warehouses. In Proceedings of ICDE'02, páginas 166 a 175, 2002.

[19] T.B Pedersen and N. Tryfona. Pre-aggregation in spatial data warehouses. Proceedings of International Symposium on Advances in Spatial and Temporal Databases, páginas 460-480, 2001.

[20] PostGIS para PostgreSQL. http://postgis.refractions.net/documentation/

[21] PostgreSQL. http://www.postgresql.org/

[22] F. Rao, L. Zang, X. Yu, Y. Li, and Y. Chen. Spatial hierarchy and OLAP-favoured search in spatial data warehouse. In Proceedings of DOLAP'03, páginas 48-55, Louisiana, USA, 2003.

[23] S. Rivest, Y. B_edard, and P. Marchand. Modeling multidimensional spatio-temporal data warehouses in a context of evolving speci_cations. Geomatica, 55 (4), 2001.

[24] Sitio oficial de JUMP project (Java Unified Mapping Platform): http://www.jump-project.org.

[25] WCF (Web Component Framework). http://jpivot.sourceforge.net/wcf/index.html

[26] L. Zang, , Y. Li, F. Rao, , X. Yu, and Y. Chen. An approach to enabling spatial OLAP by aggregating on spatial hierarchy. In Proceedings of DaWak'03, páginas 35-44, Prague, Czech Republic, 2003.

8.2 Glosario

Nodo: punto donde un par de líneas de una misma geometría o de diferentes geometrías se intersecan.

Componentes geométricos: geometrías contenidas en los layers del mapa.

Subgeometrías: polígonos abiertos, polilíneas abiertas y nodos generados durante el proceso de subpoligonización de un mapa.

Subnodos: los nodos dentro del conjunto de subgeometrías de una subpoligonización.

Sublíneas: las líneas dentro del conjunto de subgeometrías de una subpoligonización.

Subpolígonos: los polígonos dentro del conjunto de subgeometrías de una subpoligonización.

Hechos: los hechos en Piet pueden ser definidos en la parte de aplicación (Ej.: ventas de un por producto durante cada día de 1997) o en forma de funciones constantes asignados a atributos de componentes geométricos (Ej.: población en el polígono que define la ciudad "Buenos Aires"). También puede tomarse como hechos la estructura de dimensión GIS (Ej.: la geometría "Rosario" está dentro de la geometría "Stanta Fe"). A estos tres tipos de hechos se les podrá asignar medidas.

Medidas: las medidas en Piet pueden corresponder a medidas definidas en la parte de aplicación (Ej.: promedio de ventas), medidas definidas sobre la parte geométrica (Ej.: cantidad de geometrías) o medidas definidas sobre hechos de la parte geométrica (Ej.: cantidad total de población). Estas 3 clases de medidas reciben el nombre de medidas OLAP, medidas geométricas y medidas sobre atributos geométricos, respectivamente.

Consultas geométricas: consultas que involucran únicamente componentes geométricos. Permiten realizar operaciones geométricas como intersección, superposición, etc.

Consultas OLAP: consultas que involucran únicamente información de la parte de aplicación. Pueden ser expresadas y resueltas con herramientas OLAP.

Consultas GIS-OLAP: consultas que relacionan información de la parte geométrica con información de aplicación.

Consultas de agregación geométrica: consultas que utilizan medidas definidas sobre la parte geométrica, que podrán estar definidas sobre relaciones entre componentes geométricos o sobre hechos asociados a los mismos.

FeatureSchema: objeto que define la estructura de atributos y tipo de datos que pueden contener una clase de Feature. Por ejemplo, un featureSchema definirá la estructura de "Ciudades" e incluirá los atributos "geometría" de tipo "Geometry", "nombre" de tipo "String" y "población" de tipo "double".

Feature: objeto que representa una instancia de un FeatureSchema con valores asignados a los atributos. Con el ejemplo anterior, "Buenos Aires" sería la instancia, con un polígono para representar la geometría, el string "Buenos Aires" y "3000000.0" asignado a cada uno de los atributos definidos en el FeatureSchema.

JTS (Java Topology Suite): es una API open source, desarrollada en Java, de funciones espaciales aplicables a 2 dimensiones.

JUMP (Java Unified Mapping Platform): es una aplicación gráfica open source, desarrollada en Java, que permite visualizar y procesar información espacial, incluyendo las funciones espaciales y GIS más comunes. JUMP está diseñado para funcionar como un framework al que pueden agregarse en forma rápida y fácil, extensiones desarrolladas a medida.

PostGIS: es un plugin que agrega a la base de datos relacional **PostgreSQL** soporte para objetos geográficos, permitiendo utilizar PostgreSQL server como base de datos de backend de un sistema GIS.

Mondrian: es un server OLAP escrito en Java, que permite analizar en forma interactiva grandes conjuntos de datos almacenados en bases de datos relacionales, sin necesidad de escribir sentencias SQL para consultarla.

Open GIS: es una especificación libre del modelo de componentes GIS que posibilita el intercambio de información entre diferentes sistemas GIS.

MDX (Multidimensional Expressions): es un lenguaje de consulta para bases de datos OLAP. Un ejemplo de consulta MDX para obtener la "cantidad de unidades vendidas y total de ventas por sucursal, detallada para los diferentes productos durante el segundo cuatrimestre de 1997" sería:

```
SELECT {[Measures].[Unit Sales], [Measures].[Store Sales]} ON COLUMNS,
  {[Product].members} ON ROWS
FROM [Sales]
WHERE [Time].[1997].[Q2]
```