Piet: a GIS-OLAP Implementation

Leticia I. Gomez
Instituto Tecnologico de Buenos Aires
lgomez@itba.edu.ar

Joint work with Ariel Escribano, Bart Kuijpers and Alejandro A. Vaisman

Outline

- *GIS-OLAP motivation*
- Query Language
- Data Model
- Implementation Details
- Experimental Results
- Conclusion

GIS Systems

1. Organize geometric objects in thematic layers
2. Spatial objects can be annotated with numerical and categorical information
3. Two kinds of queries: pure geometric queries and geometric aggregation queries.
4. Queries use some indexing technique like R-tree or its variation

GIS Systems

1. *Organize geometric objects in thematic layers*
2. Spatial objects can be annotated with numerical and categorical information
3. Two kinds of queries: pure geometric queries and geometric aggregation queries.
4. Queries use some indexing technique like R-tree or its variation

GIS Systems

1. *Organize geometric objects in thematic layers*
2. *Spatial objects can be annotated with numerical and categorical information*
3. Two kinds of queries: pure geometric queries and geometric aggregation queries.
4. Queries use some indexing technique like R-tree or its variation

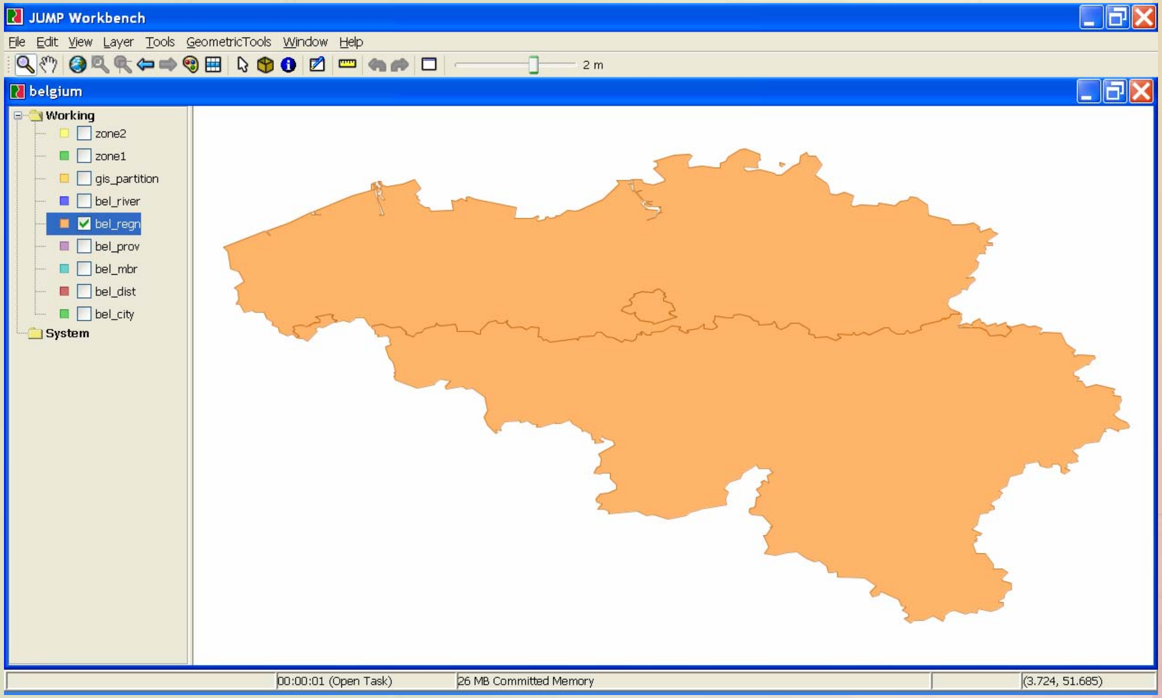
GIS Systems

Example: Map of Belgium organized in 5 layers with demographic and economic information

GIS Systems

Example: Map of Belgium organized in 5 layers with demographic and economic information

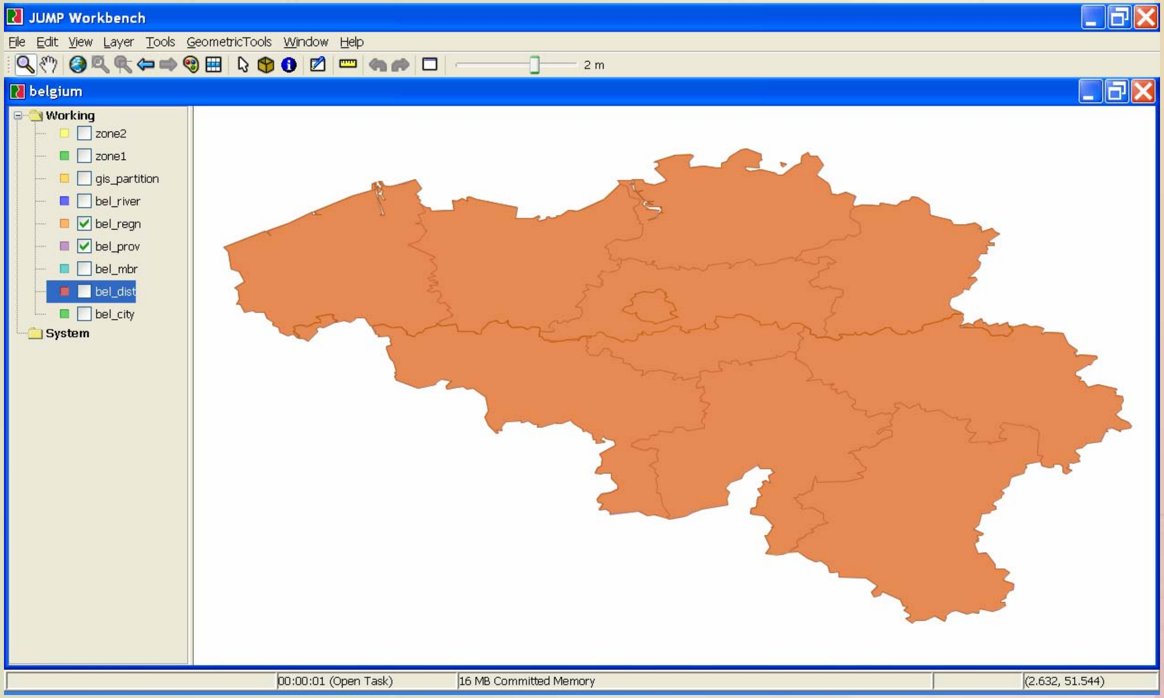
- *regions (polygons)*



GIS Systems

Example: Map of Belgium organized in 5 layers with demographic and economic information

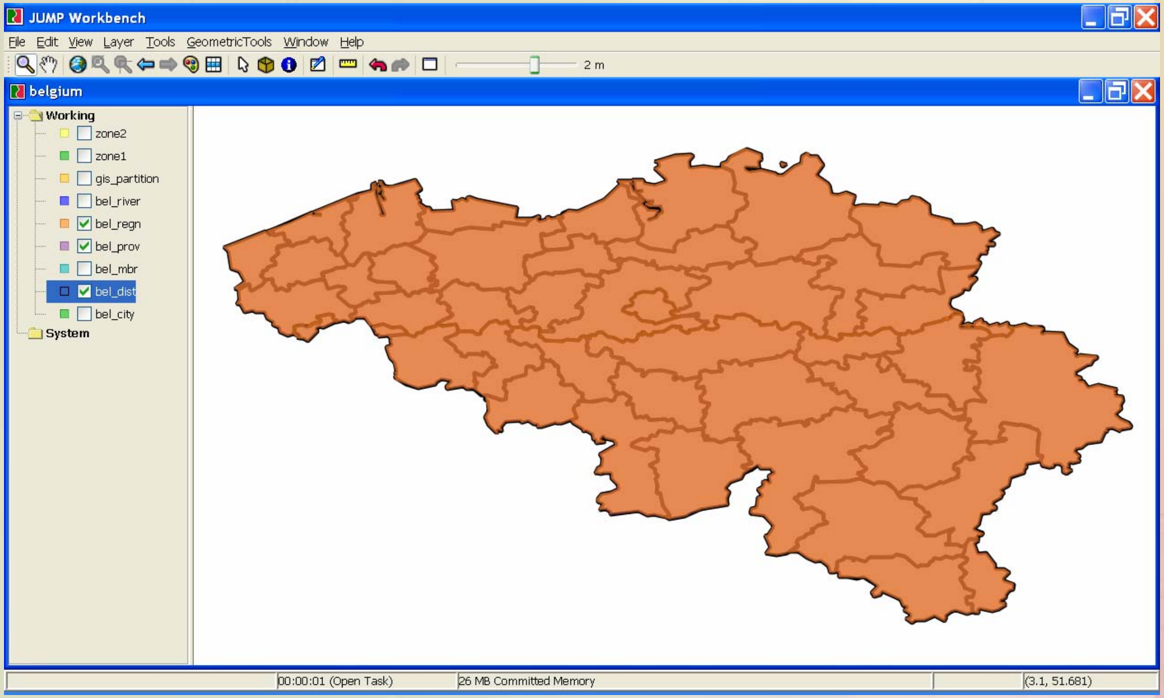
- regions (polygons)
- *provinces (polygons)*



GIS Systems

Example: Map of Belgium organized in 5 layers with demographic and economic information

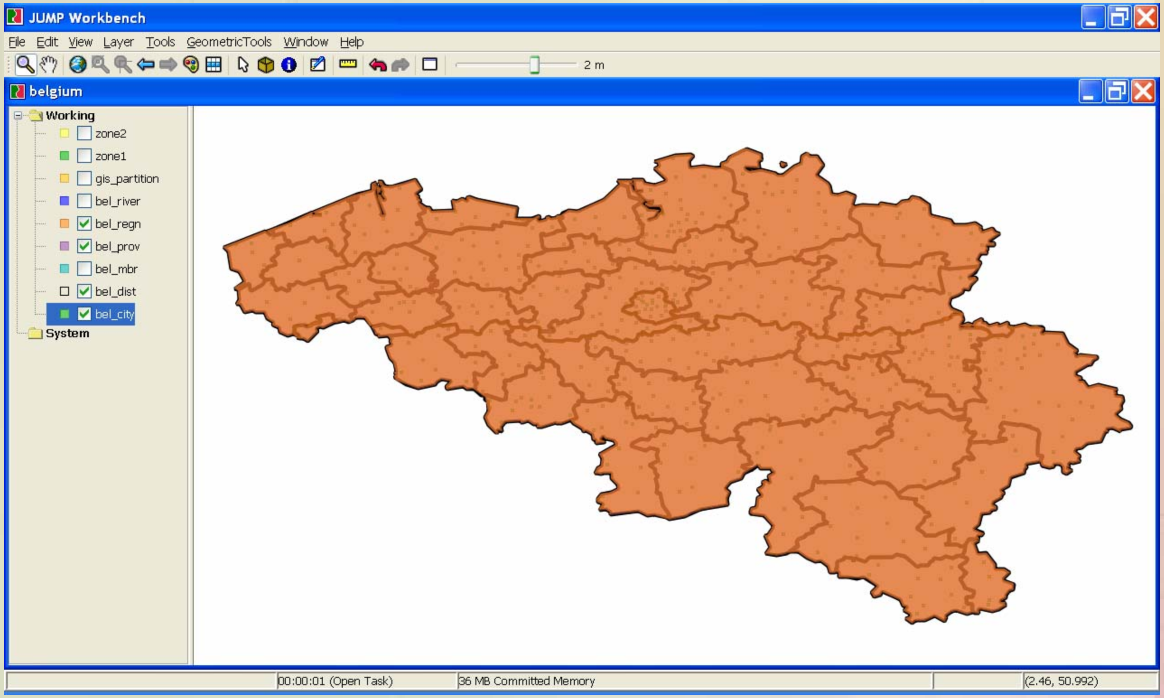
- regions (polygons)
- provinces (polygons)
- *districts (polygons)*



GIS Systems

Example: Map of Belgium organized in 5 layers with demographic and economic information

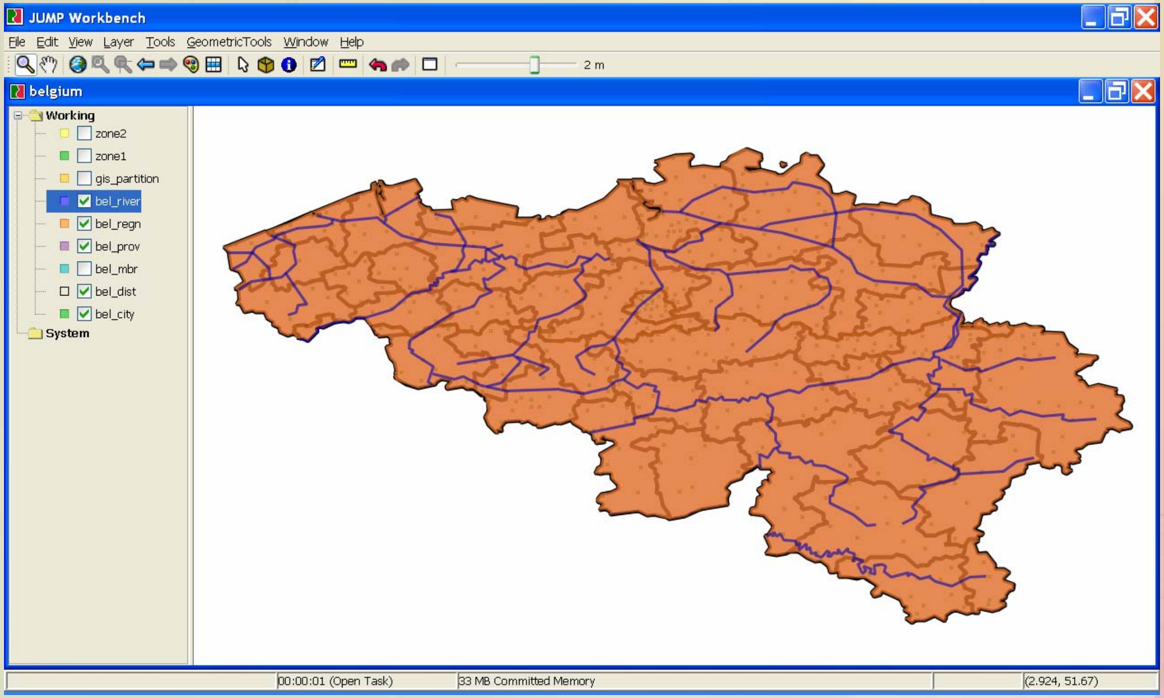
- regions (polygons)
- provinces (polygons)
- districts (polygons)
- *cities (points)*



GIS Systems

Example: Map of Belgium organized in 5 layers with demographic and economic information

- regions (polygons)
- provinces (polygons)
- districts (polygons)
- cities (points)
- *rivers (polylines)*



GIS Systems

1. Organize geometric objects in thematic layers
2. Spatial objects can be annotated with numerical and categorical information
3. Two kinds of queries: pure geometric queries and geometric aggregation queries.
4. Queries use some indexing technique like R-tree or its variation

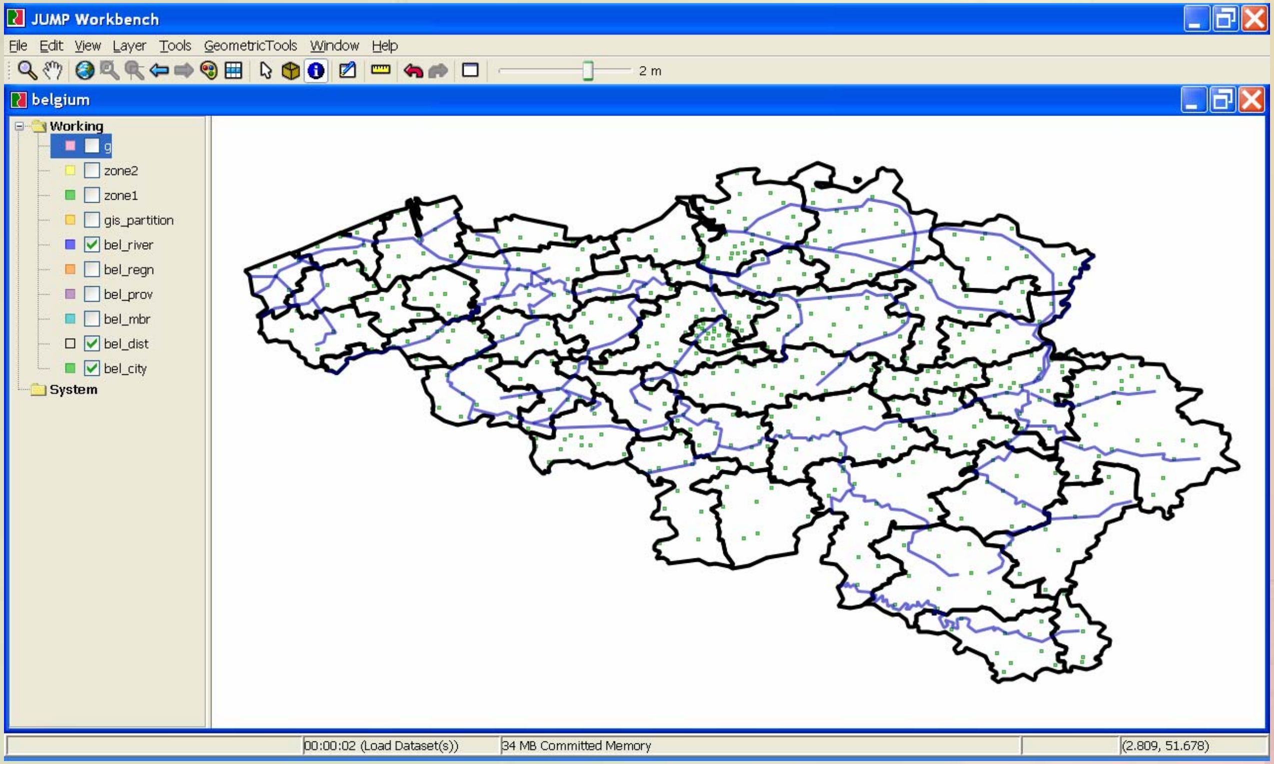
GIS Systems

1. Organize geometric objects in thematic layers
2. Spatial objects can be annotated with numerical and categorical information
3. *Two kinds of queries: pure geometric queries and geometric aggregation queries.*
4. *Queries use some indexing technique like R-tree or its variation*

GIS Systems - Queries

Pure Geometric Queries

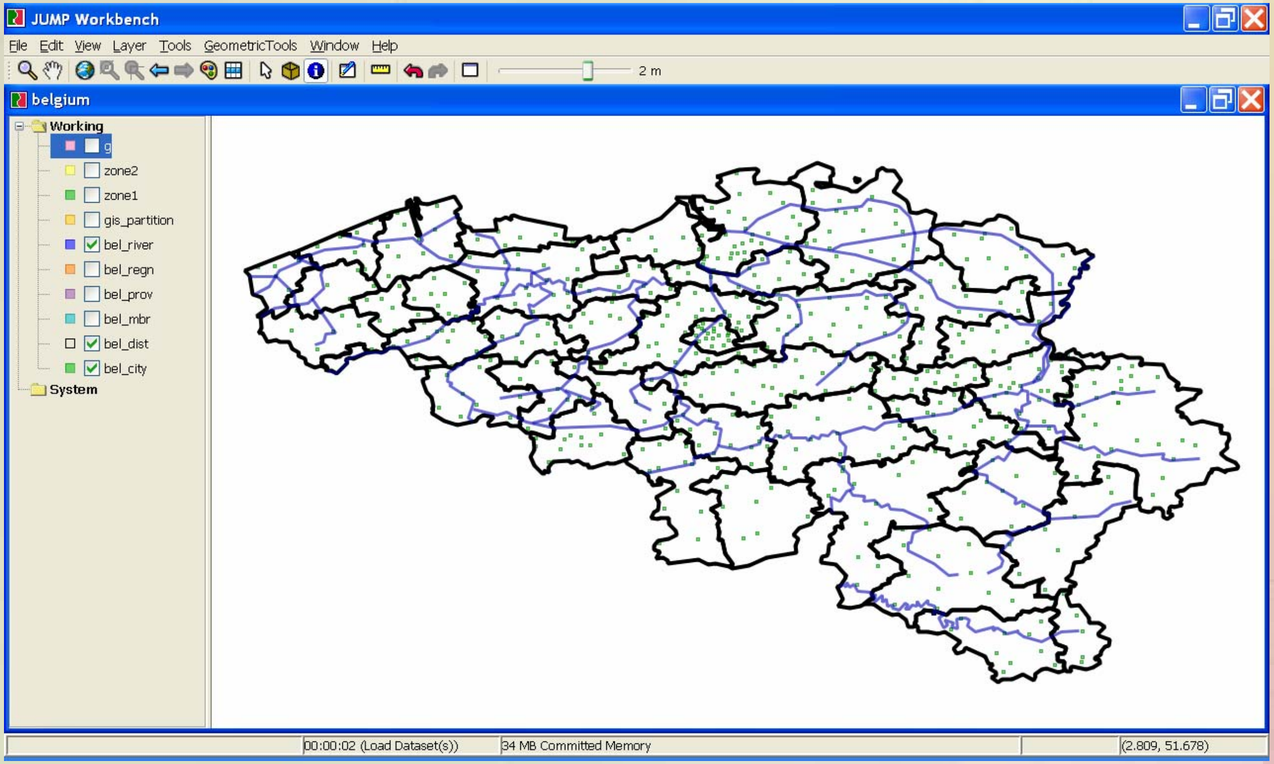
"*Districts* and their *cities*, only for districts crossed by *rivers*"



GIS Systems - Queries

Geometric Aggregation Queries

“For each *district* crossed by at least one river, show the total number of its *cities*”



GIS Systems

1. Organize geometric objects in thematic layers
2. Spatial objects can be annotated with numerical and categorical information
3. Two kinds of queries: pure geometric queries and geometric aggregation queries.
4. *Queries use some indexing technique like R-tree or its variation*

OLAP & Multidimensional Databases

1. We assume that non-spatial information resides in **data warehouses.**
2. Data perceived as a cube, where the dimensions provide contextual information and the cells contain **measures** of facts.
3. OLAP tools used to exploit *multidimensional databases*.

OLAP & Multidimensional Databases

1. *We assume that non-spatial information resides in* ***data warehouses*****.**
2. Data perceived as a cube, where the dimensions provide contextual information and the cells contain **measures** of facts.
3. OLAP tools used to exploit *multidimensional databases*.

OLAP & Multidimensional Databases

1. We assume that non-spatial information resides in **data warehouses.**
2. *Data perceived as a cube, where the dimensions provide contextual information and the cells contain **measures** of facts.*
3. OLAP tools used to exploit *multidimensional databases*.

OLAP & Multidimensional Databases

1. *We assume that non-spatial information resides in* ***data warehouses*****.**
2. Data perceived as a cube, where the dimensions provide contextual information and the cells contain **measures** of facts.
3. *OLAP tools used to exploit multidimensional databases.*

OLAP & Multidimensional Databases

Example: Information about stores and sales in Belgium

OLAP Queries

"*Unit Sales, Store Cost and Store Sales* for *products and promotion media offered by stores* in provinces, during 1997"

Motivation for GIS-OLAP

- Data aggregation marginally present in commercial GIS
- Light integration between spatial and non-spatial data
- A single framework for GIS and OLAP is needed.
- This requires a formal data model and query language

GIS-OLAP: Our Proposal

- Based on a solid formal model (see "*Spatial aggregation: Data model and implementation*" [Gomez,Haesevoets, Kuijpers, and Vaisman]).

- Allows expressing a wide range of aggregation queries over a GIS map and a data warehouse.

- Implements a novel query evaluation technique, called "sub-poligonization" that go beyond the typical R-tree-based strategies

***Classical Solution*:**

For Geometric Queries => database spatial extenders, such as R-trees

For Geometric Aggregation Queries => techniques not implemented in commercial DBMS like aR-trees

Outline

- GIS-OLAP motivation
- *Query Language*
- Data Model
- Implementation Details
- Experimental Results
- Conclusion

Query Language

Pure Geometric Queries *(SQL Approach)*

“*Districts* and their *cities*, only for districts crossed by *rivers*”

Query Language

Pure Geometric Queries *(SQL Approach)*

"*Districts* and their *cities*, only for districts crossed by *rivers*"

```
SELECT layer.bel_dist, layer.bel_city;
FROM PietSchema;
WHERE intersection(layer.bel_river,layer.bel_dist) AND
    contains(layer.bel_dist,layer.bel_cities)
```

Query Language

Geometric Aggregation Queries *(SQL Approach)*

“For each *district* crossed by at least one river, show the total number of its *cities*”

Query Language

Geometric Aggregation Queries *(SQL Approach)*

"For each *district* crossed by at least one river, show the total number of its *cities*"

```
SELECT layer.bel_dist, measure.CitiesQuantity;
FROM PietSchema;
WHERE intersection(layer.bel_river,layer.bel_dist) AND
        contains(layer.bel_dist,layer.bel_city)
```

Query Language

OLAP Queries *(MDX)*

“Browse *Unit Sales, Store Cost and Store Sales* for *products and promotion media offered by stores* in provinces, during 1997”

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]

Query Language

GIS-OLAP Queries *(SQL Approach + MDX)*

"Browse *Unit Sales, Store Cost and Store Sales* for *products and promotion media offered by stores* in provinces

crossed by rivers,

during 1997"

SQL Approach

|

MDX

Query Language

GIS-OLAP Queries *(SQL Approach + MDX)*

```
SELECT layer.bel_prov;
FROM PietSchema;
WHERE intersection(layer.bel_river,layer.bel_prov);
|
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]
```

Query Language

GIS-OLAP Queries *(SQL Approach + MDX)*

```
SELECT layer.bel_prov;
FROM PietSchema;
WHERE intersection(layer.bel_river,layer.bel_prov);
|
SELECT
{[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store
    Sales]} ON columns,
{([Promotion Media].[All Media], [Product].[All F
FROM [Sales]
WHERE [Time].[1997]
```

The first part is a "geometric query". Once solved, the province IDs are passed to the MDX part

Query Language

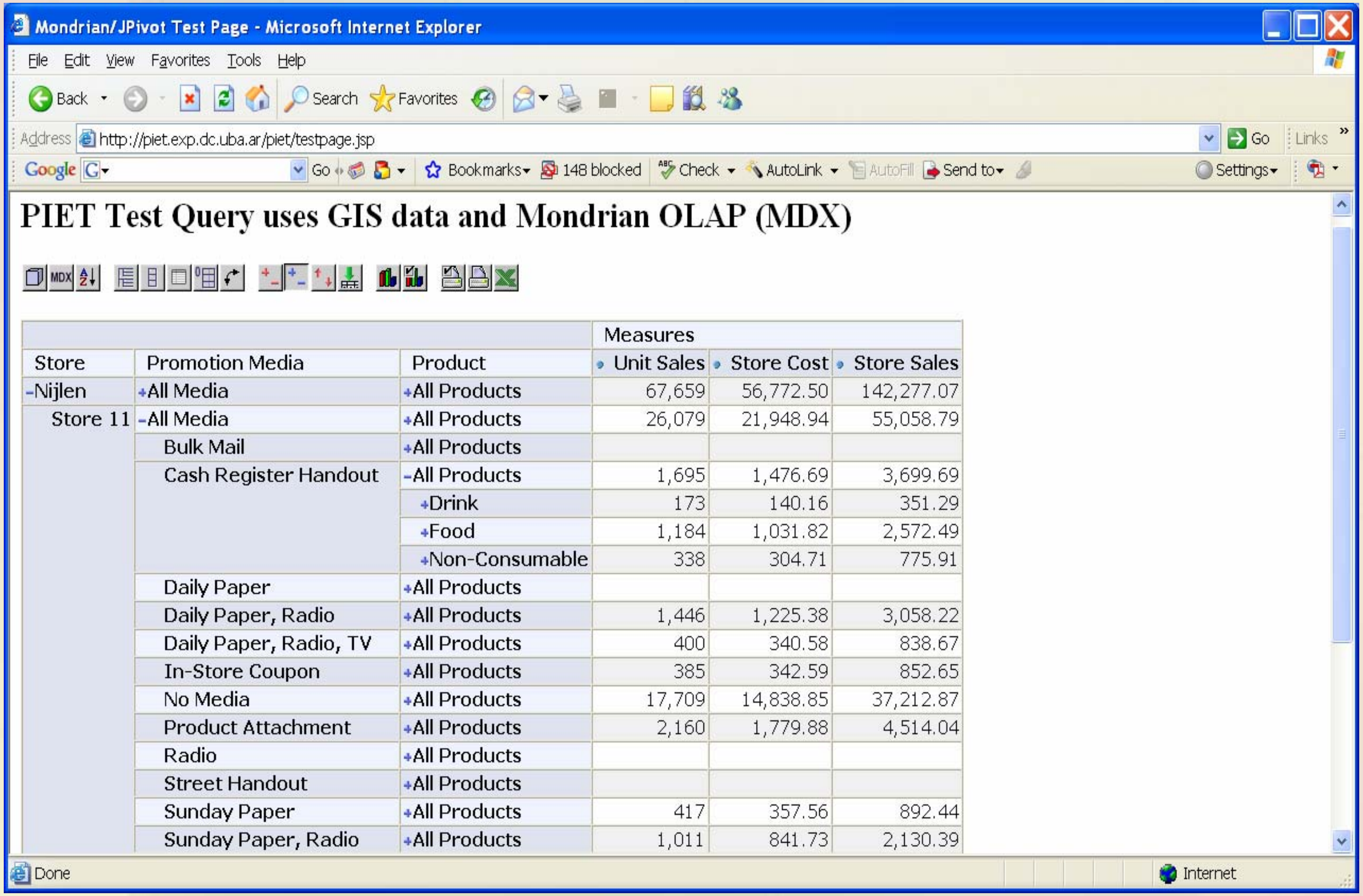
Suppose that these province IDs correspond to "ANT", "LIE" and "LUX" then the MDX is rewritten as:

```
SELECT
{[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store
    Sales]} ON columns,
Crossjoin(Hierarchize(Union(Union(
{[Store].[All Stores].[BEL].[ANT].Children},
{[Store].[All Stores].[BEL].[LIE].Children}),
{[Store].[All Stores].[BEL].[LUX].Children})),
```

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]
```

Query Language



PIET Test Query uses GIS data and Mondrian OLAP (MDX)

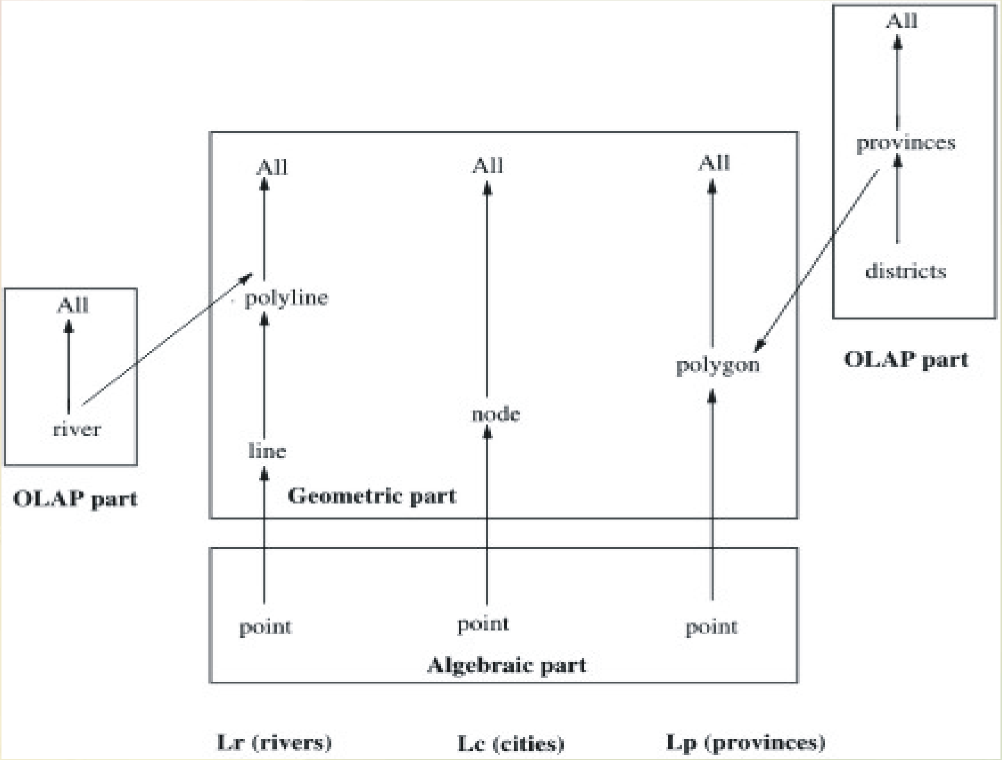
			Measures		
Store	Promotion Media	Product	Unit Sales	Store Cost	Store Sales
-Nijlen	+All Media	+All Products	67,659	56,772.50	142,277.07
Store 11	-All Media	+All Products	26,079	21,948.94	55,058.79
	Bulk Mail	+All Products			
	Cash Register Handout	-All Products	1,695	1,476.69	3,699.69
		+Drink	173	140.16	351.29
		+Food	1,184	1,031.82	2,572.49
		+Non-Consumable	338	304.71	775.91
	Daily Paper	+All Products			
	Daily Paper, Radio	+All Products	1,446	1,225.38	3,058.22
	Daily Paper, Radio, TV	+All Products	400	340.58	838.67
	In-Store Coupon	+All Products	385	342.59	852.65
	No Media	+All Products	17,709	14,838.85	37,212.87
	Product Attachment	+All Products	2,160	1,779.88	4,514.04
	Radio	+All Products			
	Street Handout	+All Products			
	Sunday Paper	+All Products	417	357.56	892.44
	Sunday Paper, Radio	+All Products	1,011	841.73	2,130.39

Outline

- GIS-OLAP motivation
- Query Language
- *Data Model*
- Implementation Details
- Experimental Results
- Conclusion

Data Model Overview

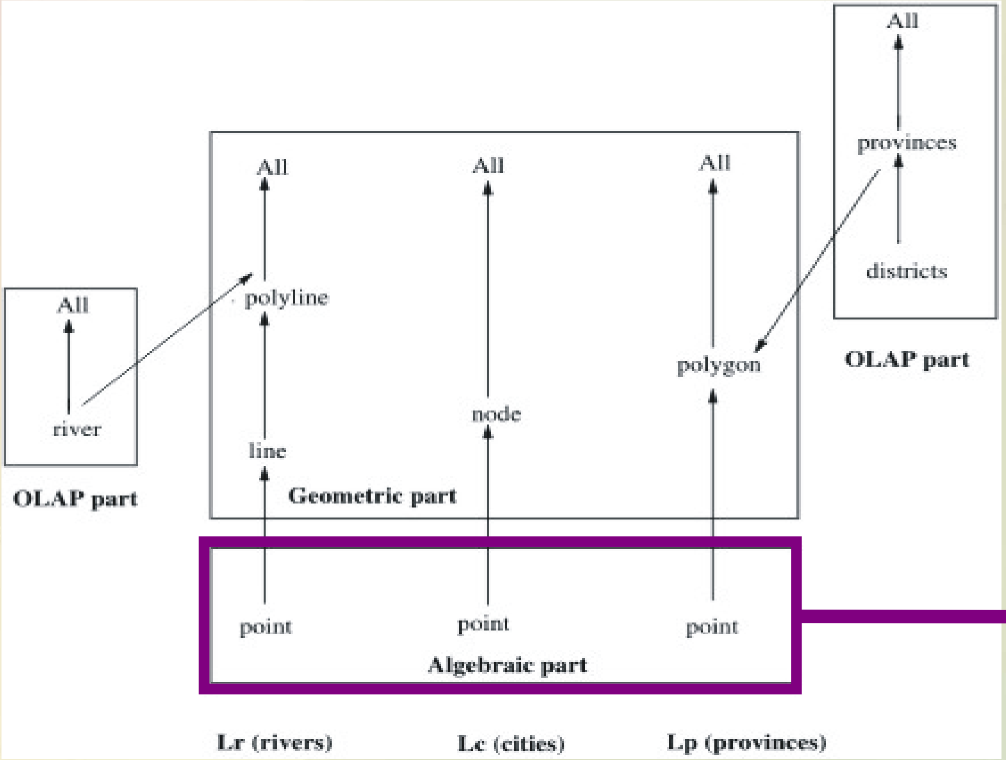
A ***GIS dimension*** is a set of graphs, each one describing a set of geometries in a thematic layer. The dimension is composed of schema and instances.



The ***GIS dimension schema*** has three parts

Data Model Overview

A ***GIS dimension*** is a set of graphs, each one describing a set of geometries in a thematic layer. The dimension is composed of schema and instances.

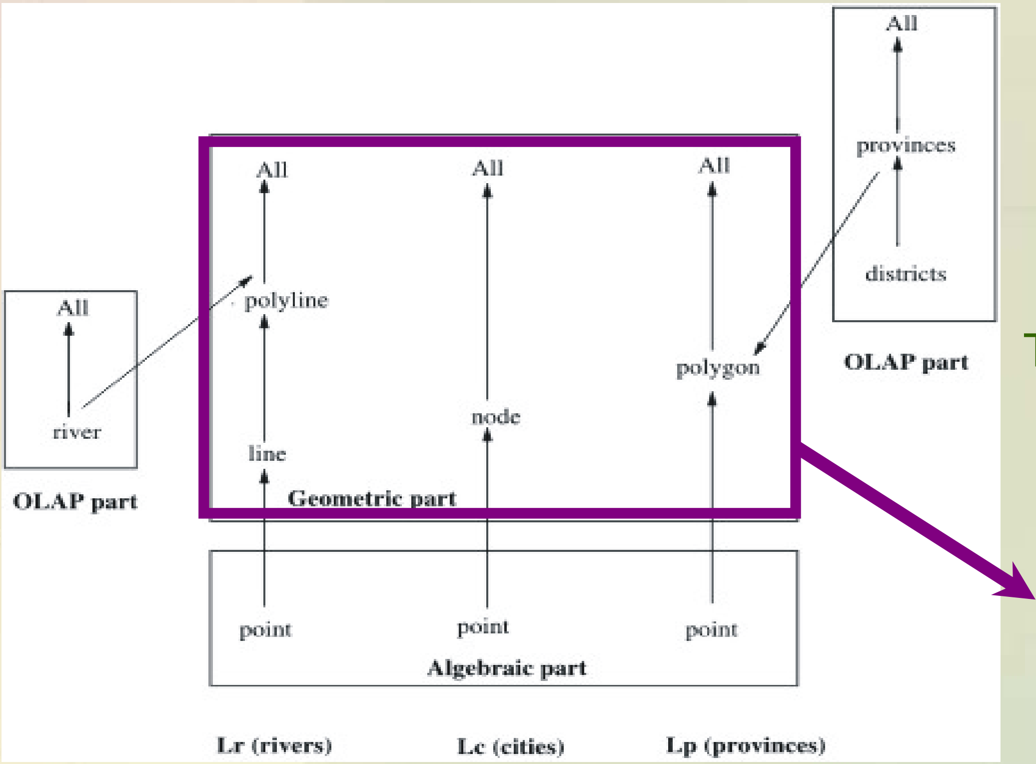


The ***GIS dimension schema*** has three parts

The algebraic part contains the infinite points in a layer and could be described by means of linear algebraic equalities and inequalities

Data Model Overview

A ***GIS dimension*** is a set of graphs, each one describing a set of geometries in a thematic layer. The dimension is composed of schema and instances.

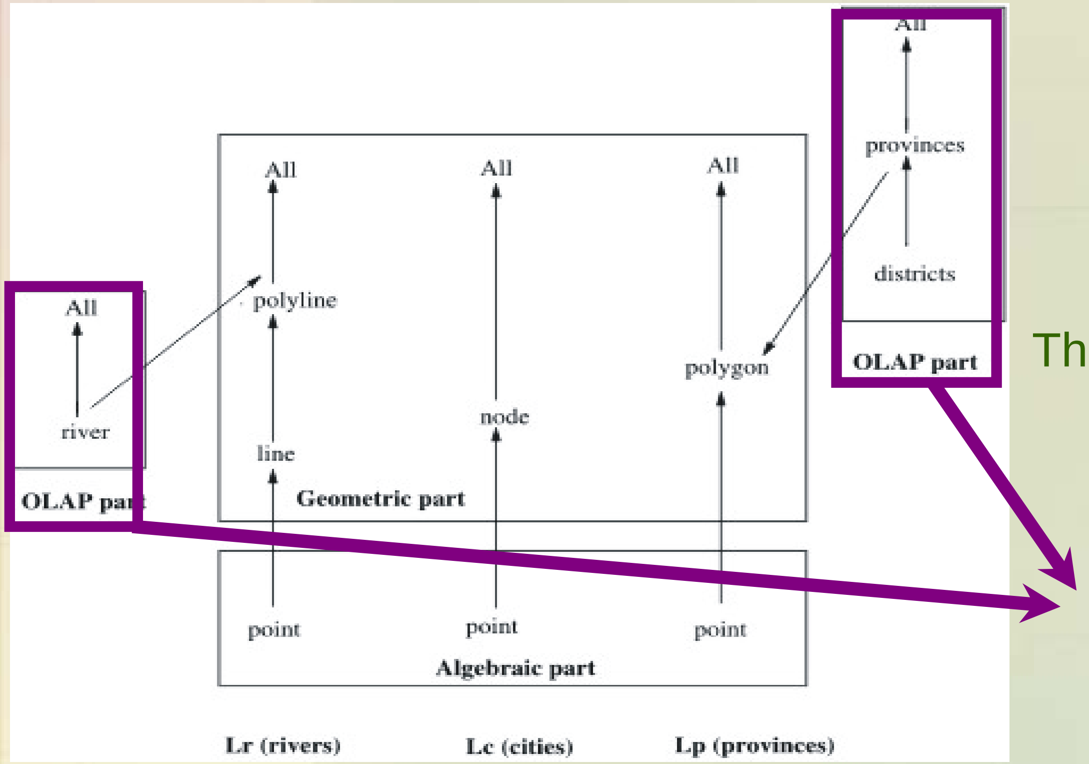


The ***GIS dimension schema*** has three parts

The geometric part stores the finite identifiers of certain geometries (elements of the layers in the GIS)

Data Model Overview

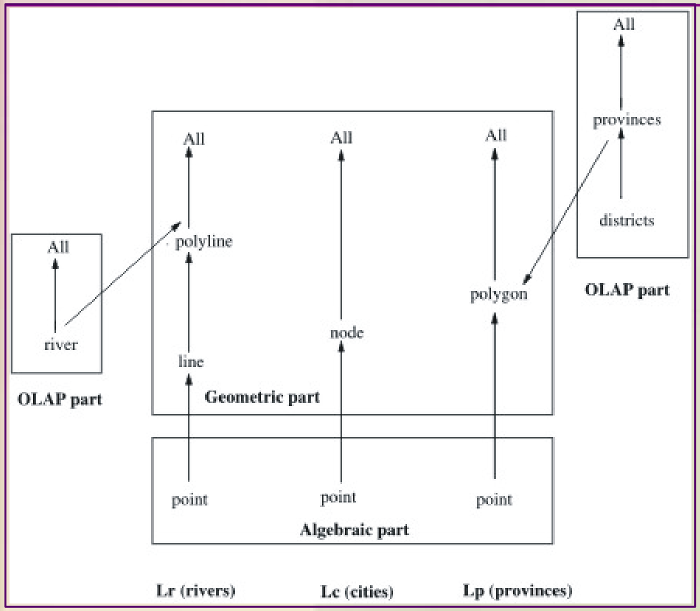
A ***GIS dimension*** is a set of graphs, each one describing a set of geometries in a thematic layer. The dimension is composed of schema and instances.

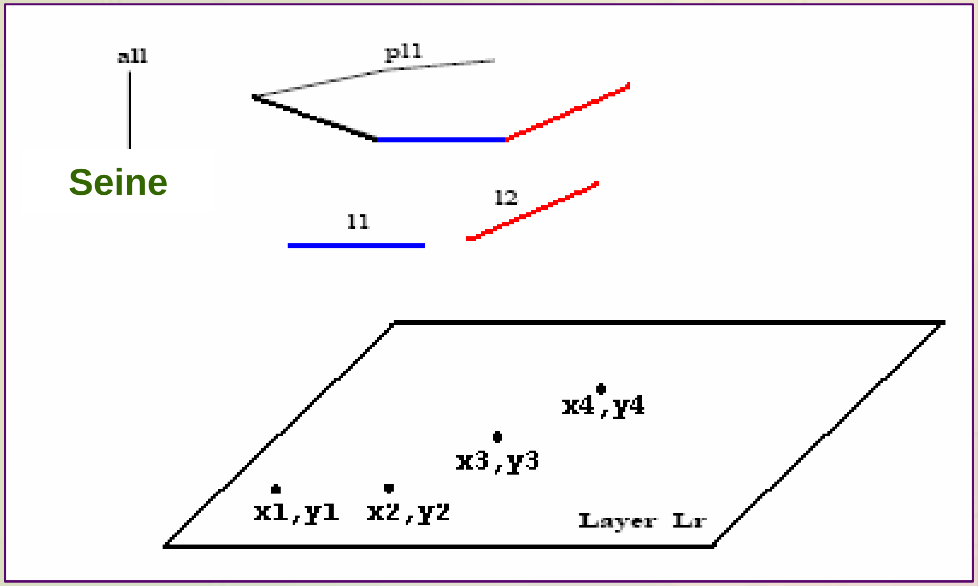


The ***GIS dimension schema*** has three parts

The OLAP part stores non-spatial data

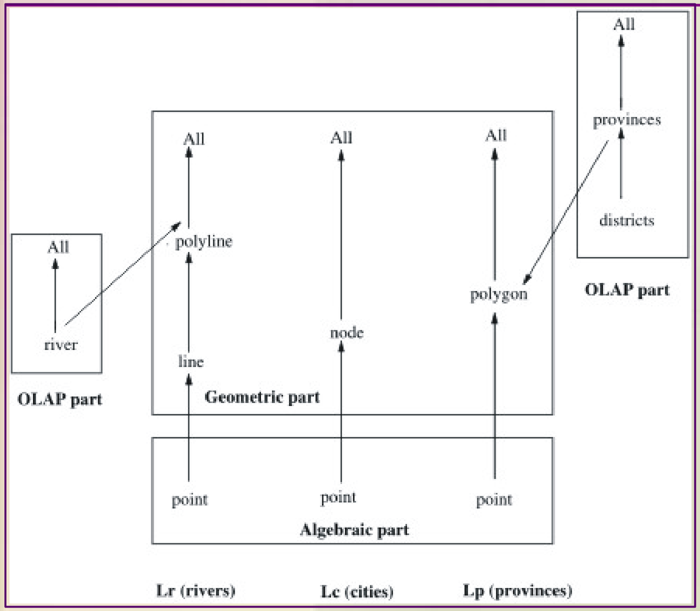
Data Model Overview

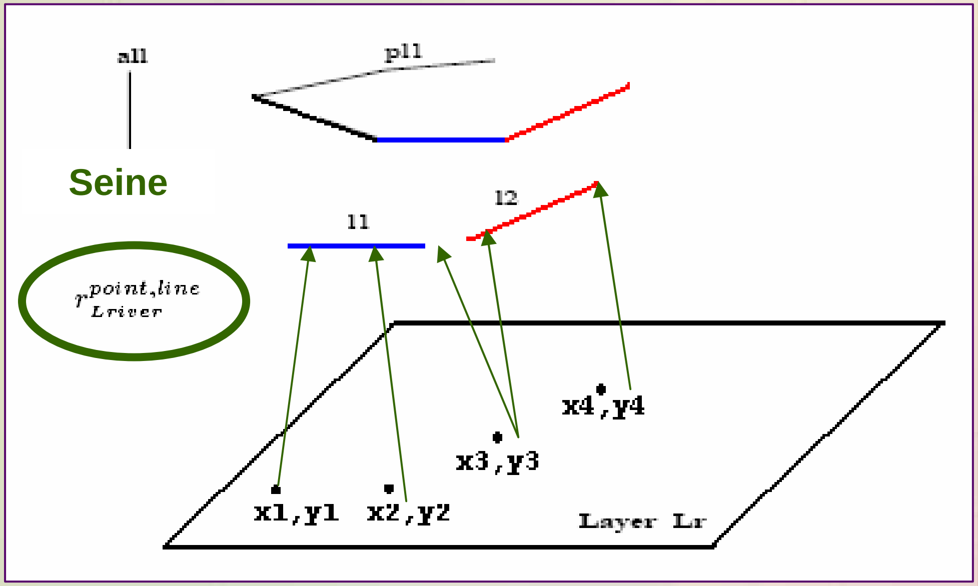




At the ***GIS dimension instance*** we have *rollup relations* between geometries and *association functions*

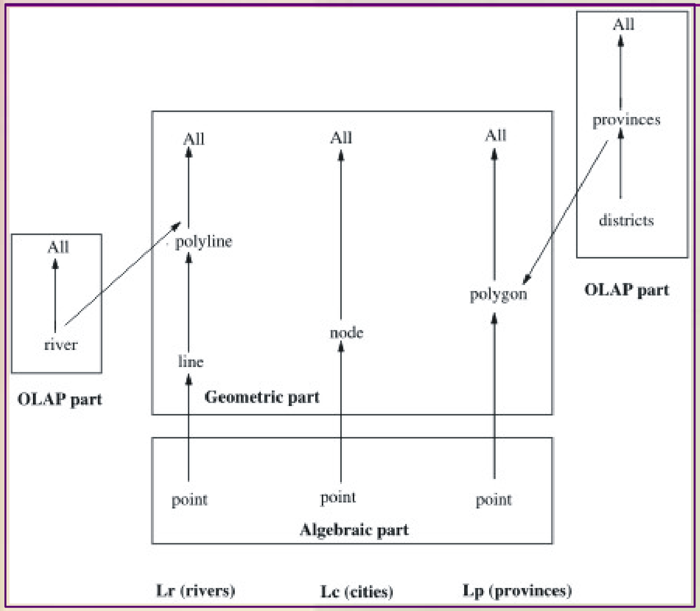
Data Model Overview



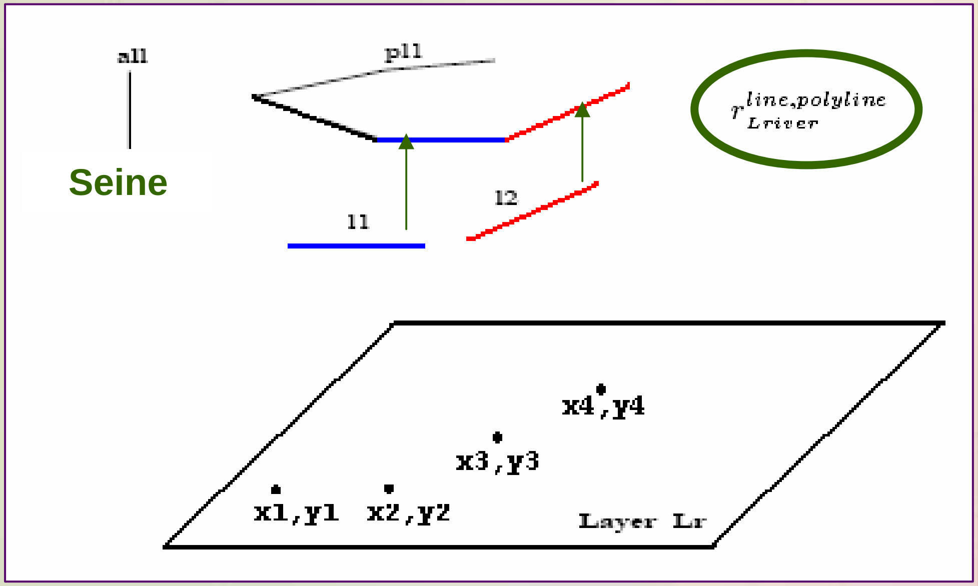


At the ***GIS dimension instance*** we have *rollup relations* between geometries and *association functions*

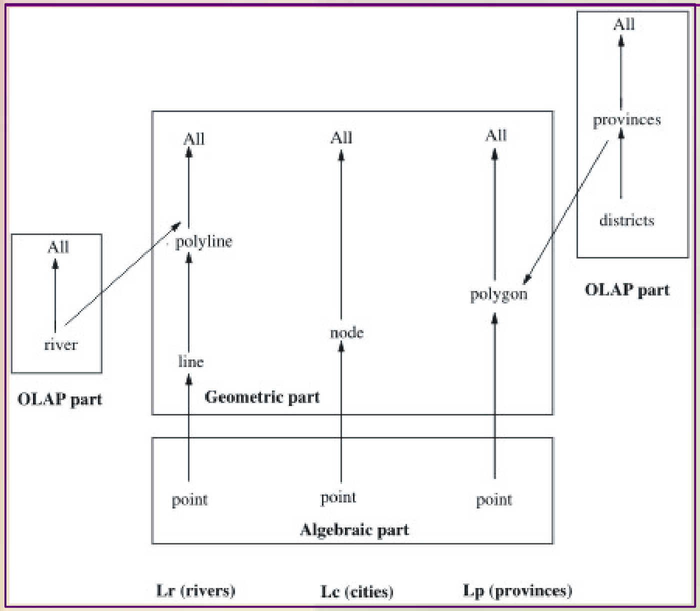
Data Model Overview

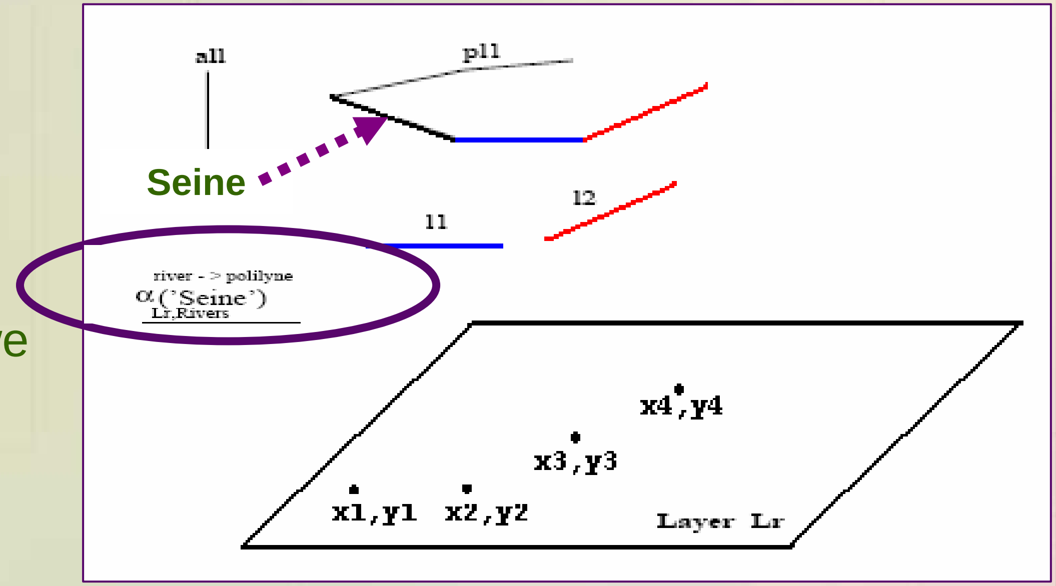


At the ***GIS dimension instance*** we have *rollup relations* between geometries and *association functions*



Data Model Overview





At the ***GIS dimension instance*** we have *rollup relations* between geometries and *association functions*

Data Model Overview

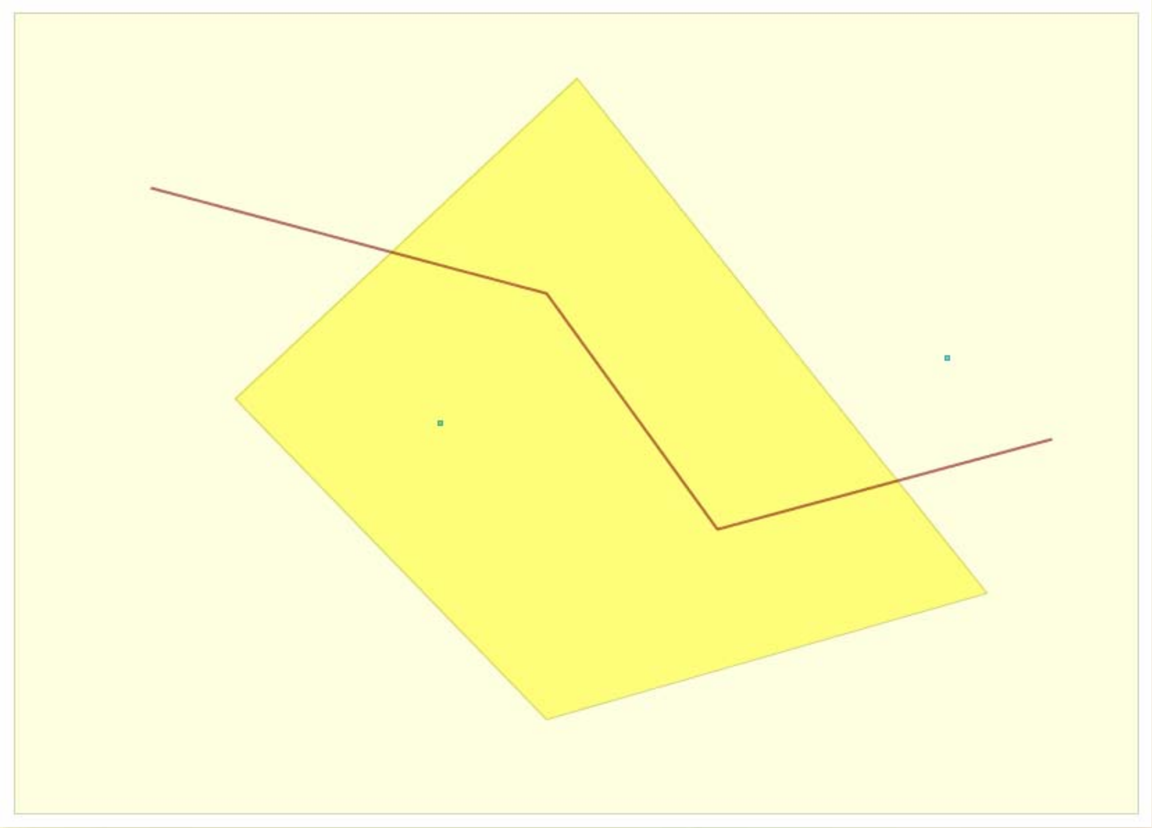
Piet processing technique

1) The *sub-polygonization* strategy

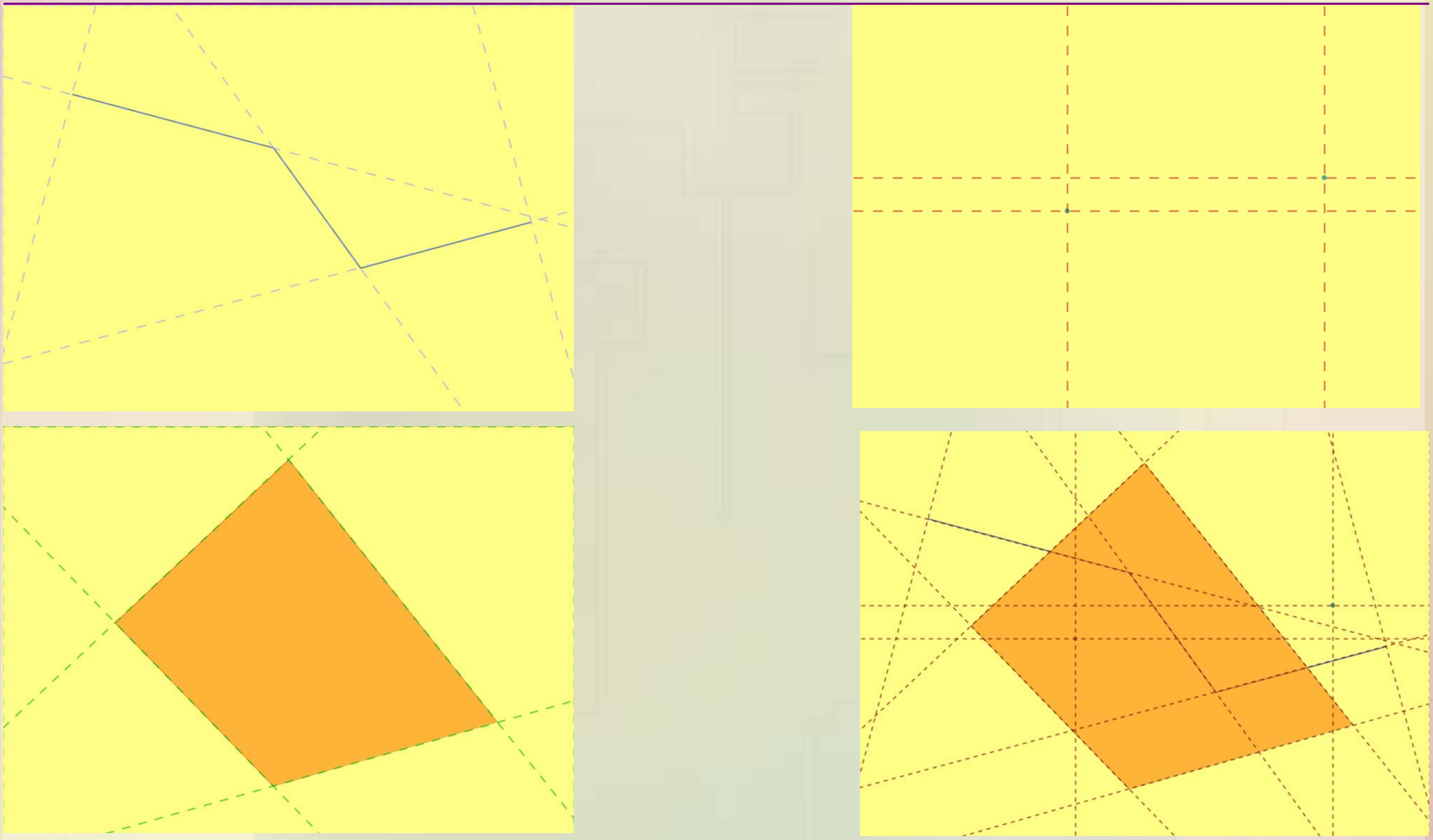
Decomposes each thematic GIS layer into sub-geometries (open convex polygons, open line segments and points) using the concept of carrier lines.

Data Model Overview

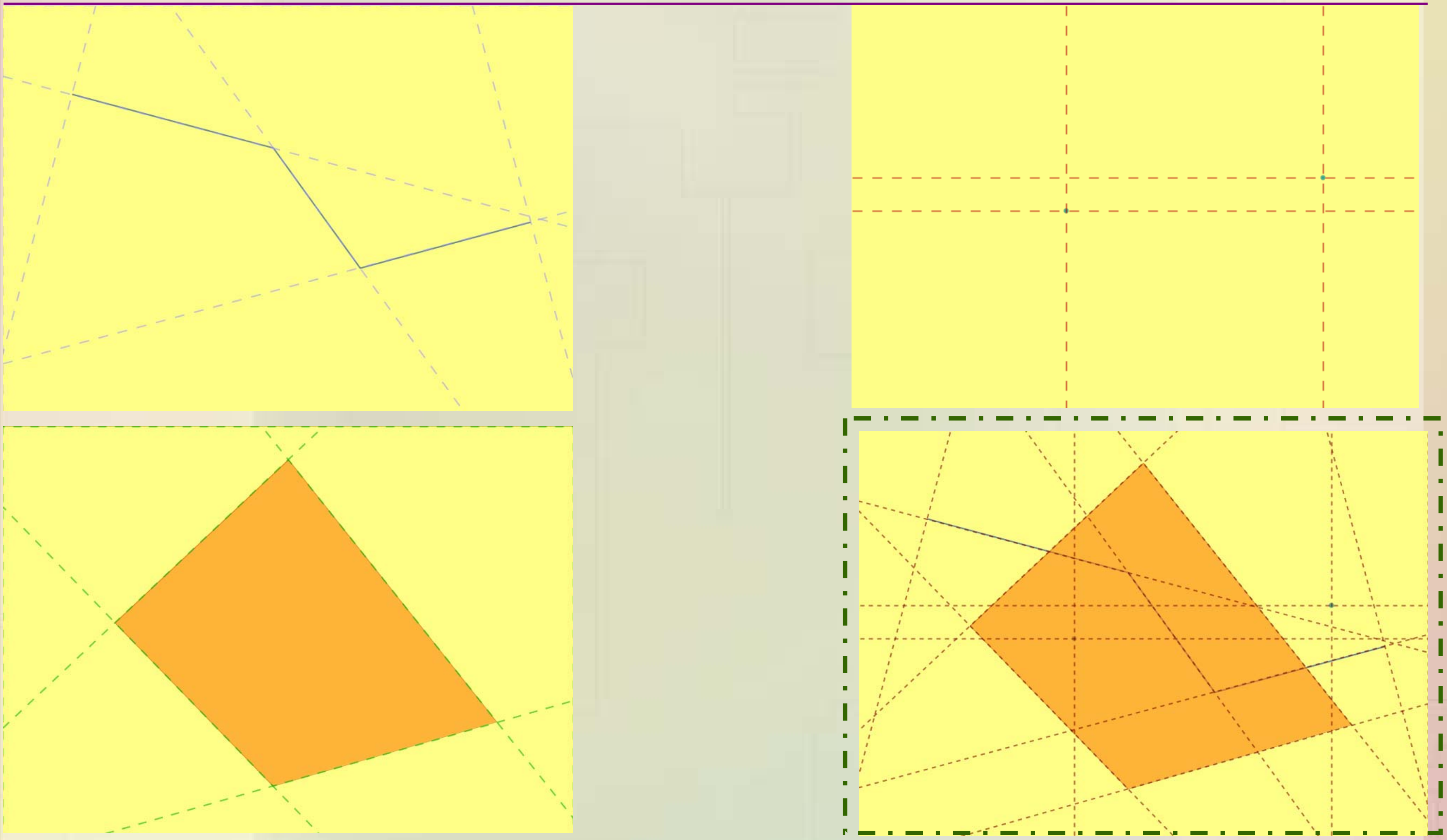
Example



Data Model Overview



Data Model Overview



Data Model Overview

Piet processing technique

1) The *sub-polygonization* strategy

Decomposes each thematic GIS layer into sub-geometries (open convex polygons, open line segments and points) using the concept of carrier lines.

2) *Preoverlay* of layers

All geometries in common are pre-computed and stored in the database.

3) *Evaluate queries* using the pre-computed geometries in common

These shared geometries between layers are used to solve queries.

Data Model Overview

Summable Queries

If we have an aggregate function uniformly distributed over the space, we can easily compute its total value by adding the partial values of the sub-geometries involved.

For example, to calculate the total population of cities crossed by the 'Lis' river we can use the following formula

$$\sum_{g_{id} \in Region} population(g_{id})$$

where the set Region contains the identifiers of the geometries that verify the condition.

In the case the Region contains the IDs of geometries of cities crossed by 'Lis' river. Formally

$$Region = \{g_{id} \mid (\exists x)(\exists y)(\exists pol)(r_{L_{river}}^{point \to polyline}(\mathrm{x, y, pol}) \wedge r_{Lcity}^{point \to node}(\mathrm{x, y, gid}) \wedge \alpha_{L_{river}}^{river \to polyline}(\text{'Lis'}) = \mathrm{pol}\}$$

Outline

- GIS-OLAP motivation
- Query Language
- Data Model
- *Implementation Details*
- Experimental Results
- Conclusion

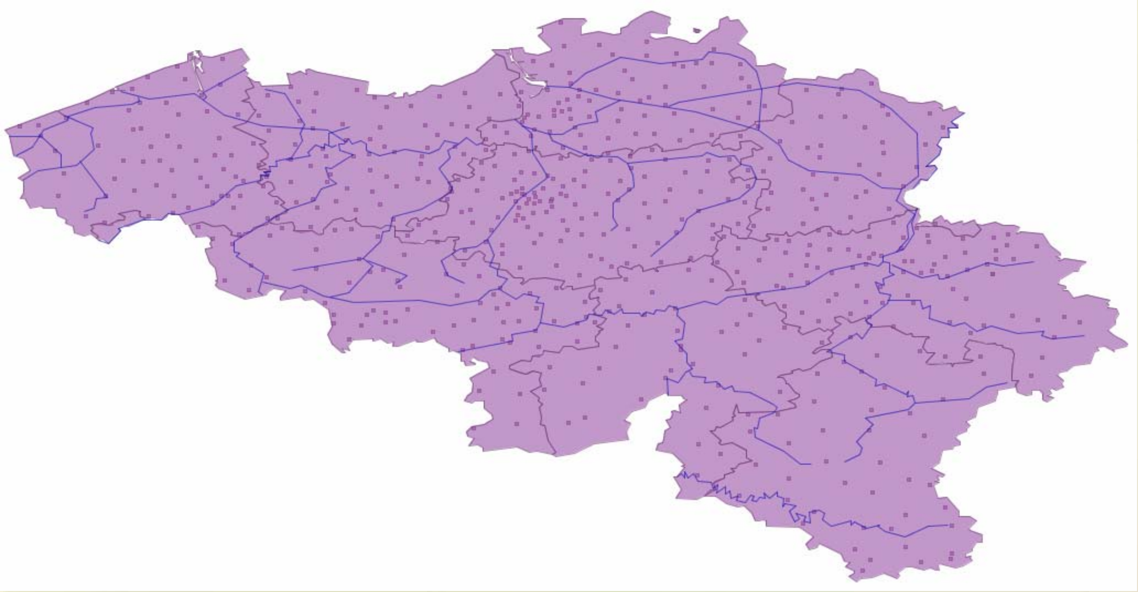
Implementation Details

- PostgreSQL 8.2.3 database with Postgis 1.2 spatial extensions
- Java 1.5
- OLAP Mondrian (MDX query language)
- Xerces
- Castor
- Tomcat Apache 5.5 WebServer (for Web version)
- Jump 1.2 (for stand-alone version)

Implementation Details

Scalability

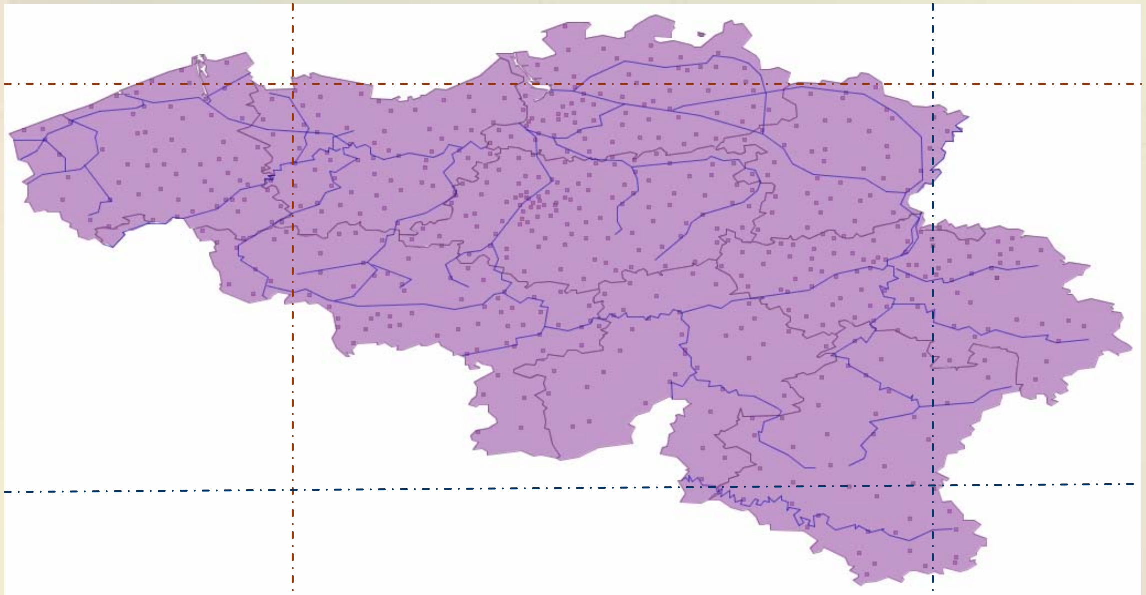
Real-world maps present irregularities: holes, bays, gulfs



Implementation Details

Scalability

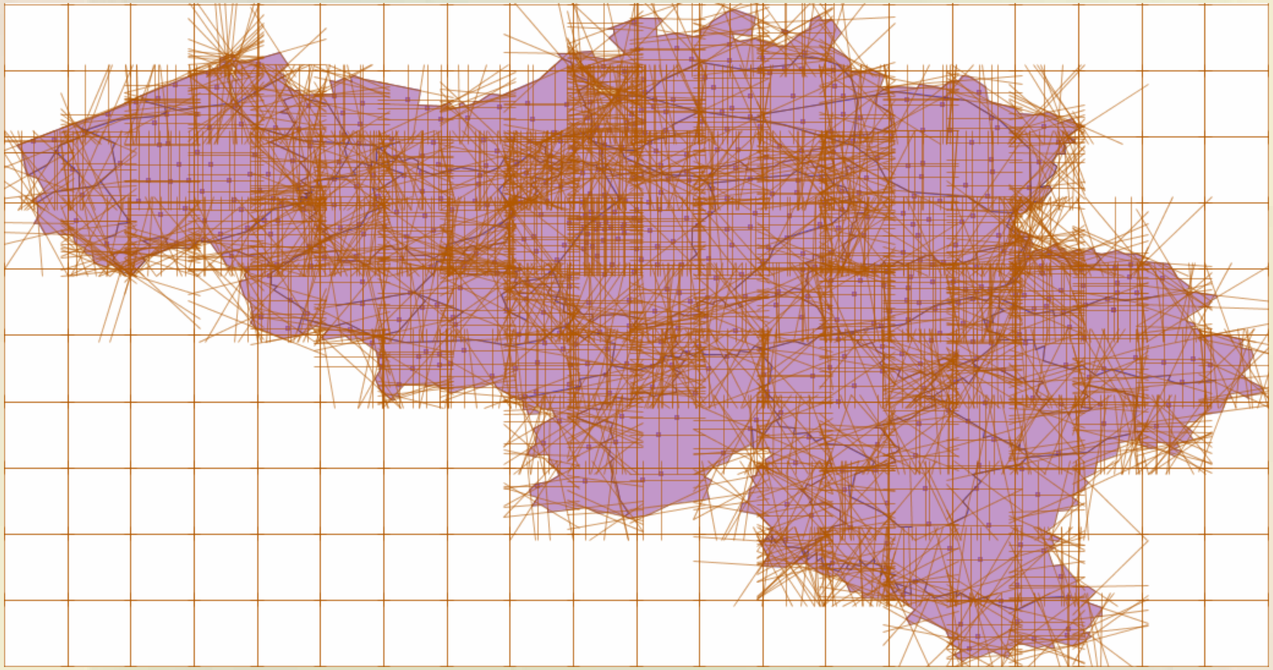
Real-world maps present irregularities: holes, bays, gulfs



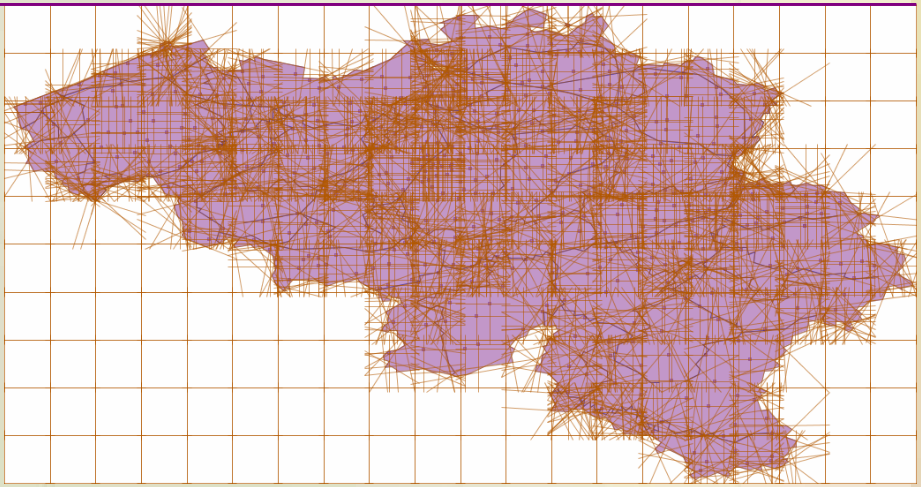
The number of carrier lines induced would become huge, and their interaction become unnecessary as they produce irrelevant partitions and increase the computational cost of algorithms

Implementation Details

Partitioning



Implementation Details



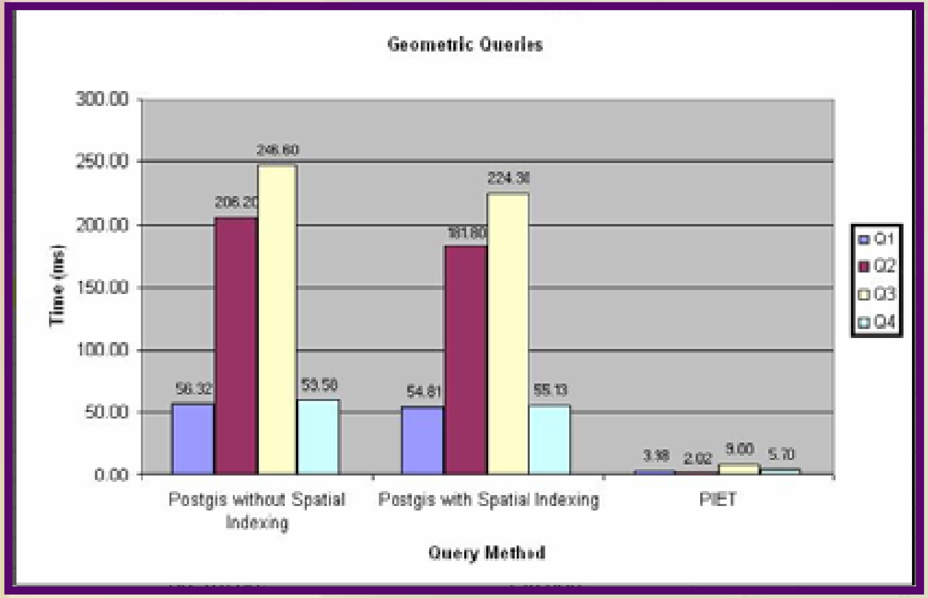
Benefits of Partitioning

1) Reduce the density of carrier lines, and database volume.
2) Partitions that contain no geometries, would not have carrier lines, therefore they do not produce unnecessary storage.
3) The algorithm can easily run in a parallelized environment.
4) Only zones with high density of carrier lines can be further divided into smaller partitions, instead of dividing the entire map in more zones
5) If some zone changes over time, it can be re-computed without affecting the calculated previous zones.

Outline

- GIS-OLAP motivation
- Query Language
- Data Model
- Implementation Details
- *Experimental Results*
- Conclusion

Experimental Results



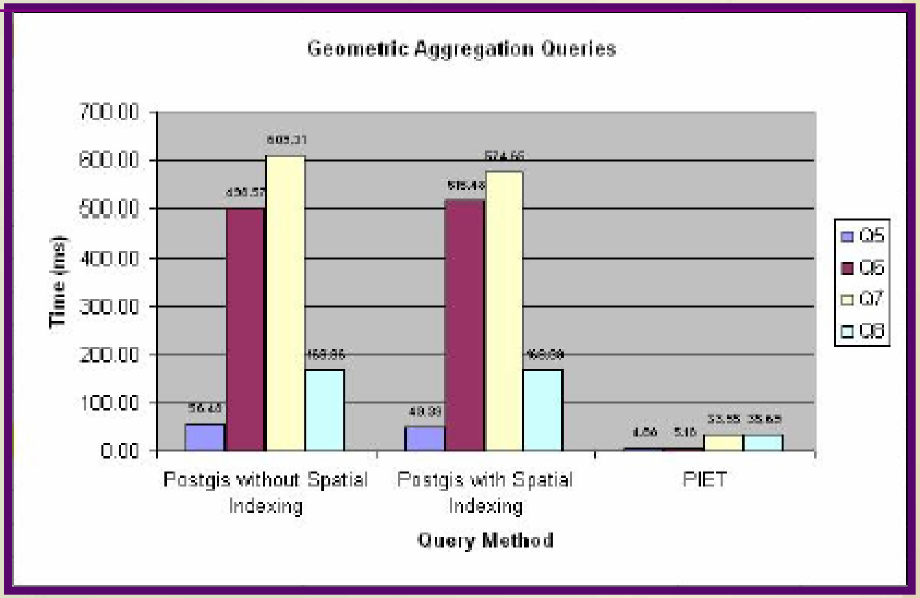
Q1: Districts crossed by at least one river

Q2: Districts and the cities within them

Q3: Districts and the cities within them only for districts crossed by at least one river

Q4: Districts crossed by al least five rivers

Experimental Results



Q5: List each region with the total number of rivers that crossed it

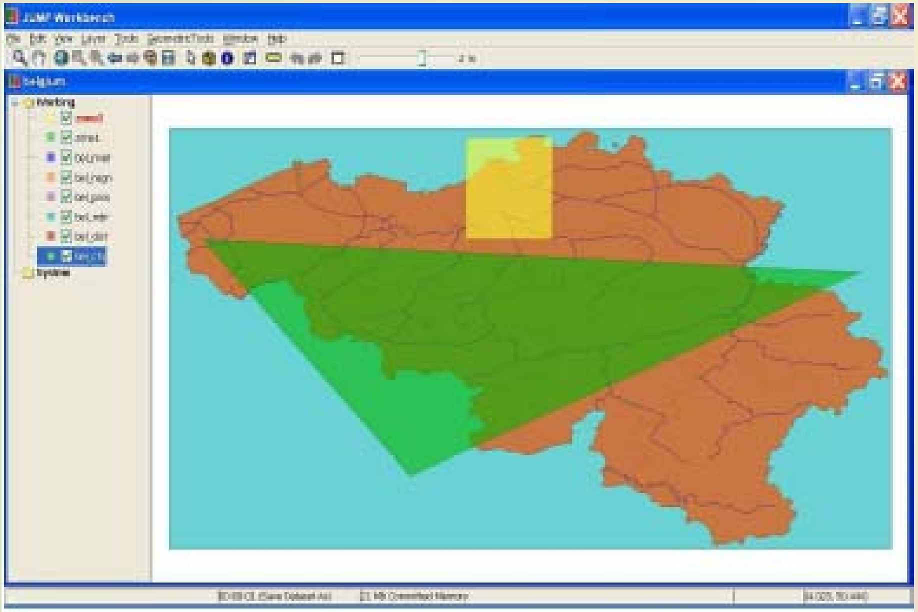
Q6: List each region with the total number of rivers that crossed it, only for regions that contains at least 20 cities

Q7: List each district with the total number of rivers that crossed it and the total number of cities that contains

Q8: For each region show the total length of the part of rivers which intersects it, only for regions with at least an area under cereal cultivation equal or higher than 1000 Km2.

Experimental Results

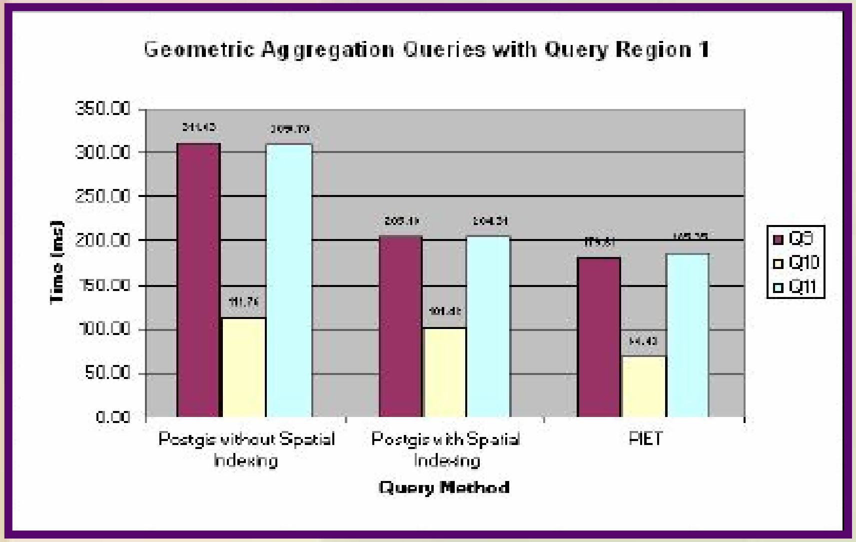
Q9: List each region with the total number of rivers that crossed it, considering only the part of the river that lies within the query region

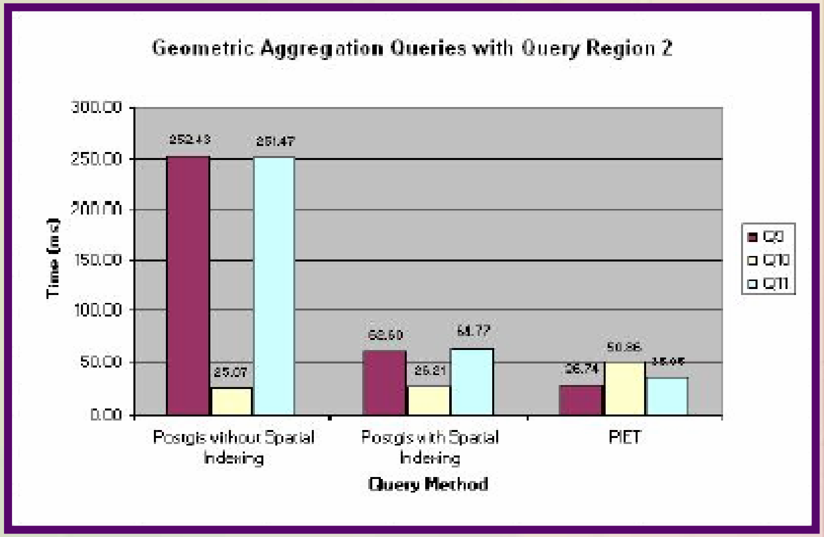


Q10: For each district show the total number of cities, for cities within the query region

Q11: For each region show the total length of the part of each river which intersects it, only for regions containing at least area under cereal cultivation equal or higher than 1000 Km2, considering only the part of the river that lies within the query region

Experimental Results





Outline

- GIS-OLAP motivation
- Query Language
- Data Model
- Implementation Details
- Experimental Results
- *Conclusion*

Conclusion

1) The data model provides a unified view of GIS and OLAP data

2) This data model can be efficiently implemented

3) The GIS-OLAP Query Language introduced is simple and uses well-known sub-query syntax.

4) Our experiments show that *summable queries* can be solved without using special indexing techniques.

We can implement all these features, without waiting for *extenders* to be incorporated in databases. Using tables and B-trees we can reach very good performance

5) Visit our site http://piet.exp.dc.uba.ar/piet

Ongoing Work

Not only integrating OLAP and GIS,

but also spatio-temporal data, in the form of trajectories of moving objects

We are developing specific techniques to reduce the huge volume of the data obtained by sensors without losing relevant information ("Aggregation Languages for Moving Object and Places of Interest Data" [Leticia Gomez,Haesevoets, Bart Kuijpers, and Alejandro Vaisman]).