Chapter 5

PROTOTYPE

A natural skill of people is the interpretation of visual data; therefore, this is a fact we must consider to get advantage during the data mining process. In [30] the authors say that a future direction for the *KDD* research field is the design and use of user interfaces: "one can create a query language which may be used by non-database specialists in their work. Such a query interface can be supported by a Graphical User Interface (*GUI*) which can make the process of query creation much easier".

The challenge consists of improving the capabilities for displaying the generated results (i.e., from a query or the data mining process) in a graphical mode. The idea is that if we are able to analyze the results in such a graphical way, we may give feedback to the user so that he can refine the analysis process and/or guide the direction for further study. This is the principle in relevance feedback (do an initial query, get feedback from the user, and then incorporate information obtained from prior relevance judgments to redefine the query).

We developed a prototype system that provides a graphical user interface to perform the data mining phase (and data analysis) using our proposed graph-based representation for spatial and non-spatial data. The analysis is implemented by using spatial and non-spatial queries and the data mining process is performed with the Subdue system, a graph-based data mining tool. In our research we used two test domains to evaluate our proposal. The first one is a database storing data from the Popocatépetl volcano (see Section 3.4 for details). The other one is a database containing data related to a population census from the year of 1777 in Puebla downtown as described in Section 5.1.

5.1 Population Census from the year of 1777 in Puebla downtown

As our test domain we have worked in the project "*Habitar y vivir. Análisis del espacio habitacional de la ciudad de Puebla 1690-1890*". This is a project directed by Dra. Rosalva Loreto López, a researcher of Urban History. Our objective is to make use of data mining techniques to find interesting relations and patterns between the population and habitation spaces in Puebla downtown during that time period. We argue that our model can find patterns involving non-spatial data (i.e., characteristics of people living in the zone), spatial data (i.e., distribution of the space), and relations between them (i.e., characteristics of houses based on people social status and/or number of people living in a house) in a single pattern.

Figure 5.1 shows the spatial structure we implemented for representing the spatial concepts in the census. The structure has 5 spatial aggregation levels. *Parish* (spatial concept for representing a physical area) is the upper spatial component. A *Parish* is defined by one or more *Neighborhoods*. A *Neighborhood* can involve several *Blocks*. A *Block* is defined by four *Streets* and finally a *Street* gives the location to a *House*, where a family lives.

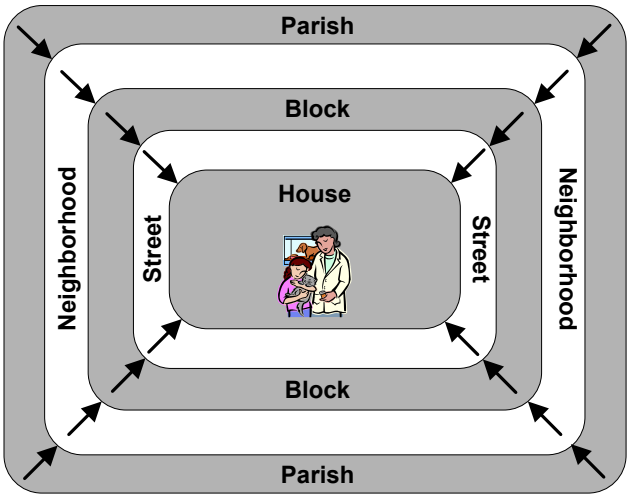

Figure 5.1. Representation of spatial concepts in the census from the year of 1777.

After defining the spatial concepts involved in our domain, we need a graph-based data representation to describe our data as a graph. We have implemented two graph-based representations for the non-spatial data of the census.

These representations have three main components: (1) **HOUSE** involves the attributes describing a *House*. It contains one or more *Uh* (atomic physical area where a family lives). One or more families might inhabit in a *House*, but each family lives in an *Uh*. (2) **UH** involves the attributes describing the living space. (3) **MEMBER** involves the attributes describing a member of a family.

Figure 5.2 shows the first representation created for the population census. A description of the representation is as follows: a *Parish* contains one or more *Neighborhoods*. A *Neighborhood* contains one or more *Blocks* or *Locations* (spatial concept for identifying with precision a physical area -i.e., north of-). *Block* and *Location* contain one or more *Streets*. A *Street* contains one or more **HOUSES**. A **HOUSE** has two attributes (*NCasa* -Id assigned to the *House*- and *NHabCasa* -number of *Uh* in a *House*-). A **HOUSE** contains one or more **UH**. An **UH** has several attributes describing it (*Uh* -Id assigned to the *Uh*-, *Etnicidad* -predominant ethnic group among the members of a family-, *TFamilia* -type of family, i.e., family with children-, *TUh* -type of *Uh*- and *NMiembrosFamilia* -number of members integrating a family-). An **UH** contains one or more **MEMBERS**. Finally a **MEMBER** has several attributes describing it (*JFamilia* -head of family-, *TitPersona* -title of the member, i.e., don, doña-, *Nombre* -first name-, *Apellido* -last name-, *Sexo* -sex-, *EdoCivil* -marital status-, *Parentesco* -social relationship with respect to the head of family-, *GpoEtnico* -ethnic group- and *Edad* -age-).

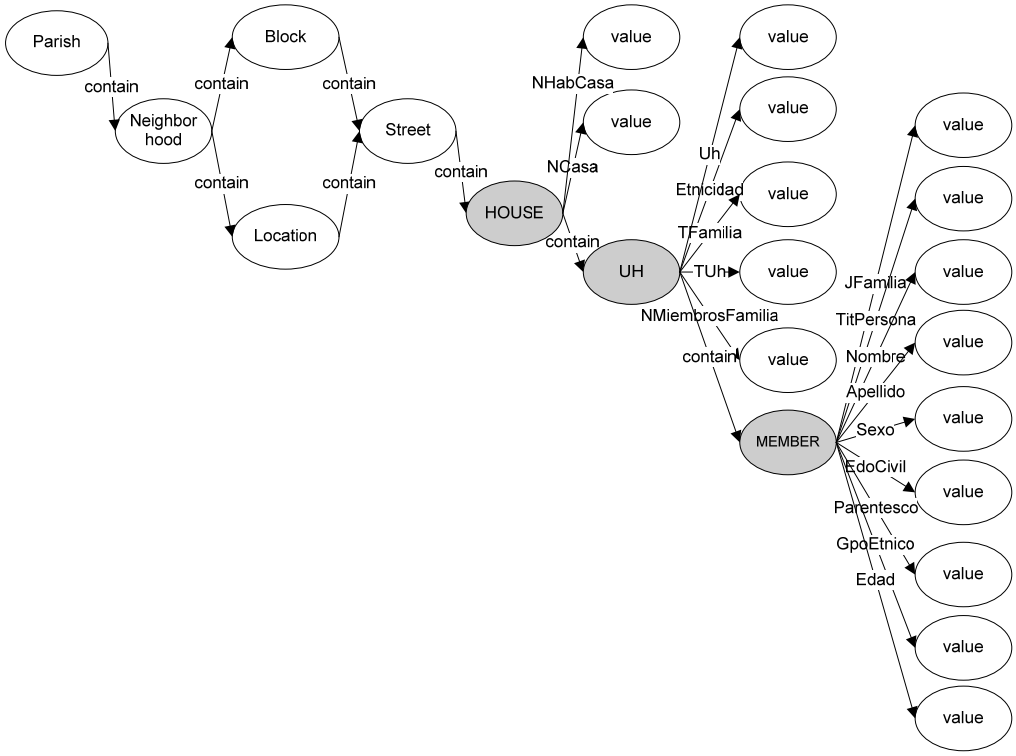

Figure 5.2. First representation for the non-spatial data in the population census from the year of 1777.

The second representation is presented in Figure 5.3. The representation can be read as follows: A **HOUSE** is the main item. A **HOUSE** has several attributes describing it (*Parish*, *Neighborhood*, *Block*, *Location*, *Street*, *NCasa* and *NHabCasa*). A **HOUSE** contains one or more **UHS**. An **UH** has several attributes describing it (*Uh*, *TUh*, *Etnicidad*, *TFamilia*, *NMiembrosFamilia*). In a **UH** lives (contains) one or more **MEMBERS**, and finally, a **MEMBER** has several attributes describing it (*Jfamilia*, *TitPersona*, *Nombre*, *Apellido*, *Sexo*, *EdoCivil*, *Parentesco*, *GpoEtnico* and *Edad*).

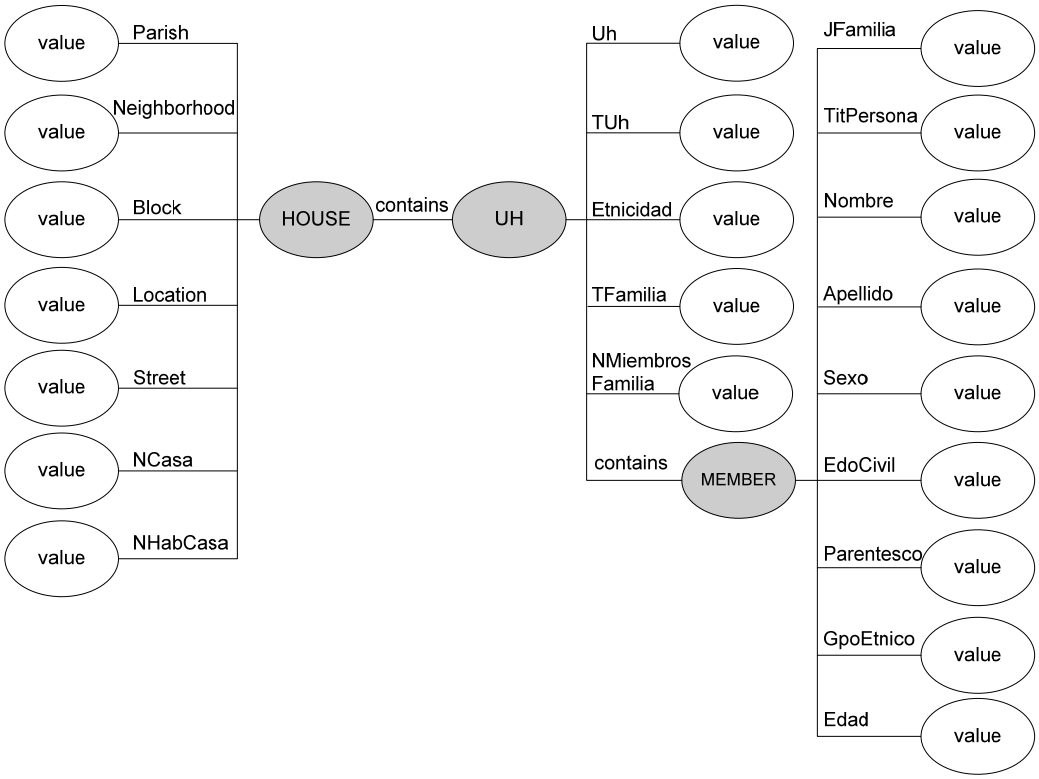

Figure 5.3. Second representation for the non-spatial data in the population census from the year of 1777.

5.2 Modules

This section presents a prototype system developed for testing our graph-based data representation model for spatial data mining as proposed. The prototype is implemented in Java. We use *Oracle DBMS version 9i* for storing and processing our data. Oracle has a module to manage spatial data named Oracle Spatial; we take advantage of the spatial operators, geometry functions and spatial aggregate functions implemented in the module for either identifying the objects in a region over a spatial layer or obtaining/validating the spatial relations between two spatial objects. The prototype is divided into seven modules:

Data Preparation and Cleaning. The data preparation and cleaning phase is a step of the Knowledge Discovery in Databases (*KDD)* process. Our module helps the user to validate the data and creates the necessary structures to store it in the database. In this process, the data mining expert and the user must work together to identify the activities to be performed that are related to the process (i.e., to identify noise and remove it, treat missing values, etc). Once these activities have been defined, the process is transparent to the user when adding more data (belonging to the same database schema for the domain).

Query. We have implemented the interface shown in Figure 5.4 to allow the user to create non-spatial queries. The goal is to allow the user, to make use of spatial and non-spatial attributes, to query the database and to build graphs from the obtained results.

The *Query* panel allows the user to query the database using an *SQL-like* approach. The queries are created in real time and then submitted to the database for answering the user request. The results are presented to the user in two ways. The first uses a relational approach and the second is represented over a map.

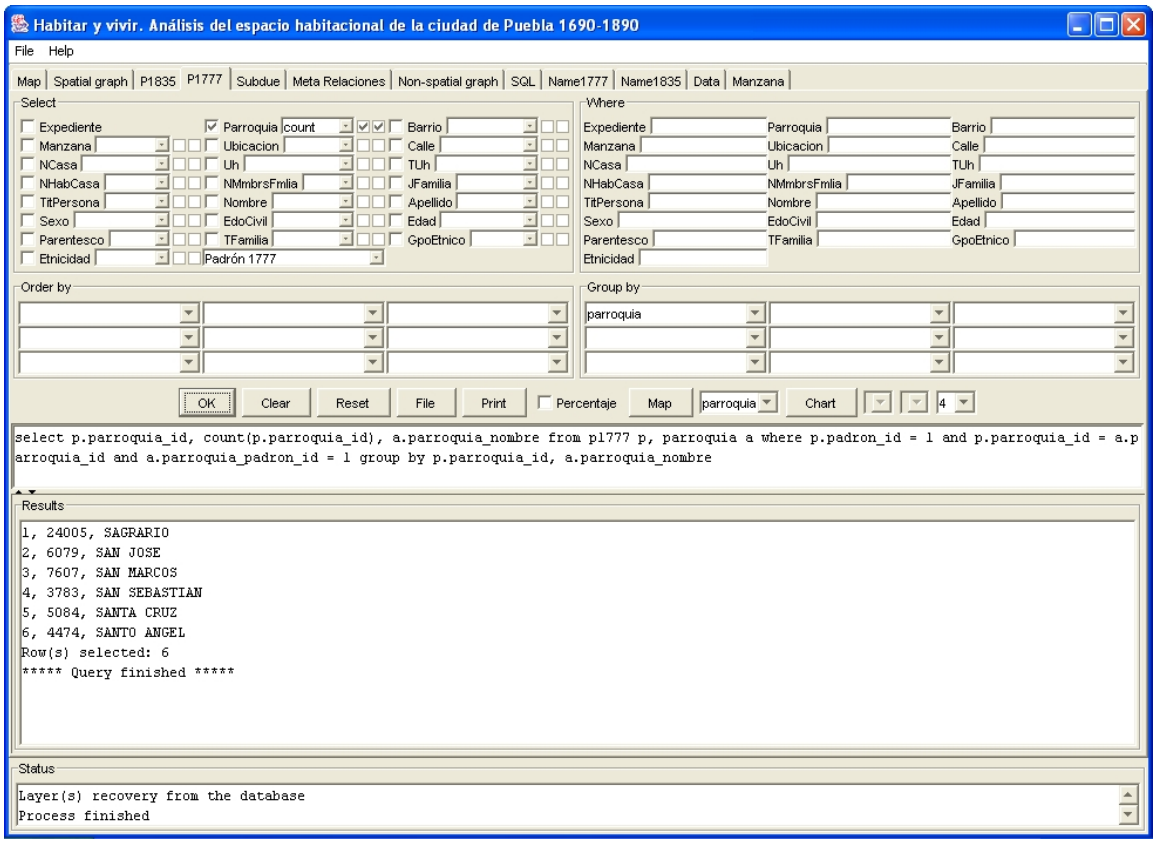

Figure 5.4. The query panel.

The interface is divided in 2 sections, the *Control* and the *Results* sections. The *Control* section is divided in 4 subsections: *Select*, *Where*, *OrderBy* and *GroupBy*. The Results section is divided in 2 subsections: *Query visualization* and *Generated results*.

In the *Control* section the user creates the query. The query is created by specifying the 5 most usual clauses in a *SQL* statement: *Select–From-Where-OrderBy-GroupBy*. The first two elements are mandatory, the last three are optional.

The *Select* subsection presents the twenty-two fields that compose the core of the database. Twenty one of these fields have four options implementing query functionalities as follows: *option1 – attribute name – option2 – options3 – option4*. The first option is used to indicate if the field will be included in the query. The second option contains the keywords used to implement, currently, five grouping functions: "*count*", "*count*(*distinct*())", "*distinct*", "*max*" and "*min*". These keywords are used to group the data by the field(s) selected in the *GroupBy* subsection. Options three and four are used to indicate if the descriptive attributes associated to the field will be included in the query. If the first option is not selected then all the other options are ignored. Additionally, the Select subsection has an item containing the name of the schema (i.e., 1777 census) to create the "*from*" clause of the query.

The *Where* subsection is used to indicate the set of conditions, by field, for restricting the rows to be selected (the dataset returned by the query). A condition specifies a combination of one or more expressions composed of attribute names, attribute values, and logical operators. These conditions are entered manually by the user, so knowledge about the domain is required.

The *OrderBy* subsection allows the user to indicate if he wants to sort the rows returned by the query and which field(s) will be used to perform the operation. We can sort the rows returned using the twenty-two fields composing the database and we can also implement combinations of these elements (i.e., sorting by the *Name* and *Sex* fields).

The *GroupBy* subsection was implemented to allow the user to group the selected rows based on the value(s) of each row and return a single row with a summary of the

information for each group. We can group the rows using twenty-one of the twenty-two fields of the database and we can also implement combinations of these elements (i.e., grouping by the *Parish* and *Sex* fields). The grouping aggregation function (i.e., *count* or *sum*) is chosen in the *Select* subsection. As we have already mentioned, we have implemented five grouping functions.

Our prototype system creates queries by selecting the different fields and options contained in the four previous sections. This query is created in real time, displayed in the *Query visualization* area and then submitted to the *DBMS* for processing. The obtained result is displayed in the *Generated results* area; it is presented by using a relational approach (rows and columns).

SQL. Using this interface the user can create a query by typing it directly (in manual form). The principle is the same as we described in the *Query* panel, we want to create queries and to use the results of the queries to create graphs (based in our model). These graphs will be the data source for our data mining algorithm so that we can find patterns that allow us to understand and describe our data.

In the *Query* panel (see Figure 5.4) the user creates the query in real time by using a graphical interface but if he wants to modify it, it is not possible (the query is displayed just for visualization purpose). Each time the user creates a query in the *Query* panel, the created query is copied to the *Query visualization* area in the *SQL* panel so if the user wants to enhance or modify the query he is able to do it. The generated results are presented to the

user in the *Generated results* area. Both interfaces (*Query* and *SQL* panels) provide the user with the capability to send the generated results to a text file or a printer.

Map. Another way to create graphs in the prototype is using the *Map* panel. In this case the interface displays in a graphical way the spatial layers stored in the database (see Figure 5.5).

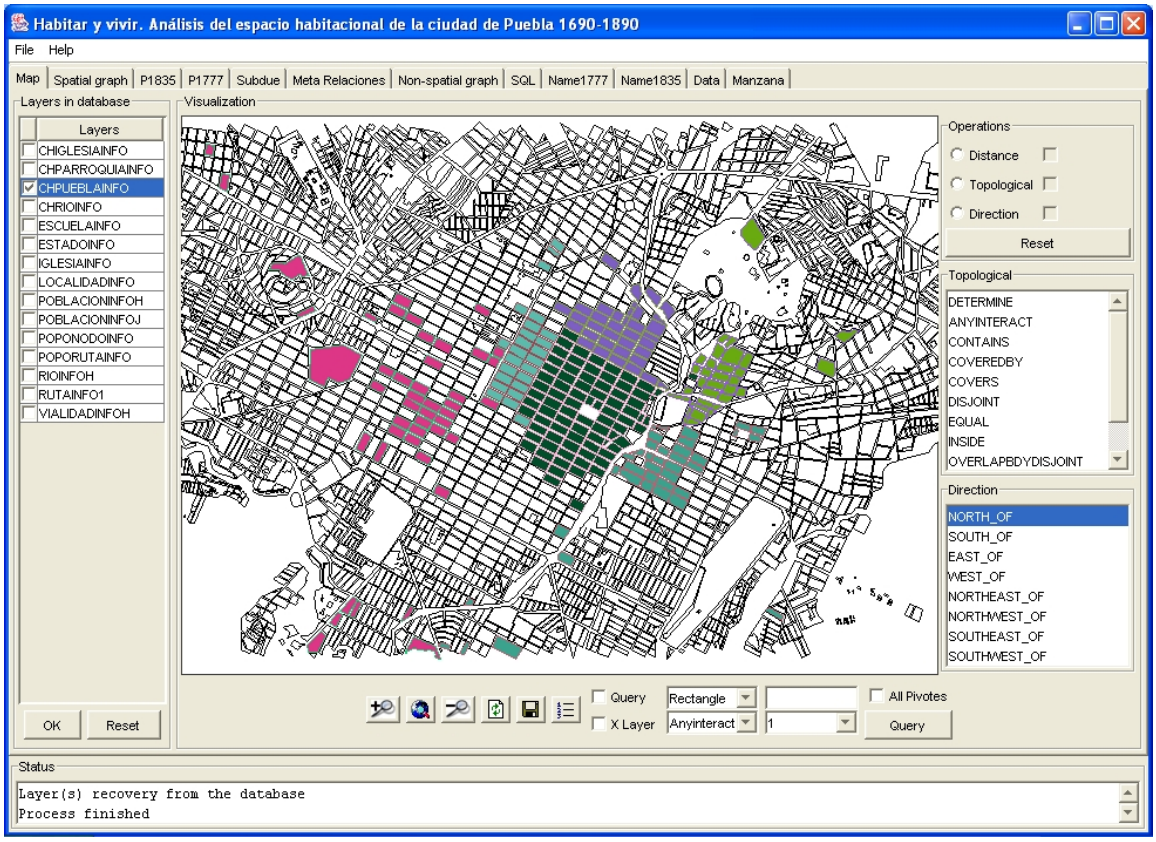

Figure 5.5. The map panel.

By using the interface, the user can delimit the set of spatial and non-spatial data that will be used for creating the graphs. Some times the user wants to analyze only some regions in a spatial layer so it is not necessary to include all the data in the graph. Additionally, we

have to take into account that if we have a huge database we will build huge graphs and this feature has a direct impact over the data mining algorithm, so this is an important issue we have to face. A solution for facing the problem of creating huge graphs is delimiting the set of elements to be included in it by using *selection windows* as we mentioned in Chapter 3.

A second method for delimiting the set of elements to be considered while creating a graph is by using the results of the non-spatial queries created and processed in the *Query* and *SQL* panels. This is implemented by a process that identifies the spatial objects involved in the results generated by a non-spatial query (i.e., a query computing the number of people, grouped by *Sex*, living in each *Parish* in the census from the year of 1777 so we can select and show the *Blocks* or the *Streets* belonging to each *Parish* over the map).

The interface is divided in four sections: *Visualization area*, *Layers in database*, *Operations control* and *Map control*. The *Layers in database* section displays the name of the spatial layers stored in the database. The component is also used for selecting and identifying the spatial layers to work with.

We also include information about the spatial relations to be considered in the graph. The *Operations control* section includes the operations (*Topological*, *Distance* and *Direction*) implemented to validate the spatial relations among spatial objects. In the case of the topological relations we have implemented the validations supported by the Oracle Spatial module.

The *Map control* section has five buttons implementing the *Zoom in*, *Zoom out*, *Show all*, *Reset all* and *Save map* operations. The *Visualization area* works in two operational modes: *Query mode* for creating *selection windows* and *Zoom mode* for implementing the *Zoom in* and *Zoom out* operations. The option "*X Layer*" is used to indicate if the validation of spatial relations among spatial objects will just be among objects belonging to different spatial layers (i.e., objects belonging to the *Parish* and *Neighborhood* layers) or among all objects belonging to all layers. The last two options are used to control the characteristics of the *Selection window*: a *Selection window* tool can be a *Rectangle* or a *Circle*, and we can specify if we want to select just the elements inside the *Selection window* or the elements inside and touching its border.

Once the user has selected the working area, the following steps identify the objects inside the area and validate the spatial relation(s) among them. We have implemented the validation of topological, distance and direction relations; only the objects meeting the relation(s) chosen by the user will be candidates to become objects in the graph.

Spatial Graph. The next task is to create the graph. First, the user must select the non-spatial attribute(s) describing the spatial objects in the graph (see Figure 5.6). Remember that the spatial objects and spatial relations among the objects that will be included in the graph were selected in the *Map* panel.

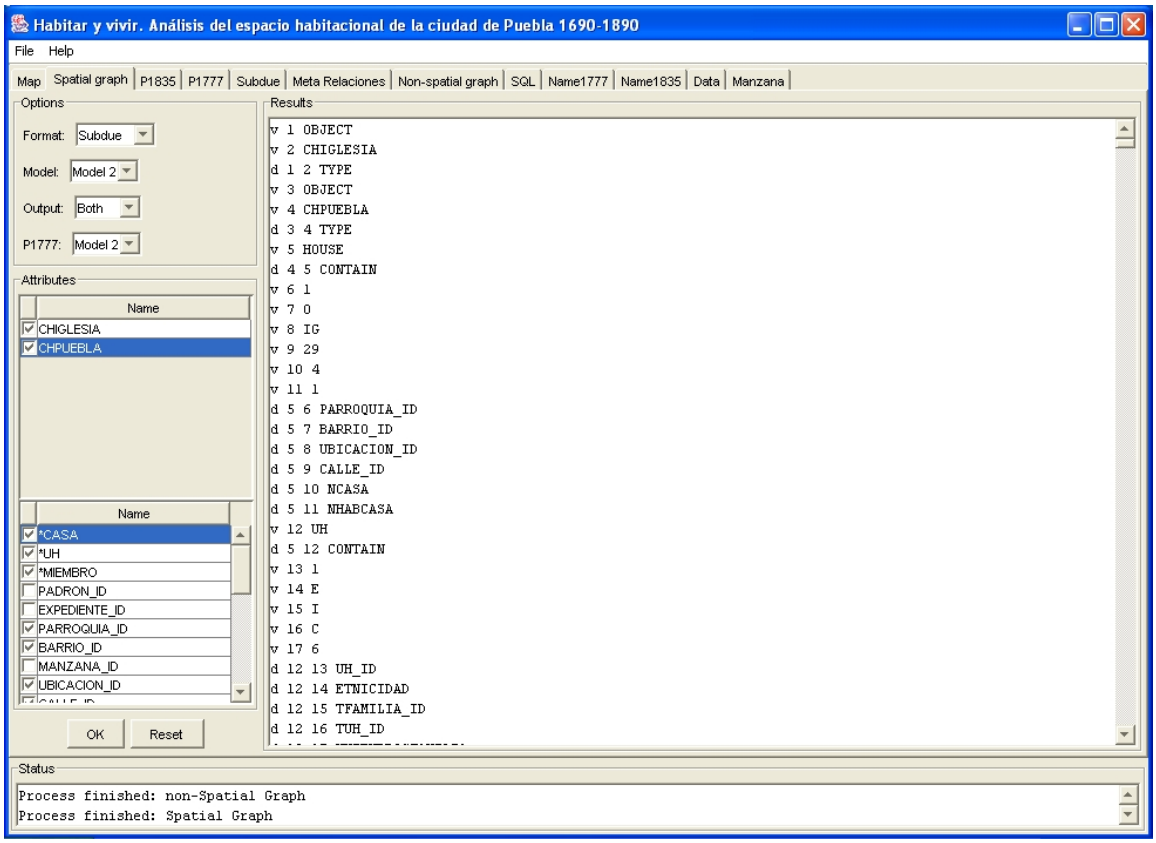

Figure 5.6. The spatial graph panel.

By using this interface, the user can select the non-spatial attributes, for each spatial layer he works with, that will be related to the spatial objects in the graph. For instance, suppose the user is working with the spatial layers *A* and *B*; the spatial layer *A* has five non-spatial attributes and the spatial layer *B* has three non-spatial attributes. So, he may select from layer *A* two attributes and from spatial layer *B* three attributes. The idea is to give the user the capability to select the non-spatial attributes, for spatial layer, that he considers relevant for the mining task.

From the *Graph characteristics* section the user defines the format, model and output device characteristics for the graph. The graph generated by the system can be created following the Subdue or the Graphviz [19] layouts. In the first case the graph is created for feeding the Subdue system, and in the second case it is created for visualization purposes. Currently, we have implemented five graph-based representation models in our prototype. Each model expresses a representation proposal for creating graphs involving the three basic elements found in a spatial database (spatial, non-spatial data, and spatial relations among spatial objects). The resulting graph can be visualized on the screen or stored in a text file. The last option was implemented in order to provide the Subdue and Graphviz systems with their corresponding input files.

We can see at the right side of Figure 5.6 an example of a created graph using the Subdue layout. This sample graph was generated from 2 spatial layers (i.e., chiglesia and chpuebla). From each of them the user selected one or more attributes that were related to the spatial objects. Figure 5.7 presents a fragment of the same graph but this time it is drawn using the Graphviz system.

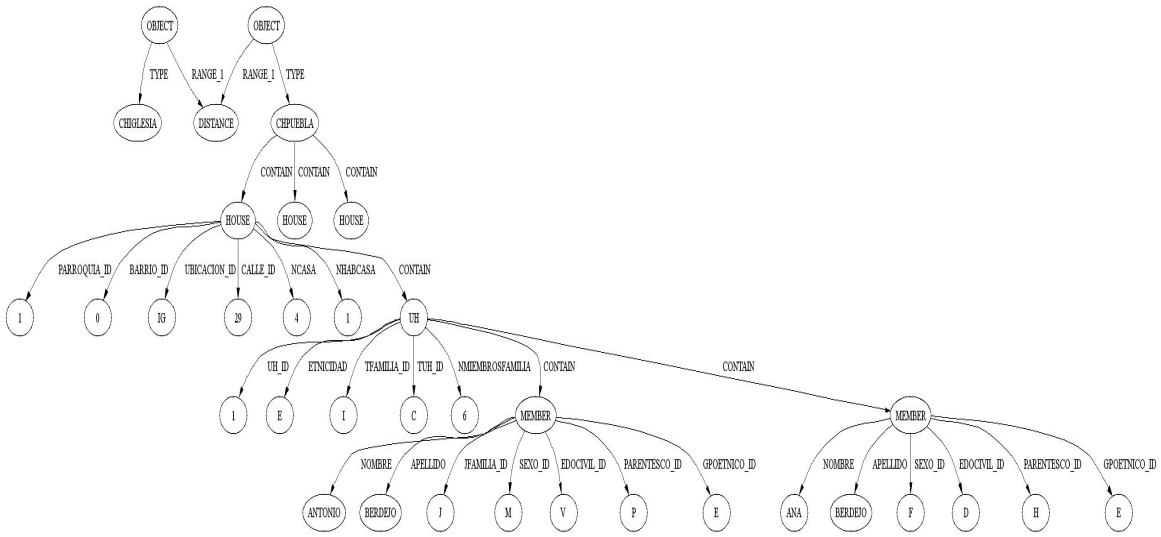

Figure 5.7. Graph representation of processed data.

Non-Spatial Graph. Focusing in the "*Habitar y vivir. Análisis del espacio habitacional de la ciudad de Puebla 1690-1890*" project we have implemented an interface for allowing the user the creation of graphs based on the two graph-based representations created for the non-spatial data of population census (they are shown in Figure 5.2 and Figure 5.3). These graphs are created using only non-spatial data. Our objective for implementing graphs with this characteristic is to have metrics that allow us to compare and to evaluate the results generated by the data mining algorithm when we use graphs containing spatial data, non-spatial data, and spatial relations at the same time against graphs containing only non-spatial data.

The interface implemented is shown in Figure 5.8. The user selects the non-spatial attributes and defines the settings that will be used for creating a query (as in the spatial graph). The result obtained from the query is used for creating the non-spatial graph.

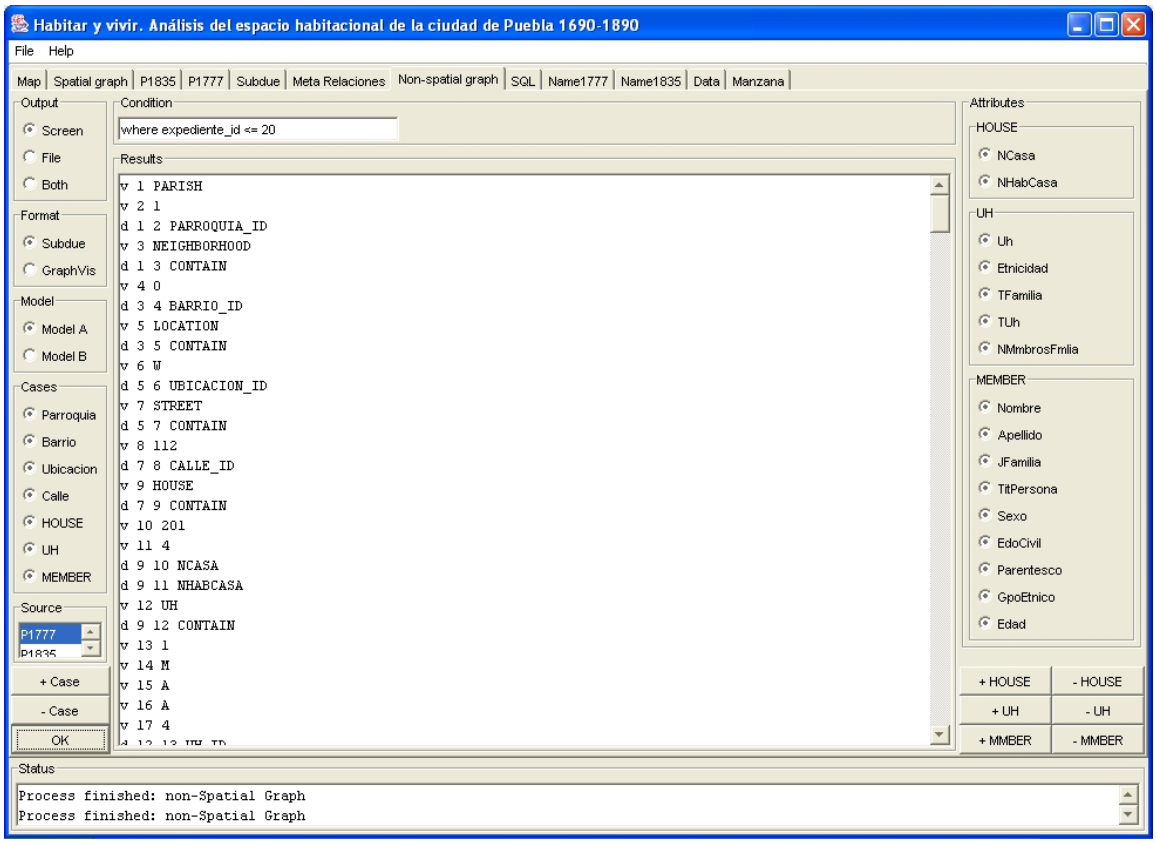

Figure 5.8. The non-spatial graph panel.

Subdue. The *Subdue* panel contains the interface developed for calling the Subdue system (see Figure 5.9). In order to run Subdue, the user must select the corresponding text file (a file containing a graph) and define the parameters that will guide the Subdue's substructure discovery system for finding substructures.

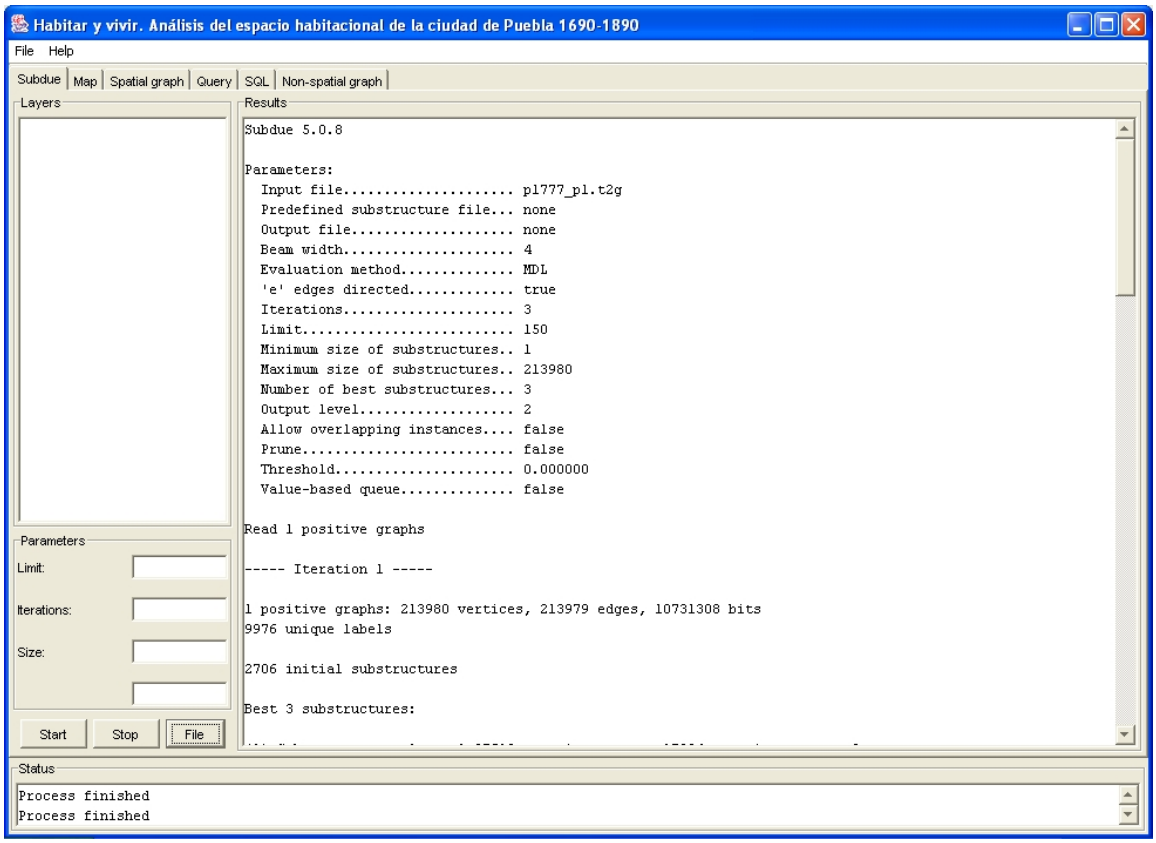

Figure 5.9. The Subdue panel.

Figure 5.10 shows an example of a Subdue's standard output (only for one substructure) for displaying the discovered substructures (i.e., patterns) from an input graph. Since our definition of instances and substructures as graphs (see Chapter 4.1 for details), the Subdue's output is also a graph, in our example, it can be read as follows:

- Substructure value = 1.01706. Represents the value of the substructure in the input graph.
- Pos instances = 3865. This value tells us how many instances of the substructure exist in the input graph.

- Graph (2*v*, 1*e*). Number of vertices ("v") and edges (in our example directed edges "d") in the substructure.
- "v 1 SUB_4". Substructure's first vertex labeled as *SUB_4*.
- "v 2 X". Substructure's second vertex labeled as *X*.
- "d 1 2 ETNICIDAD". A directed edge from vertex 1 to vertex 2 labeled as ETNICIDAD in the substructure.

```
Substructure: value = 1.01706,
pos instances = 3865,
neg instances = 0
Graph(2v,1e):
v 1 SUB_4
v 2 X
d 1 2 ETNICIDAD
```

Figure 5.10. Example of Subdue's standard output.

As we have already commented, Subdue is a system that finds substructures in a hierarchical way, that is, a substructure found in a previous iteration can appear in a new iteration. When this happens, those substructures are represented by a vertex labeled as *SUB_x*, which represents the best substructure discovered at iteration "*x*".

In our example (Figure 5.10), *SUB_4* is itself a substructure defined by 2 vertices and 1 edge where its first vertex is labeled as *SUB_2*. This means that the definition of *SUB_4* is composed by the definition of *SUB_2*. Substructure *SUB_2* is defined by 2 vertices and 1 edge where its first vertex is labeled as *SUB_1*. Again, this means that the definition of *SUB_2* is composed by the definition of the previously discovered substructure *SUB_1*. Finally, *SUB_1* is a substructure defined by 2 vertices and 1 edge.

As we can see, the lecture and interpretation of the substructures discovered by Subdue may be a complicated task. We have implemented a parsing function for reading a text file containing the results generated by Subdue, so we can create the necessary data structures for presenting to the user the same results but using an easier way to read it as we show in Figure 5.11 (the figure presents the example described in Figure 5.10). This layout is created by using the Graphviz system and is saved as a *JPEG* image file but we can save it in any output format supported by Graphviz. The goal is to improve the way the user can read and interpret the results generated by Subdue.

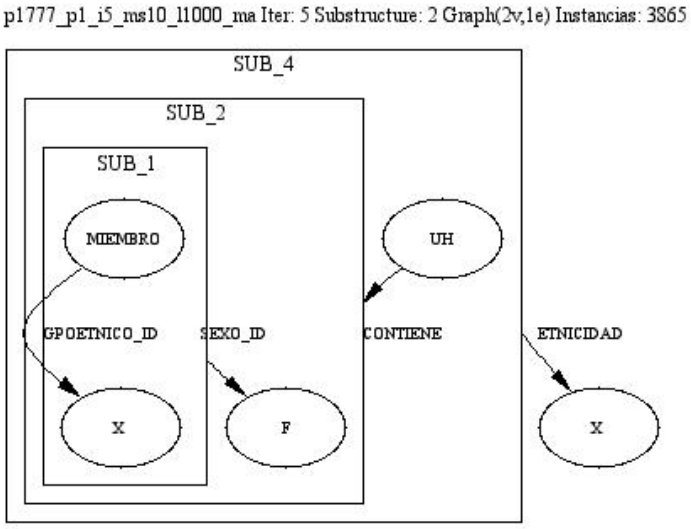

Figure 5.11. Layout for reading the Subdue's discovered substructures.

5.3 Conclusion

As test domain we have designed and built a spatial database to store both a population census from the year of 1777 in Puebla downtown and a map representing the blocks in the

zone. This data is part of a project directed by Dra. Rosalva Loreto López, a researcher in the urban history domain.

We have developed a prototype system implementing our model to represent together spatial data, non-spatial data and the spatial relations among the spatial objects. The prototype allows the user to select the spatial layers to work with, to create spatial and non-spatial queries that will be used to select the spatial objects that will be included in the graph. For each spatial layer the user work with, he has the capability to select the attributes that will be related to the spatial objects in the graph.

We have also implemented a visualization tool which helps us to display in a graphical way (by using the Graphviz system) the hierarchical discovered substructures by Subdue.

In the next chapter we present three cases showing the applicability of our methodology for modeling and mining spatial data mining using a graph-based representation.