Information Access through Conceptual Structures and GIS

Uta Priss, John Old
School of Library and Information Science, Indiana University
Bloomington, IN 47405, upriss@indiana.edu, jold@indiana.edu

Abstract

This paper presents a new technique for information access based on a combination of Geographic Information Systems (GIS) and conceptual structures as modeled in relational concept analysis. It is based on the idea that traditional relational databases restrict their data types to linearly ordered scales. Since spatial data are not linearly ordered, they cannot be modeled directly in relational databases, but require GIS. A Conceptual Query Interface (CQI) separates the query syntax from conditions for the data types and presents the data in a conceptual environment. It therefore allows a common modeling of linearly ordered and spatial data. Previously CQI has been applied to building a graphical interface for faceted thesauri. The main concern of our modeling of GIS and conceptual structures is not geography, but applications that utilize abstract spaces, such as information spaces, knowledge spaces or the World Wide Web. We present a graphical interface that allows access of spatial, hierarchical, and linear data in a common manner. We discuss its features and limits.

Introduction

Traditional database management systems tend to have limitations concerning their usability by untrained or moderately trained people. The most commonly used query language, SQL, is easily applied to simple queries (such as 'Select * from table where attribute=value'), but is highly complicated in cases where the natural language quantifiers 'all' and 'only' are to be modeled, especially if this involves joining several tables. Query by example (QBE) languages, for example the click-and-drag interface of Microsoft Access have similar limitations because only simple queries can be formulated by clicking and dragging. More complicated queries usually require the users to include SQL statements or parts of SQL queries. The situation gets worse if the data are not of nominal or linearly ordered data types. (Examples for nominal data are surnames. Examples for linear orders are the real and the integer numbers. Partial orders, for example tree orders, do not have to be linear, but all linear orders are partial orders. A mathematical lattice is a partial order with additional properties concerning infima and suprema.) The formal concept analysis software TOSCANA (Vogt & Wille, 1995) provides a means of representing partially ordered set (or mathematical lattice) data types. Each database field is scaled into a conceptual scale that is represented as a mathematical lattice. Users do not have to type queries. They simply select the data base fields they are interested in. TOSCANA combines these and allows for the graphical navigation of different scales. Therefore instead of formulating a query as a string and getting the result in the form of a list (as in SQL), users select areas of interest which the system conceptually analyses and graphically represents. Query languages that give results in the form of lists leave users unaware of other items that 'almost' fulfill the query and the general distribution of the items concerning the requested conditions. This is especially frustrating if the result list is empty. TOSCANA on the other hand, represents the results within their conceptual environment. The results are precise, but still open for user interpretation. As an analogy, if users are interested in all cities that are close to New York, SQL-like queries would return a list of city names whereas TOSCANA-like systems would return a map that highlights a circle around New York from which the users can read the city names. Unfortunately, TOSCANA does not have the capability of representing non-partially ordered data types, such as geographical maps. Therefore in this paper we describe a system that is similar to TOSCANA, but incorporates further data types.

Figure 1 demonstrates how our system interacts with existing technology. Database queries are implemented using SQL and Geographic Information Systems (GIS) technology. (For a comprehensive introduction to GIS see Laurini & Thompson (1992).) A conceptual query interface (CQI) allows the translation of SQL/GIS queries into a conceptual representation that is based on formal and relational concept analysis (FCA and RCA). Concept analysis software is used to ensure query consistency and to provide the graphical representations. Users formulate natural language queries in their minds. The system has the drawback that users must learn how to map natural language queries to the conceptual representations and how to interpret the diagrams correctly. Our assumption is that although user training is required the improved query results will justify the increased effort. Usability testing will investigate that. Furthermore, we plan in the future to expand the interface by adding a menu from which users can choose common natural language queries. That will reduce the training needed.

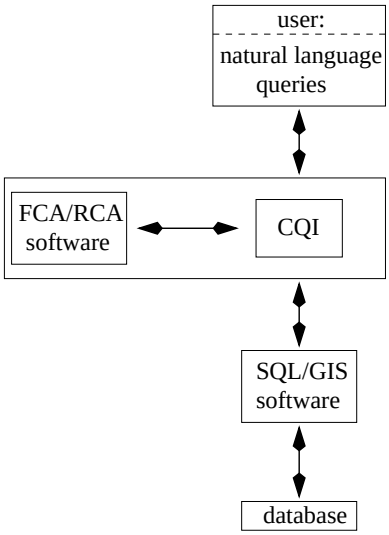

Figure 1: The components of the system

The following SQL statement can be used to explain how CQI improves SQL.

Select $\star$ from table where table.field<"value"

The formula following 'where' is limited to the use of '=', '<', '>', and so on. This corresponds to an implied linear order on the data type of the values. For textual data this is automatically the alphabetical ordering. If the values of a field are modeled to be non-linearly ordered, the ordering has to be stored in a separate table (with fields 'larger_value', 'smaller_value') instead of storing it as part of the data type. If, similarly to the query mentioned above, all rows for which an attribute has a smaller value than a given value are to be selected, the SQL query would involve the join of tables and become considerably more complicated. On the other hand, in CQI it is possible to define a non-linearly ordered data type, therefore a query similar to the one mentioned above could be used. GIS often provide an Extended-SQL interface (such as GEOQL, Spatial SQL or SpaSQL (Adam & Gangopadhyay, 1997)) that does allow the formulation of where statements, such as 'where table.field within 5 miles of location' and 'where table.field intersect location'. This means that spatial data types can be adequately modeled. CQI is more general than SQL/GIS because any data types can in principle be modeled. And, in addition, nested line diagrams (see below) facilitate the visualization of the combination of different data types. On the other hand, each data type requires specific rules. Therefore adding a new data type to the system may require some programming.

Formal and relational concept analysis

Formal concept analysis (Ganter & Wille, 1996) starts with the definition of a *formal context* $\mathcal{K}$ as a triple (G, M, I) consisting of a set of *(formal) objects* (denoted by G), a set of *(formal) attributes* (denoted by M), and a relation I between G and M (i.e. $I \subseteq G \times M$). The relationship is written as gIm or $(g, m) \in I$ and is read as 'the formal object g has the formal attribute m'. A formal context can be represented by a cross table which has a row for each object g, a column for each attribute m and a cross in the row of g and the column of m if gIm. The upper half of Figure 2 shows two examples of formal contexts. They have 'filly', 'mare', and so on as formal objects, and 'female', 'juvenile' ('horse', 'cow', respectively) and so on as formal attributes. In a context (G, M, I) the set of all common attributes of a set $A \subseteq G$ of objects is denoted by $\iota A := \{m \in M \mid gIm \text{ for all } g \in A\}$ and, analogously, the set of all common objects of a set $B \subseteq M$ of attributes is $\varepsilon B := \{g \in G \mid gIm \text{ for all } m \in B\}$. For example, in the left formal context in Figure 2, $\iota\{\text{ram}\} = \{\text{adult, male}\}$ and $\varepsilon\{\text{female}\} = \{\text{filly, mare, cow, ewe}\}$ hold.

	female	juvenile	adult	male
filly	x	x		
mare	x		x	
colt		x		x
stallion			x	x
cow	x		x	
ram			x	x
bull			x	x
ewe	x		x	
foal		x		
calf		x		
lamb		x		

	horse	cow	sheep	animal
filly	x			x
mare	x			x
colt	x			x
stallion	x			x
cow		x		x
ram			x	x
bull		x		x
ewe			x	x
foal	x			x
calf		x		x
lamb			x	x

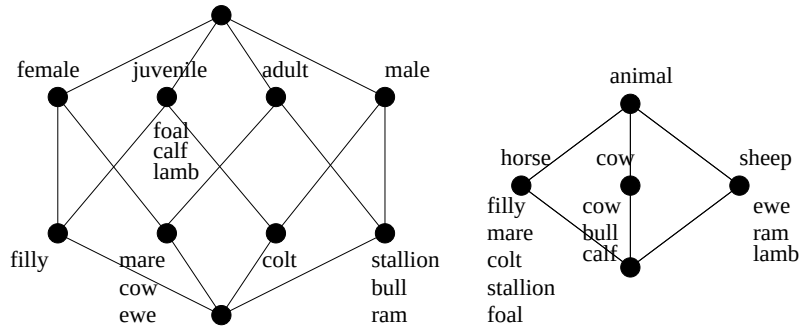

Figure 2: Formal contexts and line diagrams of their concept lattices

A pair (A, B) is said to be a *(formal) concept* of the formal context (G, M, I) if $A \subseteq G$, $B \subseteq M$, $A = \varepsilon B$, and $B = \iota A$. For a concept $c := (A, B)$, A is called the *extent* (denoted by $Ext(c)$) and B is called the *intent* (denoted by $Int(c)$) of the concept. In the right example of Figure 2, ({cow, bull, calf}, {cow, animal}) is a concept, because $\iota\{\text{cow, bull, calf}\} = \{\text{cow, animal}\}$ and $\varepsilon\{\text{cow, animal}\} = \{\text{cow, bull, calf}\}$. The set of all concepts of (G, M, I) is denoted by $\mathcal{B}(G, M, I)$. The most important structure on $\mathcal{B}(G, M, I)$ is given by the subconcept-superconcept relation that is defined as follows: the concept c_1 is a *subconcept* of the concept c_2 (denoted by $c_1 \leq c_2$) if $Ext(c_1) \subseteq Ext(c_2)$, which is equivalent to $Int(c_2) \subseteq Int(c_1)$; c_2 is then a *superconcept* of c_1. For example, ({foal, calf, lamb, filly, colt}, {juvenile}) as a superconcept of ({filly}, {female, juvenile}) has more objects but less attributes than ({filly}, {female, juvenile}). The relation '$\leq$' is a mathematical order relation called *conceptual ordering* on $\mathcal{B}(G, M, I)$ with which the set of all concepts forms a mathematical lattice denoted by $\underline{\mathcal{B}}(G, M, I)$.

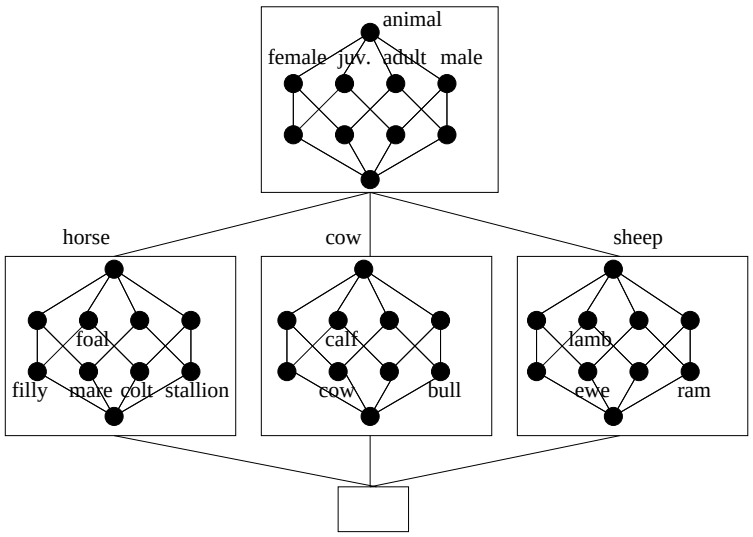

Figure 3: A nested line diagram

Graphically, mathematical lattices can be visualized by line diagrams which represent a concept by a small circle. For each object g the smallest concept to whose extent g belongs is denoted by γg. And for each attribute m the largest concept to whose intent m belongs is denoted by μm. The concepts γg and μm are called *object concept* of g and *attribute concept* of m, respectively. In a line diagram it is not necessary to write the full extent and intent for each concept, instead the name of each object g is written slightly below the circle of γg and the name of each attribute m is written slightly above the circle of μm. The lower half of Figure 2 shows the line diagrams of the concept lattices of the examples. To read the line diagram, the extent of a concept consists of all objects which are retrieved by starting with the concept and then collecting all objects that are written at subconcepts of that concept. Analogously, the intent is retrieved by collecting all attributes that are written at superconcepts of the concept.

Relational concept analysis (Priss, 1996) is an extension of formal concept analysis that considers additional relations on objects or attributes. It furthermore shows that context composition is equivalent to binary matrix multiplication which is itself equivalent to natural joins of database tables. Relational concept analysis is therefore used in the CQI to translate between database relations and formal contexts. The details of which cannot be discussed here.

Nested structures

Nested line diagrams simplify diagrams by identifying lattices that can be used as inner and outer structures of the line diagram. Parallel lines of the outer structure are omitted. In the example in Figure 3, the left lattice of Figure 2 serves as an inner structure, the right lattice of Figure 2 as an outer structure. The concepts in each box are subconcepts of the corresponding concepts in other boxes that are subconcepts in the outer structure. For example, the object concept of 'foal' is a subconcept of the attribute concept of 'juvenile'. Since the bottom box does not contain any objects, it is not completely represented. In database terms (see Figure 4) objects are usually the rows of a table (that means they can be represented by the primary keys). Each database field represents a many-valued attribute. To convert a database table into a formal context, the many-valued attributes are 'scaled' into single valued attributes (Ganter & Wille, 1996). In the example 'gender' and 'age' are scaled into one context and 'species' is scaled into another one. Scaling involves the creation of formal contexts that have the field values as formal objects and new attributes as formal attributes. In the example the field values and the new attributes are identical, except for

'animal' which is added. As mentioned in the introduction, scaling allows the representation of the values of a database field in a non-linear manner because the field values are mapped (via their object concepts γg) into a concept lattice. TOSCANA allows the navigation of nested diagrams. That means users can switch inner and outer scales, zoom into the boxes, and so on.

key	gender	age	species
filly	female	juvenile	horse
mare	female	adult	horse
colt	male	juvenile	horse
stallion	male	adult	horse
cow	female	adult	cow
ram	male	adult	sheep
bull	male	adult	cow
ewe	female	adult	sheep
foal		juvenile	horse
calf		juvenile	cow
lamb		juvenile	sheep

Figure 4: A database table for the data in Figure 2

The 'nested structure' described in the following as a generalization of a 'nested line diagram' makes it possible to replace the inner structure with a non-lattice structure, such as a geographical map from a GIS. To define a nested structure the following sets and mappings are required: The set of elements of the nested structure is denoted by E. Let $\alpha : E \to \underline{\mathcal{B}}(G, M, I)$ be a surjective mapping from the elements of the nested structure into concepts of a concept lattice $\underline{\mathcal{B}}(G, M, I)$. Let $\beta : A \to S$ be a surjective mapping from E into a set S. If the following conditions 1) and 2) are fulfilled then the lattice $\underline{\mathcal{B}}(G, M, I)$ and the set S can be represented as a *nested structure* with $\underline{\mathcal{B}}(G, M, I)$ as the outer structure and S as the inner structure. For $a_i, a_j, a_k, a_l \in A$

$$1)\ \alpha a_i \leq \alpha a_j \Longrightarrow \forall_{a_k \text{ with } \alpha a_j = \alpha a_k} \exists_{a_l \text{ with } \alpha a_i = \alpha a_l} : \beta a_k = \beta a_l$$

$$2)\ \alpha a_i \vee \alpha a_j = \alpha a_k \text{ and } \beta a_i = \beta a_j \Longrightarrow \exists_{a_l \text{ with } \alpha a_k = \alpha a_l} : \beta a_k = \beta a_l$$

The first condition ensures more general structures are contained in more specific structures. The second condition defines the join of two inner structures. The meet of two inner structures is not defined in general, but from the first condition it follows that the union of inner structures can serve as a meet. A nested line diagram is obviously a nested structure. For field-based (or layer-based, see Adam & Gangopadhyay (1997)) GIS applications, the set S is the set of all coordinates of a map. For object-based GIS applications, the set S is the set of geographical objects.

GIS applications

Not all attributes, not even all spatial attributes in a GIS require a spatial representation. For example, in a query:

Select city from table where area > "200 km^2"

'area' is a spatial attribute, but its values are linearly ordered therefore it can be represented as a normal concept lattice. On the other hand, in the query:

Select city from table where distance < "200 km" from location(x,y)

the attribute 'distance' applies to pairs of points in a two dimensional space and cannot easily be represented in a linear order or in a concept lattice. In this case a representation as a map that contains a circle around the 'location(x,y)' and highlights all cities within that circle seems to be more adequate.

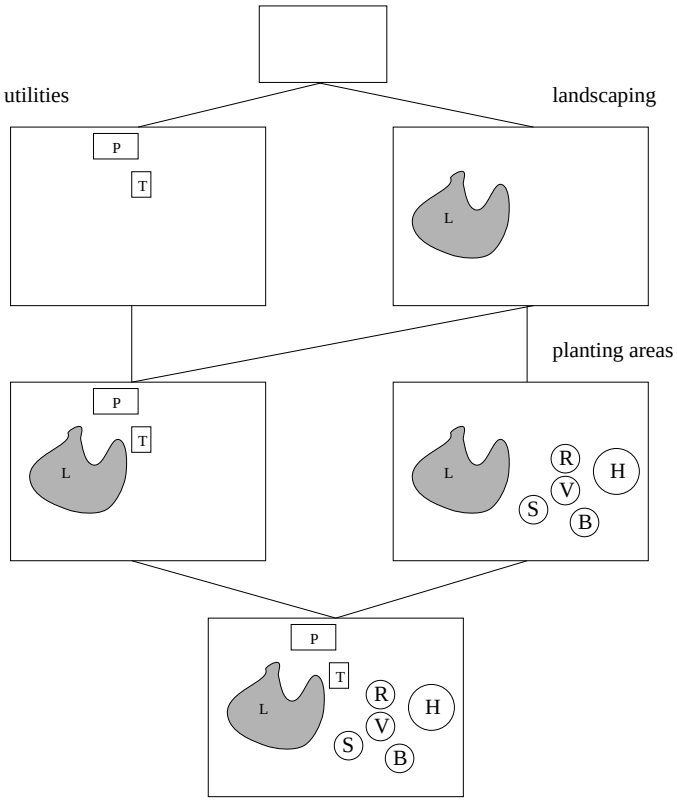

Figure 5: A nested structure for the design of a recreational park

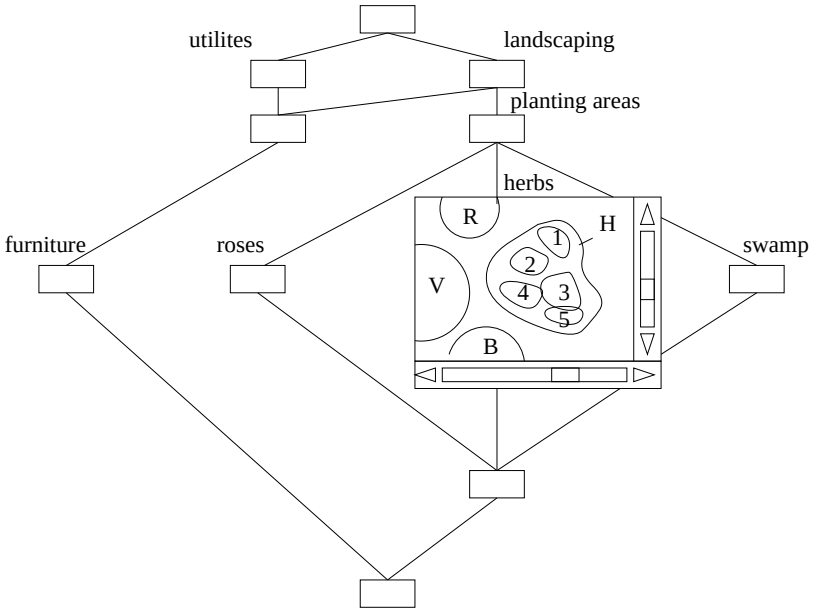

Figure 6: A detail in the design of a recreational park

The example in figures 5 and 6 shows an application in the design of a recreational park. The outer structure represents the conceptual structure of components of the park. On the first level, utilities, such as the parking place 'P' and the tea house 'T' are distinguished from the basic landscaping: lake 'L' versus solid ground. In more detail landscaping involves planting areas (roses 'R', herbs 'H',

vines 'V', swamp 'S', and bromelia 'B'). On the next level (figure 6) the utilities contain furniture and the different plant species are further distinguished. As mentioned before, in a GIS spatial data can be stored layer-based or object-based. In the example of figures 5 and 6 the elements of the nested structure are the objects of a GIS. The mapping to the concepts is given by a database field that indicates the level of detail at which the GIS objects are to be displayed. Adding more details to a map requires increasing the resolution which means that more detailed maps are larger and need scroll bars. To avoid not having enough space on the screen for all maps, in figure 6 the concept nodes are reduced to small empty boxes. Users can click on any of the boxes to view the maps. They can open as many maps as they can position on the screen. As an example, the map that contains the details on the herb planting area is opened. For simplification, the different kinds of herbs are numbered. The node below 'herbs', 'roses', 'swamp' contains all details on all planting areas. The bottom node of the lattice contains all GIS objects of the application. The advantages of this kind of representation are the distinction of conceptual levels of design. For example, the higher levels can be assigned to one design team, the lower levels to another design team, and the different planting area levels to different plant experts. In traditional GIS it is already possible to display different grades of resolution and details, but only in a linear ordering. Our model allows a non-linear ordering that is based on conceptual entities.

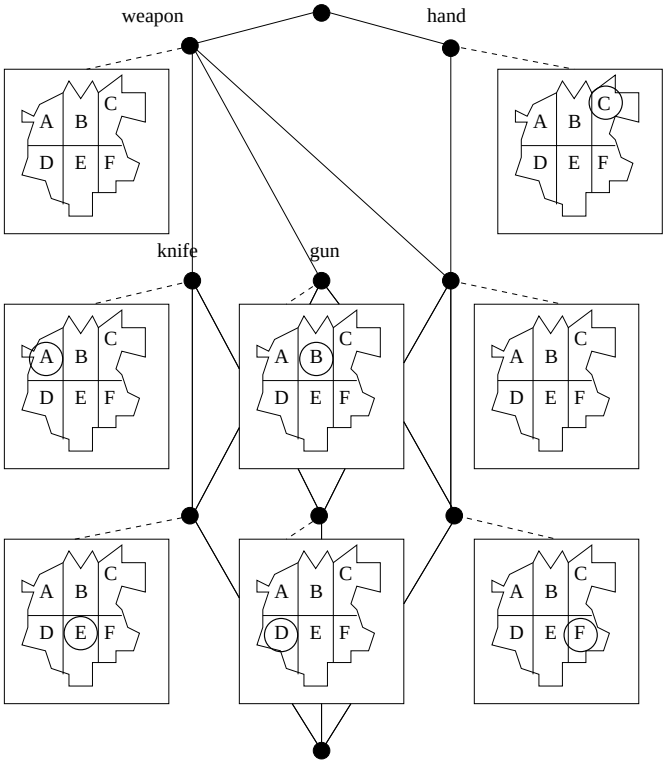

Figure 7: Crime statistics for a town (objects represented only once)

In contrast to the previous example, the example in figures 7 and 8 is not a nested structure, but simply a modified representation of objects. This is indicated by drawing the concept nodes as circles whereas the maps are connected to the circles by dotted lines. In a normal concept lattice objects are simply represented as lists of object names. TOSCANA also allows the representation of the numbers of objects at each node or the numbers of objects in the extent of each node. In SQL terms this corresponds to 'Select count(object) from ...' where the where-condition denotes the intent of a concept. Similarly aggregate functions (sum(object), max(object)), could be represented in TOSCANA. Finally, the SQL 'order by' statement corresponds to an alphabetical or incremental ordering of the list of objects for each concept or for each extent of a concept. CQI further extends

these possibilities for the representation of the objects. Instead of lists of objects, other structures can be displayed that contain and highlight the objects. The example in figures 7 and 8 is based on crime statistics. A police crime report classifies types of crime into certain classes. In our example, the assault data is classified by 'assault by weapon', 'assault by knife', 'assault by gun' and 'assault by hand, fist'. Other classes for assault, such as 'simple assault' and 'aggravated assault' are modeled in a separate scale and not shown in the example. Although the town represented in the map in figures 7 and 8 is fictitious, the crime types are based on a real world example. The town is subdivided into six areas (area 'A' to area 'F'). The relation between the objects 'town area' and the attributes 'crime type' is defined as: 'during a certain time period the type of crime occurred at least once in the area'. Depending on the circumstance this can be modified to 'during a certain day ...', 'during a year ...', 'the type of crime occurred at least 10 times', '90 % of all assaults were of type ...', and so on. In a second scale the different time periods can be combined so that a development over time can be analyzed. Figure 7 shows the objects attached to their object concepts whereas figure 8 shows the full extents of the concepts. For example, in area 'E' both assault by knife and by gun occurred. Assault by knife occurred in area 'A', 'E', and 'D'. To read this from figure 7 all concepts below the attribute concept 'knife' must be considered whereas in figure 8 the result can be read directly from the map at the attribute concept 'knife'. It may seem that these different ways of representing the data are confusing, but in real applications it would not constantly be varied between versions such as figure 7 and 8 and figure 5 and 6. After a testing period one way of representing would be selected as the primary one for the kind of application and end users would only be trained in that one method.

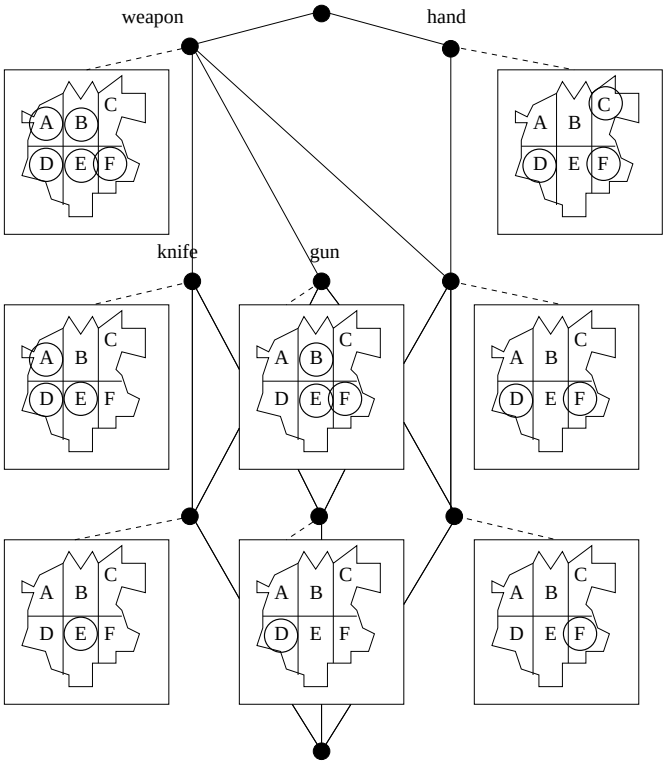

Figure 8: Crime statistics for a town (extent representation)

Conclusion

The main concern of our modeling of GIS and conceptual structures is not geography, but applications that utilize abstract spaces, such as information spaces, knowledge spaces or the World Wide

Web. Since the technology is explained in the previous sections, it may be sufficient to verbally describe such applications. An information space can be represented by taking documents (or other text units) as objects and keywords (or other classificatory structures) as attributes of a concept lattice (Priss, 1997). Additional relations, such as cross references between the documents can be represented as a network and since networks are part of GIS, CQI techniques as described above can be applied. Other similar network applications are, first, the connectivity between web pages. These can be combined with conceptual scales such as

- page connectivity: gateway sites - subpages within the domain - subpages outside the domain
- multimedia content: text - pictures - audio - video
- development over time
- origin of the page: scientific discipline - commercial - private homepage

Second, citation networks (as network maps) can be combined with conceptual scales such as

- temporal dependence
- source: book - journal article - conference proceedings - other
- frequency of citation

Although the conceptual query interface is relatively independent of the domains if the data is given in a relational database, each application requires a considerable amount of data preprocessing: database design, design of conceptual scales, input of data into GIS and relational database, resolution of possible compatibility issues between database, GIS, and CQI, and so on. Real world applications therefore require the full support of a company or institution and should be intended to be used for a certain amount of time otherwise the initial investment may not pay off. In many cases other technologies, such as statistical applications, are already implemented and companies or institutions are often reluctant to switch to new technologies, especially if they are not well known. Therefore even TOSCANA has not been applied as widely as one would expect from its theoretical potential. We are developing a prototype of a CQI for a crime statistics database at the moment that is part of a larger data collection and will be available to the public on the WWW. Hopefully this will lead to further applications.

References

Adam, N., Gangopadhyay, A. (1997). *Database issues in geographic information systems.* Boston: Kluwer Academic Publishers.

Ganter, B., Wille, R. (1996). *Formale Begriffsanalyse: Mathematische Grundlagen.* Berlin, Heidelberg: Springer-Verlag.

Laurini, R., Thompson, D. (1992). *Fundamentals of spatial information systems.* London: Academic Press.

Priss, U. (1996). *Relational concept analysis: semantic structures in dictionaries and lexical databases.* Dissertation, TU Darmstadt.

Priss, U. (1997). A graphical interface for document retrieval based on formal concept analysis. In: E. Santos (Ed.), *Proceedings of the 8th Midwest Artificial Intelligence and Cognitive Science Conference.* AAAI Technical Report CF-97-01, 1997.

Vogt, F., Wille, R. (1995). TOSCANA - A graphical tool for analyzing and exploring data. In: Tamassia & Tollis (Eds.), *Graph Drawing* (pp. 226-233). Heidelberg: Springer-Verlag.