Towards querying and visualization of large spatio-temporal databases

by

Sugam Sharma

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:

Shashi Gadia, Co-major Professor

Udoyara Sunday Tim, Co-major Professor

Johnny Wong

William Gutowski

Simanta Mitra

Iowa State University

Ames, Iowa

2013

DEDICATION

Dedicated to my parents and my wife Ritu

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I express my sincere thanks to all those people who made contributions to this work so I could attain this milestone. Shashi Gadia and Udoyara Sunday Tim, my co-major professors, guided me throughout my graduate study at Iowa State University until now. They sensed my deep interest in spatio-temporal databases and encouraged me to investigate further the same research area. They spent plenty of time in discussions with me to explore many ideas and technical details that eventually led to this work. Shashi K Gadia is with the Department of Computer Science and has very profound understanding of implementation of database systems. Udoyara Sunday Tim belongs to the Department of Agricultural and Biosystems Engineering and is an expert on the Geographical Information System. Their distinguished domain expertise fostered a good understanding and exposed me to such research, where I could not only venture into spatio-temporal database implementation, but visualize the results for analysis purposes. I am very thankful to my other committee members - Johnny Wong, William J. Gutowski, and Simanta Mitra for their great efforts to this work. Their valuable comments and insights on this work have helped improved its quality.

I am thankful to Kevin Kane (Director), and Robin McNeely (GIS Analyst & Lab Manager) in the Geographic Information System, Support, and Research Facility at Iowa State University. They provided useful resources and helped improved this work.

I thank present and past members of the Laboratory of Database Models and Access - Shihe Ma and Daniel Patanroi - for their implementation of the dynamic XML pagination

algorithm and improvement, along with Srikanth Krithivasan, Jia Tao, Valliappan Narayanan, Xinyuan Zhao, and Enruo for their help.

Finally, I thank my parents and my wife, Ritu for their endless love, and support. Over the years, I have sensed deep inside my heart, their sacrifice and I am greatly indebted to them.

ABSTRACT

In any database model, data analysis can be eased by extracting a smaller set of the data of interest, called subset, from the mammoth original dataset. Thus, a subset helps enhance the performance of a system by avoiding the iteration through the huge parental data in further analysis. A subset, its specification, or the formal process for its extraction can be complex. In the database community, subsets are extracted through SQL-like queries and through visualization in the Geographic Information System (GIS) community. Both are iterative processes. An SQL query can be a composition of subqueries. Each subquery can be seen as an iterative step toward the extraction of the desired subset. For this to work, subqueries should result into relations that have the same structure as the relations in a given data model. Although it may not be immediately obvious, the visualization can be iterative too. Each community works in its own compartment. Either one uses subprocesses that are only subqueries or only visual interactions. Mixing these two subprocesses would yield a more powerful expressibility in the hands of users.

Parametric Data Model is well-known for handling multidimensional parametric data, such as spatial, temporal, or spatio-temporal. In the parametric approach, the object is modeled as a single tuple, creating one-to-one correspondence between an object in the real world and a tuple in the database. The parametric approach relies on its own SQL-like, but richer, query language called ParaSQL which mimics the classical SQL. However, it is simpler and avoids self-join operations; hence, enhances performance. In the parametric approach, the attribute values are defined as a function, allowing large

values, also. The execution of a query in the existing prototype of the Parametric Data Model results in data out, as stream in a raw text format that cannot be queried further. This is unlike classical databases, where a subset provides additional strength to a system and the prototype lacks this potential functionality. The real power of ParaSQL lies in the *where* clause, and previous versions of the prototype had a very simple implementation. It is expanded further in this research work to harness its hidden potential. To perform the preliminary investigation, exploratory visual analysis is an important aspect in any spatio-temporal database system. Previous versions of the prototype of Parametric Data Model completely lacked the visualization functionality.

This work ensures the output of a ParaSQL (possibly a subset) will be a relation having the same format as relations in the model rather than plain text. It also attempts to expand the power of the *where* clause, ensuring a clean logic and more generic nature. Some important basic steps are taken to bring a visual in a way that is conducive to the structures in Parametric Data Model. The richness of GIS visualization serves as the foundation for the visual functionality of the Parametric Data Model. The query is executed on the parametric side, while the results are visualized on GIS side. This integration equips the Parametric Data Model with visualization functionality. GIS visualization also offers a click-based selection of a subset and its persistence, which later can be consumed by Parametric Data Model also. This research work establishes a two-way communication between the two communities-Parametric Data Model and GIS-where the output of one can serve as the input for the other and is an attempt to bring them together.

CHAPTER 1. INTRODUCTION

In any database model, data analysis can be eased by extracting a smaller set of the data of interest, a subset, from the mammoth original dataset. Thus, a subset helps enhance the system's performance by avoiding iterations through the huge parental dataset in further analysis. A subset, its specification, or the formal process for its extraction can be complex. In database community subsets are extracted through SQL-like queries and through visualization in Geographical Information System (GIS) community. Both of them are the iterative processes. An SQL query can be a composition of subqueries. Each subquery can be seen as an iterative step toward the extraction of the desired subset. For this to work, subqueries should result in relations that have the same structure as relations in a given data model. Although it may not be immediately obvious, visualization can be iterative too. Possibly a user may require tweaking of visual elements as well as attribute information to fine tune it. Each community works in its own compartment. Either one uses subprocesses that are only subqueries or only visual interactions. However, mixing the two subprocesses could yield more powerful expressibility in the hands of users.

In databases, generally the term dimension refers to time or space. Unlike conventional databases, which help with business data processing, dimensional databases handle the data that have dimensions associated with it, such as – time, space or both. The databases equipped with these dimensions are termed as *temporal*, *spatial* and *spatio-temporal databases* respectively. Although significant research has been achieved with spatio-temporal databases to manage spatially and temporally referenced data, a growing interest from the scientific community still continued in this research area. Today, most

real world applications deal with space and time dimensions, and require efficient data processing systems to handle these dimensions.

The Parametric Data Model (Gadia and Nair, 1993) is one such model that has been actively studied since the mid 1980s for its usability in dimensional data. This model assumes an underlying hypothetical space, called *parametric space*, that can simply be viewed as a set of points. The parametric elements - *spatial*, *temporal* or *spatio-temporal* - are subsets of the parametric space. This allows the Parametric Data Model to manage the heterogeneous dimensions (such as spatial and temporal) uniformly at the abstract level; thus, easing the handling of a variety of dimensions.

In the Parametric Data Model, the attribute values are defined as functions over the parametric space, allowing the model to capture spatial and temporal variability. The domain of an attribute is represented as a parametric element. In the Parametric Data Model, the set of all parametric elements should be closed under set theoretic operations - *union*, *intersection*, and *complementation*, and this is an important requirement (Gadia and Nair, 1993; Gadia and Vaishnav, 1985; Tansel et al., 1993). The involvement of the closure property of the set operations (Gadia, Chopra, and Tim, 1993) reduces the complexity of the parametric structure query language (ParaSQL), a SQL-like query language used in parametric approaches (Gadia and Nair, 1998). In ParaSQL, the set operations - *union*, *intersection*, and *complementation*, can be directly mapped to *or*, *and*, and *not* respectively, in natural languages. In addition to this, the Parametric Data Model captures an object in a single tuple, leading to a one-to-one

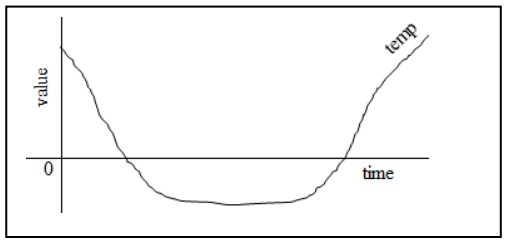

Figure 1.1 Temp function

correspondence between objects in the real world and tuples in a database. This avoids the multi-way joins (especially self-join), invoked in other models very frequently and held responsible for slower system performance. Also, this contributes to reducing the complexity of ParaSQL. In the Parametric Data Model, the concept of domain is very important. Figure 1.1 shows an example of temperature function and assume v is that domain of the function when the temp is below the freezing point, i.e. v= [[$temp \leq 0$]]. Suppose a query, Q, applied to parametric value temp returns subset u when the temp was below the freezing point; then, $u = temp \lceil v$. This means, in the context of a value temp, u is completely characterized by its domain. Thus, the query could concentrate upon the returning domain. To design very good language for a domain is easy. The domain can be generated and computed from visualization and subexpression.

A working prototype (Noh, 2006; Narayanan, 2009) of the Parametric Data Model has already been developed and is validated for an interesting use case, a spatio-temporal NC-94 dataset (NCRA, 2004). It is one of the most complete records of temporal and spatial variables for climate-crop-soil in the North Central Region in the United States. A sample tuple of climatic data is shown in table 1.1.

Table 1.1 NC-94 Climate Dataset Sample

FIPS	StFIP	CoFIPS	Year	Day	Radiant	MaxTemp	MinTemp	Precipitation
19001	19	1	1971	79	17.9	7.61	-6.06	0
• • •								
19169	19	169	2000	364	3.8	-8.72	-13.28	5.84
• • •								
19197	19	197	2000	364	4.8	-10	-19.5	6.35

The present design and implementation of the prototype raises other interesting issues; when a ParaSQL query is executed on a spatio-temporal dataset, such as NC-94, the resultset is a stream of data in a raw text format. The resultant can either be directed to an output console such as DOS (Disk Operating System) for display or dumped to a static text file for later use, dismantling the relational structure in both cases. This may contribute to a wide variety of applications, but falls short from various perspectives such as, the streamed out data cannot be queried further. In any database model, the analysis of the data can be eased by extracting a smaller set of the data of interest, a subset, from the mammoth original dataset.

The subset helps enhance the system's performance by avoiding the iteration through huge parental data in further analysis. A subset, its specification, or the formal process for its extraction can be complex. In the database community, subsets are extracted through an SQL-like query. An SQL query can be a composition of subqueries (a select statement inside another select statement). Each subquery can be seen as an iterative step toward the extraction of the desired subset. For this to work, subqueries should result into relations that have the same structure as relations in a given data model. Unlike the other databases where the subset provides additional strength to a data model, the previous versions of the prototype lacked this potential functionality. However, this research enriches it with subsetting capabilities.

In a database management system, the *where* clause has its own merits in the query execution sequence, since it filters out the unsatisfying tuples from the top level only; thus, avoiding unnecessary further operation and contributing to system performance. The real power of ParaSQL also lies in the *where* clause. In the previous versions of the

prototype, a very simple *where* clause was implemented. To harness its full potential, further expansion is required. In the Parametric Data Model, the *where* clause depends upon the tuples as a whole. A parametric tuple contains tuplets, which is a record that consists of attributes and is atomic data. In this research work, the expansion cleans the logic of the *where* clause and makes it more generic to handle any similar kind of spatio-temporal data in which the geography does not change with time. A general purpose technology is used for spatio-temporal object where geography does not change with time (stationary objects). To load a real dataset into a parametric relation, now a generalized loader is used, which works for any relation of this type including climate, crop, soil and their subsets.

In large spatial, temporal, and spatio-temporal databases, interactive visual analysis always helps when performing an exploratory investigation about the data (MacEachren et al., 1999; Gahegan, 2005). Thus, in any dimensional data model, visualization should be the undetachable concept. Previous versions of the prototype of Parametric Data Model entirely lacked the visualization functionality.

The Geographic Information System is widely used in geospatial visualization and is accepted as one of the most suitable technologies in spatial database research, due to its visualization capability. Most of the applications rely on GIS visualization in order to perform the spatial data analysis. Today, there are a good number of tools available to achieve GIS-related operations and one popular tool is ArcGIS (ESRI, 1990). In this research work, the comprehensive strength of ArcGIS visualization is integrated into the Parametric Data Model to serve as the visualization medium. Therefore, the integration enriches the Parametric Data Model with visualization capabilities. The visual output

published through this integration is non-terminal. It has a wide scope to be processed further for a variety of database operations, such as subsetting, localized not only to ArcGIS, but also can be sent further to the Parametric Data Model as a fertile input, where ParaSQL operations are welcomed. This entire circular process offers two-way communication between the Parametric Data Model and GIS, and helps bring them together (Figure 1.2). Their close proximity may evolve better ways to manage large spatial, temporal, or spatio-temporal databases and produce very interesting results with a better outreach and larger societal impact. The establishment of this two-way communication is not direct. Rather, it is leveraged by two intermediary software tools, developed as part of this research work, called, AutoConViz (Automatic Conversion & Visualization) and SubVizCon (Subset of Visualization & Conversion). AutoConViz facilitates the query results visualization (from Parametric Data Model to GIS); whereas, SubVizCon assists the activities, initiated on GIS side and migrate towards the Parametric Data Model for further processing.

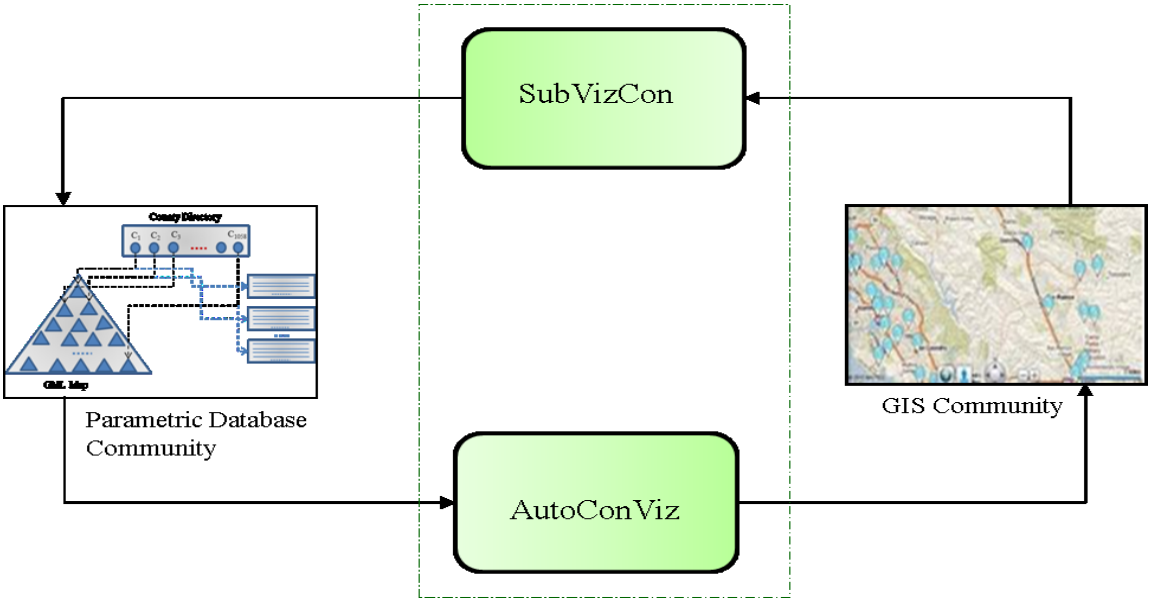

Figure 1.2 Proximity of two communities for mutual benefits

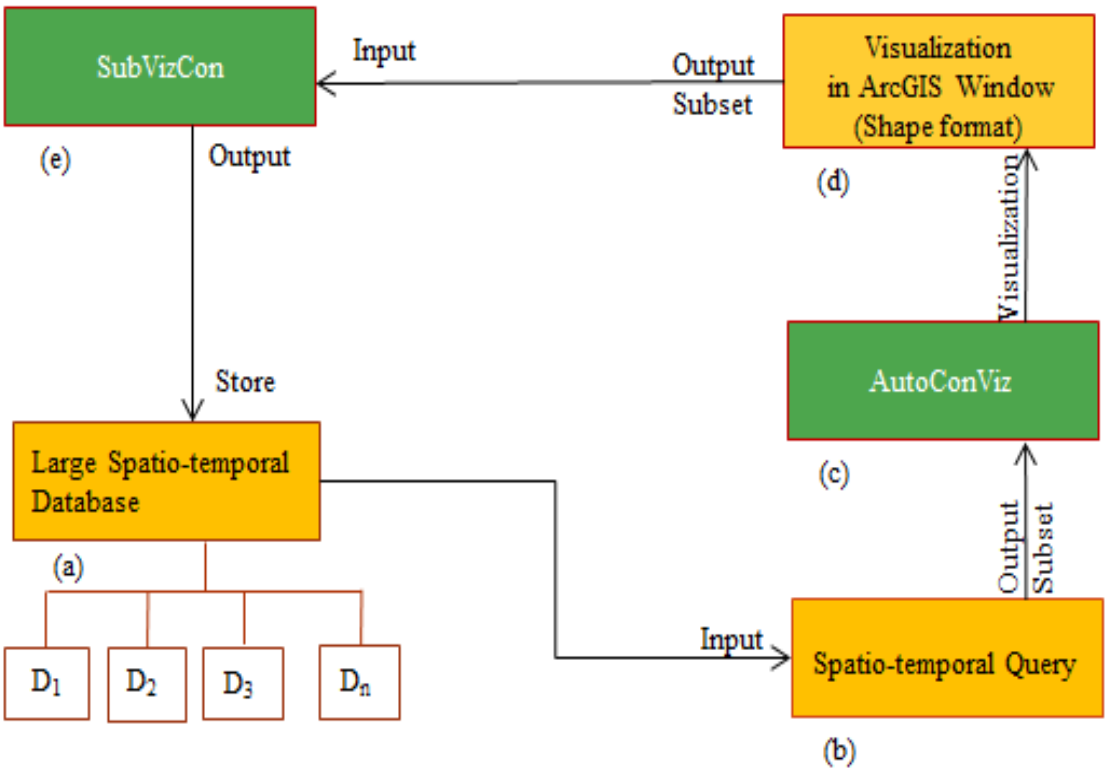

Figure 1.3 Composite framework of the study

The composite framework for this study is shown in figure 1.3. As demonstrated in this framework, to analyze a spatial or spatio-temporal dataset, the query execution is handled at the Parametric Data Model level and the query results visualization is facilitated on the GIS side. With a large database, the analysis of a plethora of data is not advisable from the efficiency point of view. Rather, the interesting subsets of the vast data are extracted to do interesting analysis. The smaller subset facilitates the desired analysis efficiently. It is assumed the database consists of multidimensional datasets. In the figure 1.3, D_1, D_2, D_3, ...Dn represent the different datasets with distinct dimensions, – such as spatial, temporal etc. Spatio-temporal Query (ParaSQL queries) (Noh, 2006) facilitates driving the results from spatial/ spatio-temporal datasets. The resultset produced by the query will be a subset of the original dataset.

The queried output can be used as an input to AutoConViz that converts it into GIS adaptable format and subsequently visualize it automatically. AutoConViz possesses all the basic functionalities required for visualization, such as zooming, panning, selection etc. In addition to this, AutoConViz consists of a navigation button, which on a single click helps render the visualization in ArcGIS (ESRI, 1990). Since ArcGIS offers huge pack of functionalities, therefore more intensive analysis is possible there.

As mentioned previously, ArcGIS is decorated with a large pool of functionalities; *Selection* is one of the core functionalities ArcGIS offers and is of great interest here. This allows a click-based interactive subset extraction of the visualized artifact that can be stored as a new layer (subset of the original map), possessing the same format as the parent artifact and readily available for further use. This means in ArcGIS, the subset of the larger dataset can be extracted without executing any SQL-like query and only click-based extractions suffice the subset requirement. This can be very helpful and interesting to users, who either lack knowledge of how to express the complex queries or want to avoid the extra burden of expressing complex queries in their correct syntax.

Such click-based interactive subset extraction in the Parametric Data Model is not possible. However, the GIS-extracted subset can serve as a fertile input, facilitated by SubVizCon, to the Parametric Data Model to produce more interesting results. In their current state, AutoConViz and SubVizCon are fully developed and validated.

The **research challenges** in this study are consolidated as goals and are addressed in this dissertation as indicated below:

1. Implementation of the concept of subset in the Parametric Data Model
2. Addition of the subquery capability to the Parametric Data Model
3. Expansion of ParaSQL
4. Integration of the visual aid to the Parametric Data Model
5. Visual exploration of large spatial and spatio-temporal datasets

The remainder of this dissertation is organized as follows. Chapter 2 comprises the background and literature review explored for the development of this work. Chapter 3 enriches the dissertation with detailed description of large spatio-temporal databases. As indicated already, a large spatio-temporal dataset NC-94 database is used as a test bed in the dissertation and this chapter describes it in greater detail. Chapter 4 is the detailed description of the Parametric Data Model and also includes the enhancements to the model. Chapter 5 focuses mainly on visualization and describes the need for visualization in large spatio-temporal databases, and the concept of visualization in Parametric Data Model. It also emphasizes on easy conversion and visualization in spatio-temporal datasets followed by the concept of visual subset in GIS. This chapter also discusses how this work brings the two communities - database (Parametric Data Model) and GIS - closer to seamlessly work together, which oftentimes have been seen working in their isolated environments. A few useful case studies are compiled in chapter 6. Chapter 7 concludes the dissertation and sheds some light on future research directions.

CHAPTER 2. BACKGROUND AND LITERATURE REVIEW

In the database world, a data model is a formal process pertaining to a set of notations for data description and a set of operation for data manipulation. A rigorous model promises a fine set of optimized query and analytical methodology to perform on dataset. The spatio-temporal data models are those data models that deal with the data representing the spatial object evolution over time and are intended to unearth the mystery in real world applications. Examples include moving point, moving line, or moving region as data and union, intersection, touch, difference, buffering, trajectory, location at one time, enters, crosses etc. as possible operations (Schneider, 2009). This chapter documents the background and literature related to spatio-temporal data models and spatio-temporal data visualization. For every spatio-temporal model referenced in this chapter, the special emphasis is given on their query language. As mentioned already, in this dissertation one such spatio-temporal data model is taken into intensive consideration for further research and in the dimensional database world, popularly known as the Parametric Data Model. In this chapter, this model is reviewed in greater detail.

2.1 SPATIO-TEMPORAL DATA MODELS

Numerous efficient spatio-temporal data models are proposed in the database literature and in this section, I shed some light on a few of them with special emphasis on their query languages. I start with a few spatio-temporal sample queries (Schneider, 2009) that

are asked in spatio-temporal database world. The sample queries are executed on two sample spatio-temporal databases: *flight* and *weather*.

Example1. *Find all flights from Frankfurt that are longer than 5000 kms*.

```
SELECT id
  FROM flight
WHERE from= 'FRA' and length (trajectory (route)) >5000
```

Example2. *Retrieve any pairs of airplanes, which, during their flight, came closer to each other than 500 meters.*

```
SELECT f.id, g.id
  FROM flight as f, flight as g
WHERE f.id <> g.id and min (distance(f.route, g.route)) < 0.5
```

Example3. *At what time was flight TB691 within a snowstorm with id RS316?*

```
SELECT deftime (intersection (f.route, w.area))
  FROM flight as f, weather as w
WHERE f.id = `TB691' and w.id = `RS316'
```

Example4. *Which are the planes that ran into a hurricane and had to traverse it?*

```
SELECT f.id, w.id
  FROM flight as f, weather as w
WHERE w.kind = `hurricane' and
          f.route Disjoint >> meet >> Inside >> meet >> Disjoint w.area
```

Though, all the sample queries are easy to understand for a naïve user, except the example4 that contains a long temporal sequence- *Disjoint >> meet >> Inside >> meet*

>> *Disjoint*. The symbol '>> is the temporal composition operator. The temporal sequence can be unraveled as follows; In the beginning, a flight on the route is disjoint from the hurricane for some time, touches (meets) the entrant boundary of the hurricane at one point of time and enter inside the hurricane, stays inside for some time and meets the exit boundary again and becomes disjoint again.

Pelekis et al. (2004) also compiled a detailed literature review on spatio-temporal database models in its breadth and depth. Their paper not only reviewed the various spatio-temporal data models existing in the literature, but covered the newly emerged contemporary theories and concepts developed in the last decades.

Vazirgiannis and Wolfson (2001) had taken moving objects database into their research consideration and defined a concise data model alongwith a query predicates, enhanced by a proposed implementation design. In the database, the moving objects in road networks were considered. The network model is nothing but a relation that represents "city blocks" which in-turn are the edges of the network graph. A tuple in the relation consists of a polyline that describes edge geometry. The tuple consists of the information that is very much application-specific. Example includes ranges of street numbers – left and right side of the road or Zip codes etc. The model was borrowed directly from a geographic data providing company. The model was not a generic and is not defined formally, and cited as application-specific. Basically it corresponds to an undirected graph with edges as city road block and node as street crossing. The geometric polylines describes the moving objects. The idea was to compute a network shortest path, subsequently assign traveling times and thus to arrive at a trajectory. In their paper, the

authors also proposed a few ad-hoc extensions - especially the modifiers in the *WHERE* clause - to SQL as the query language, few of them are expressed here.

Query1. *Retrieve each object from moving object (M_O) database for which at every time point between starttime and endtime, the object will reach R (a point on map) within 5 minutes, while travelling on its trajectory.*

```
SELECT id
  FROM M_O
WHERE id WITHIN 5 minutes  FROM R
                           ALONG EXISTING PATH
                           ALWAYS BETWEEN starttime and endtime
```

Query2. *Give the objects that sometime between starttime and endtime are less than 5 minutes from R when they travel along their designated trajectory.*

```
SELECT id
  FROM M_O
WHERE id WITHIN 5 mins  FROM R
                        ALONG EXISTING PATH
                        SOMETIME BETWEEN starttime and endtime
```

In their work, Demiryurek et al. (2012) introduced a real-world data-driven system called TransDec. It primarily deals with transportation related spatio-temporal datasets such as traffic sensor data, trajectory data, and point-of-interest data. The system facilitates the interactive and extensive data querying and supports a variety of spatio-temporal queries and few of them are described here.

Monitoring Queries on Streaming Data. Such kinds of queries are applied to a datasets which are constantly received data streams instead of finite stored data. The queries are continuous rather than one-time and example includes the following.

Query3. *Continuously report the speed and occupancy information from each highway sensor every ten seconds.*

Analysis and Mining Queries on Historical Data. TransDec supports the extraction of useful patterns from the historical data for knowledge discovery. One such query may be expressed as follows.

Query4. *Reports the average speed of a segment (e.g., on I-5 from post mile 293 to 300) within last five Mondays between 8:00am to 8:30am.*

Route Planning Queries. TransDec introduces the time-dependent route planning (TDRP) query. Unlike the existing system in the literature where edge cost is fixed, for the shortest path computation in the network, the edge cost is defined as the function of the time (time-dependent). In real-world, TDRP application may alert the travelers about the best departure time and the legalistic expected travel time for a given path.

Location-Based Queries. The TransDec system consists of the set of location-based queries. These queries are spatial queries that include k nearest-neighbors (k-NN) and range queries also. Unlike conventional queries, the location based queries use geometric data types - lines, point, and polygon - and exploits their spatial relationship. The query results of k-NN (time-dependent) query likely to be updated continuously as the points of

interest (POI) or the query points change continuously. A range query reports all the points of interest enclosed by the user-specified boundaries such as polygon or rectangle etc. The example includes the following.

Query5. *Returns the list of all the buses within two miles radius of the specific (given in the query) center.*

In his research work, Mokbel (2004) attempted to advance the spatio-temporal query processing system by developing a framework for continuous query processing. He introduced two main paradigms that distinguished his framework from the rest contemporaries - 1) Scalability in concurrent continuous query processing, 2) Partial updating in continuous spatio-temporal query results. The scalability is obtained using a shared execution paradigm for continuous spatio-temporal queries. The shared execution enables queries to be indexed similar as that of data. This reduces the evaluation of concurrent continuous spatio-temporal queries to a join between the moving objects and moving queries. The computation of the updates (positive or negative) only in the previously collected answers helps in achieving incremental evaluation. The author considered three types of objects – stationary, moving, and predictive objects. The predictive objects can be distinguished from the moving object in that the latter reports only its current locations whereas the former has the ability to send its velocity vector that helps in future location prediction. The author worked with three classifications of the query - spatio-temporal range-queries (stationary queries), spatio-temporal k-NN (K-Nearest Neighbors) queries (moving queries), and predictive spatio-temporal range-queries (predictive queries).

2.1.1 Parametric Data Model

Since the mid-eighties, the Parametric Data Model has continuously been studied in dimensional - spatial, temporal and spatio-temporal - database communities. The model has the capabilities to handle multi-dimensional data in a uniform way, reducing the complexity of the query at the user level. The Parametric Data Model is one of data models for dimensional data, which is capable of handling heterogeneous dimensions. It was introduced in by Gadia (Gadia and Nair, 1993) to model ordinary, temporal, and spatio-temporal data in a uniform way. In the Parametric Data Model, there is an underlying *parametric space* that is simply viewed as a set of points. The data model defines an attribute as a function over the parametric space. Such a modeling feature makes it possible to capture a real world object in a single tuple - in a database. Therefore, it can sustain one-to-one correspondence between tuples in databases and objects in the real world.

Domains of values in the Parametric Data Model are represented by parametric elements, which are subsets of the parametric space. It is important to note that the set of all parametric elements should be closed under set theoretic operations such as *union*, *intersection*, and *complementation*. Similar to the other spatio-temporal models, available today in the literature, the Parametric Data Model has its own query language to facilitate the data manipulation and popularly known as Parametric Structured Query Language (ParaSQL). Satisfying the closure property for the set operations articulates ParaSQL in more precise term. It naturally minimizes the complexity of ParaSQL by mapping union, intersection, and complementation to *or*, *and*, and *not* in natural languages.

It must be noted, the Parametric Data Model is orthogonal to database paradigms such as relational, object-oriented, and XML. Based upon the Parametric Data Model, the databases, called *parametric databases* can be implemented and depending upon the dimensions such as temporal, spatial, or spatio-temporal, handled by the databases, parametric databases can be seen accordingly as temporal databases, spatial databases, or spatio-temporal databases. In order to model an object in the real world in a single tuple as naturally as possible, *parametric elements* are introduced in the Parametric Data Model. Parametric elements are subset of the parametric space. The representing of parametric elements is left open, but they should satisfy the closure property of union (∪), intersection (∩), and complementation (¬). The parametric elements may be called as *temporal elements* in the temporal context, *spatial elements* in the spatial context, and *spatio-temporal elements* in the spatio-temporal context. It is worth noting that space and time dimensions are intertwined in the spatio-temporal context so that spatio-temporal elements in the Parametric Data Model can be constructed by combining temporal elements and spatial elements. For an illustration of ParaSQL, consider the following three English queries which assume underlying relations are temporal, spatial, and spatio-temporal relations, respectively.

Query6. *Give details about employees while they worked in either R&D or Sales and did not earn more than $60, 000.*

```
SELECT *
  RESTRICTED TO (  [[E.DName = "R&D"]]
                     +     [[E.DName="Sales"]]
                  )  * (~ [[E.Salary > 60000]]  )
FROM Emp E
```

This ParaSQL query retrieves tuples from Emp relation which is a parametric temporal relation. For every tuple, it restricts the tuple to a temporal domain such that an employee worked in R&D or Software department as well as the employee earned less than $60, 000 as a salary.

Query7. *Retrieve complete information about counties whose land acres are greater than 500, 000 and which grow wheat and corn.*

```
SELECT *
  FROM County C
WHERE Area ( [[ C.CName]] ) > 500000 AND C.Crop = "wheat"
                                        AND C.Crop = "corn"
```

In this ParaSQL query, the relation County is a parametric spatial relation which contains crop information with spatial regions such that crops are being cultivated. In *WHERE* clause, the ParaSQL checks if the area of a county is greater than 500, 000 acres. The spatial function Area ([[C.CName]]) returns the area of the domain of a county because [[C.CName]] returns a spatial element. It also checks if the county grows wheat and corn.

2.2 SPATIO-TEMPORAL DATA VISUALIZATION

Effective and informative presentation of spatial or spatio-temporal is a challenge and many research studies focus on that. In this section, I mainly review the previous work on visualization such as the visualization for query construction, visualization for decision making, etc. Sharma et al. (Sharma and Gadia, 2010a; Sharma, Gadia, and Goyal, 2010)

attempted to bridge the gap between database and GIS communities by exploiting the ParaSQL queries on spatio-temporal dataset, NC94 with spatial granularity as county, to calculate the coldness in Iowa and visualize that coldness using shapefile format of spatial data. The colder county was rendered using the darker shade of blue color. This work was further extended by Sharma et al. (Sharma and Gadia, 2010b), where the potential of XML as configurable input was harnessed. An xml configuration file was employed as input in the visualization of coldness and the number of the shades to be rendered was controlled from there. An interesting visualization approach was described in – GeoDec (Shahabi et al. 2010). It offers an immersive environment to visualize geospatial data and helpful in decision making. It relies on the fusion of geospatial data - vector data, satellite imagery, and raster maps - on-the-fly. In spatial databases, visualization sometimes turns out to be very helpful in analyzing and decision making, and Sharma et al. (Sharma, Tim, Smith, and Gadia, 2011) attempted to visualize the spatial dataset – census data – for decision making about SNAP eligibility in Iowa. This helps to USDA and local government to distribute food stamps. GeoVR (Huang et al., 1999) is a system that utilizes hybrid approach that combines database-aware display and display aware database scenario. Huang and others (Huang et al., 2001) demonstrated a hybrid application of Java applet and CGI that advocated a relatively balanced application for both modeling and visualization for data exploration. The extensive modulation was computed on the server side whereas visualization was taken care by client side. Authors (MacEachren et al., 1999) in their work, advocated the utility of visualization in spatiotemporal databases and mentioned that exploratory visual analysis could be witnessed as an effective means for first stage analysis. They explored

geographical visualization (GVis) of spatio-temporal datasets to construct the knowledge discovery in databases (KDD) and design an integrated environment GVis-KDD. Gahegan (Gahegan, 2005) argued that in geographic research community the innovative tools and systems developed over the years were easy neither to integrate nor to use together and vague about what scientific activities they serve. Gahegan further suggested that the entire scientific investigation process should be understood first then the roles of tools and systems they play in it. A process may include various tasks researcher may venture, and broader distinct reasoning, wished to be visualized. Kraak (Kraak, 2006) admitted that the geovisualization enhanced the visual thinking about the complex geospatial patterns, trends, and relationships and advocated the comprehension of the visual support in intelligent decision making. Authors (Takatsuka and Gahegan, 2002) introduced an environment called GeoVISTA studio, which offered a programming-free software development for geocomputation and geovisualization application. GeoVISTA studio hided lots of engineering, metadata and conceptual details from an active user and its programming interface (visual) allowed users to assemble their application using data-flow paradigm.

CHAPTER 3. LARGE SPATIO-TEMPORAL DATABASES

The large spatial-temporal databases are not new to the communities but they are growing at a very rapid pace. Today, the data is being gathered from various resources such as sensors, power meters, social networks sites, etc. and exploited vastly in numerous applications belonging to diversified areas such as geology, ecology, healthcare system, urban planning, agriculture, environmental study, transportation system, aerospace industries, etc. In this chapter, the spatial, temporal, and spatio-temporal databases are discussed in significant detail.

3.1. SPATIAL DATABASES

One of the most prominent examples of spatial data is geospatial objects located in a spatial frame, such as the Earth's surface. Over the period of time, various researchers have been motivated to store and query spatial data, resulting in spatial databases. In the scientific world, various prominent communities have actively researched on spatial data types, data models, and query languages that executed variety of ad-hoc queries. In the database world, eighty percent of modern database systems are constituted by spatial dimension in addition to a rich set of aspatial components. Generally, the aspatial attributes are in textual or numeric format.

The Geographical Information Systems (GIS) is the centralized mechanism that helps to handle the spatial databases. It not only integrates the voluminous spatial data but also

store them. There are two flavors of spatial data, felicitated by GIS -1) raster data, 2) vector data. The raster data is classified as satellite and aerial digital imagery, whereas the vector data may be represented by point, lines, and polygon. The sources of spatial data generation are not limited, but a few of them are: specialized sensors, satellites cameras or cameras mounted on aircraft – capturing spatial regions into digital images, Global Positioning System (GPS) - producing geospatial coordinated for objects in real world, modern mobile devices equipped with GPS such as smartphone. The spatial information is being disseminated very rapidly and the agencies, interested in spatial data business are collecting the geospatial data in increasing number. Generally, the spatial database records represent the object of real world and 'a grocery store in a particular city' or a planned or unplanned event 'scientific conference in a hotel' can be considered as few examples. Thus, in spatial databases, the geospatial component is a must, and alongwith the spatial queries it differs them from non-spatial databases. The geospatial component is the geo-location information and considered as one of the attributes that uniquely identifies a location geographically. In addition to that, in the spatial databases, there is rich set of aspatial (non-spatial) attributes consisting of textual or numeric format. In the above example, for a grocery store, the non-spatial (aspatial) attributes may be its description, type of grocery it hosts, year of establishment, name of the owner, etc. The grocery store will have its address (e.g. street name, zip code etc.) and this address is its location and in database this represents its spatial attribute. In spatial databases, the analyses become interesting, when aspatial (non-spatial) attributes are analyzed under the restricted spatial environment. This spatial restriction can narrow down the analysis up to the point of interest, which in the absence becomes murkier and hinders the clear

exploration. The spatial queries (query by location) consist of functions that are responsible for spatial restriction. The spatial queries have spatial functions that facilitate the desired spatial calculation. A few of them are *distance*, *touches*, *inside*, *outside*, *area*, *buffer*, *touch*, etc. Let us consider a scenario as an example. There exists a spatial database, called *Counties*. The database has the spatial attributes as the FIPS code for counties in the state of Iowa and aspatial attributes as demographic related parameters such as total population, total female population, total male population, number of Blacks, Whites, Hispanics, and Asians, income ratio, etc. in a particular year (say 1980). In the dataset, the spatial component has the granularity as county. The simple spatial queries, one may ask on this dataset are as follows.

Query 3.1. *Find all the neighbors of Story county in Iowa.*

In order to return the results of this query, the spatial system use a function, called *touch*. This function takes two polygons as input and measure whether their boundaries touch each other. The sample output of the query looks similar to the one, shown in figure 3.1.

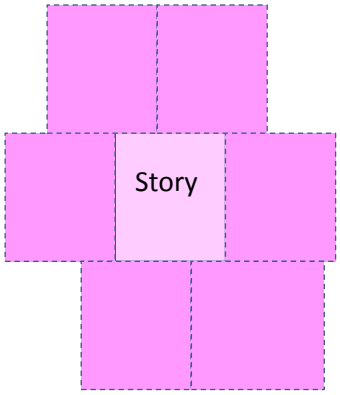

Figure 3.1 Neighbors of Story county

Query 3.2. *Find all the neighbors of Story county in Iowa in which the total population is less than x (where x is any constant number).*

This query is a bit complex than the query 3.1 and consists of two steps. In the first step, it retrieves all the neighbors of Story county. In the next step, it further restricts the outcome in the first steps and out of all the neighbor counties it allows only those as resultant that satisfy the given population criterion.

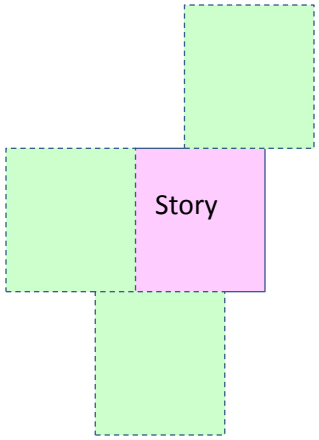

Figure 3.2 Neighbors (population < x) of Story county

As shown in sample output in figure 3.2, the population criterion filtered out three counties from the pool of Story's neighbors.

Query 3.3. *Find those counties in Iowa which are eligible for Supplemental Nutrition Assistant Program (SNAP). The qualified county needs to be the neighbor of Story county.*

This query is more complex than the query in figure 3.2. This requires a bit of mathematical calculations on the income data of the households in order to evaluate SNAP eligibility of a county. The selection of neighbors of the Story county is done by

executing query by location (pure spatial query), whereas only the SNAP eligible neighbors are selected by executing the query by attribute (aspatial query). The sample outcome is shown in figure 3.3 that demonstrates, there is one neighbor of Story county, where the socioeconomic status of the population is above poverty threshold.

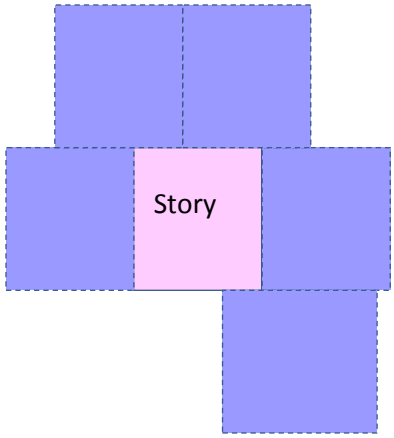

Figure 3.3 SNAP qualified neighbors of Story county

Oftentimes, the spatial databases are strongly recommended for optimization to gain better performance in order to perform even the basic operations such as information retrieval, data analysis and processing, and data visualization. The structured or semi-structured spatial data is required to be suitable for GIS database schemas, before any spatial query is executed to explore its insights.

The spatial database community focuses more on developing practical methodologies to selectively represent and process geospatial data. The most important issue in object-based models of spatial information is the choice of a basic set of spatial data types required to model common shapes on maps. The spatial data is based on the Geometry Object Model in the simple feature specification for SQL (OGC, 1999). In the model,

class *Geometry* serves as the base class and many subclasses such as *Point*, *Curve (Line)* and *Surface (Polygon)* are extended from the base class (Shekhar and Chawla, 2003).

3.2. TEMPORAL DATABASES

In a simple definition, temporal databases are defined as the databases with tightly integrated time aspect. The temporal databases are capable of storing evolution of data, thereby allowing users to examine complete object histories (Bhargava, 1989). In temporal databases, a time-period is always attached to the data that helps in expressing when the data was valid and when it was stored in the database. Unlike conventional databases, where the snapshot data is stored, the temporal databases are capable to store the complete history of the data. Based upon the temporal aspect, the history of the object can be defined in two domains - real world and database world and are represented in the form of *valid-time* and *transaction-time*. In the temporal database literature, these temporal aspects are defined as follows.

- Valid-time - The time period during which the fact about an object is true with respect to the real world.
- Transaction-time – The time period during which a fact about an object is true with respect to the database i.e. the duration for which the object is stored in the database.

Thus, to distinguish the temporal databases from the conventional databases, the very first initiative is to timestamp the data. There are three types of timestamps mainly

defined the in the database literature – *instants*, *intervals*, and *temporal elements*. In this dissertation, I describe them with the help of the examples of temporal data. Table 3.1 is a *Department* table that consists of department name (DName), manager name (MName), start time and end time. A reader can interpret it as "in a department 'x', from time t1 to time t2, person named as 'y' was the manager".

Table 3.1 Department

DName	MName	Start	End
Toys	John	11	14
Toys	Leu	45	47
Clothing	Tom	41	47
Clothing	Inga	71	NOW
Shoes	John	45	60

The timestamp *instant* represents the state of the data at a particular instant (past, present or future) of time and the start and end times for a fact are same. In the above table 3.1, the status of the data at an instant '46' will look like as table 3.2.

Table 3.2 Department (at instant 46)

DName	MName	Start	End
Toys	Leu	46	46
Clothing	Tom	46	46
Shoes	John	46	46

Table 3.1 is the representation of *time-interval* timestamp. In the 'Toys' department, 'John' is manager for time-interval [11, 14] and Leu for [45, 47] and this completes the history of 'Toys' department. The only difference between the time-interval and temporal element is that the latter is defined as a finite union of intervals.

Table 3.3 is the representation for the third type of timestamp called *temporal element*. As mentioned earlier, temporal element falls under closure property.

Table 3.3 Department (temporal element)

DName	MName
[11,49] Toys	[11,44] John [45, 49] Leu
[41,47] ∪ [71,Now] Clothing	[41,47] Tom [71,NOW] Inga
[45,60] Shoes	[45,60] John

It is interesting to note, how the temporal element captures the complete history of 'Clothing' department: [41, 47] ∪ [71, Now] and union operator makes it possible.

The applications of the temporal databases include financial, record-keeping, and scientific applications (Jensen and Snodgrass, 1999). Temporal databases are one of active research areas in the database community and tremendous research work has been done by many researchers. There are many different types of temporal data models and they have their own merits in their specific applications. Based on timestamps, temporal

data models are termed *point-based models*, *interval-based models*, and *temporal element-based models*.

It is important to emphasize that only temporal elements are legitimate domains of objects and events in the real world. However, a temporal element cannot be represented using a fixed length because it is defined as a finite union of intervals. Therefore, intervals or instants are used to timestamp fragments of objects that are stored in multiple fixed-length tuples. Typically such timestamps are attached at tuple level rather than value level. In general, temporal data models are introduced with their query languages. Among many temporal query languages, SQL/TP (Toman, 1997) and SQL*T* (Chen et al, 2003) use point-based data models; TSQL2 (Snodgrass, 1995) and IXSQL (Lorentzos and Mitsopoulos, 1997) use interval-based data models; and ParaSQL (Gadia and Nair, 1993) and NRTC (Tansel and Tin, 1997) use temporal element-based data models.

To offer better understanding to a reader about the temporal database queries, let us consider an example of temporal element based database query (ParaSQL).

Query 3.4. *Retrieve all the managers from the Department relation (in table 3.3) who were manager atleast for the duration [11, 50].*

```
SELECT x.MName
FROM Department
WHERE [[x.MName]] ⊇ [11, 50]
```

Figure 3.4 Sample ParaSQL query

MName
[11, 60] John

Figure 3.5 Output of Query 3.4

As per the syntax of ParaSQL, the query can be expressed as shown in the figure 3.4 and retrieves the relation shown in figure 3.5 when executed on the relation - Department – in table 3.3.

3.3. SPATIO-TEMPORAL DATABASES

The spatio-temporal databases are those databases that store such a dataset which has space and time as inseparable dimensions along with other attributes. They are considered as the further extension of already introduced spatial databases that adopt the concepts of spatial, temporal and spatio-temporal data. They are mainly concern with the spatial and temporal aspect of the data and exclusively deal with variations in spatial patterns with time such as location of the moving objects (Chen, 2001; Güting and Schneider, 2005). In short, spatio-temporal databases are the one that manage the spatial data whose geometry changes over time. The objects changes instead of being static. At any given instant, the spatio-temporal databases simply are conventional spatial database.

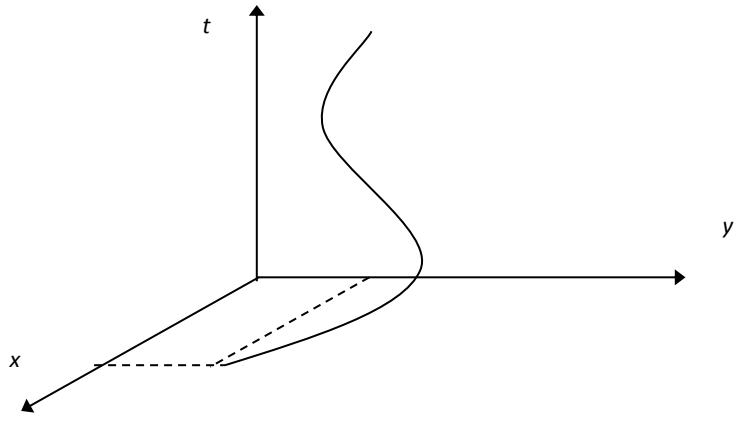

Figure 3.6 A Moving Point

Today, most real world objects and applications have some association with time and space and common examples include - Global Positioning Systems (GPS) capturing the location of a moving object with various point of time, mobile phone users within mobile networks, ad-hoc wireless communication networks having shorter lifespan within a spatial region, and environmental monitoring systems – climate or land cover data, database of epidemic disease, and transportation system are some examples. Because many applications require an ability of managing spatio-temporal data, there is a growing attention to spatio-temporal databases (Abraham and Roddick, 1999). Figures 3.6 and 3.7 show two examples of spatio-temporal objects (Jiyong, 2005). In the first figure, the object is a moving point.

The space is a two dimensional space represented by *x* and *y* coordinates and the time is the third dimension. Thus at any instant of time, a point *P* can be described as P (*x*, *y*, *t*).

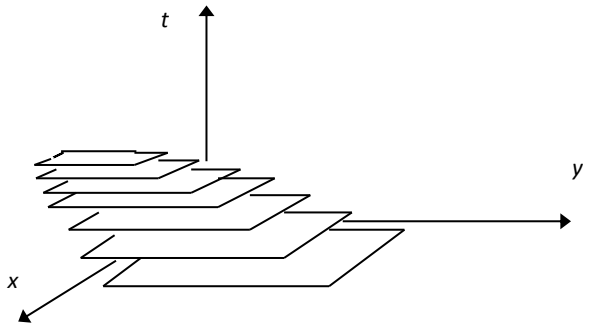

Figure 3.7 A Moving and Shrinking Region

In the second example, the object is a two dimensional region that is moving in x, y plane but shrinking with the time. In this example too, time is represented as the third dimension along z-axis. At any point of time, the region, *R* can be represented as R (*x1,*

y1, l, b, t): *l*, *b* are the length and breadth of the region and *x1*, *y1* are the starting coordinates.

Let us consider another interesting example (George, 2005) of on spatio-temporal data that may help better understanding.

Example1. This example is described by three scenarios, given below and are pictorially depicted in the figure 3.8 (a-c).

Scenario 1 (year 1990). *There are two big land parcels L1 and L2 and have one border common. Parcel L1 has its soil type as "clay" whereas L2 has soil type as "forest". There is a canal C that flows through L1 and L2.*

Scenario2 (year 2000). *L1 was divided into L1 and L11. Now L2 has soil type as "high density forest" and L11 has soil type as "sparse forest" but there is no change in the flows of canal C.*

Scenario3 (year 2010). *The canal shifts its flow direction a bit and becomes C1.*

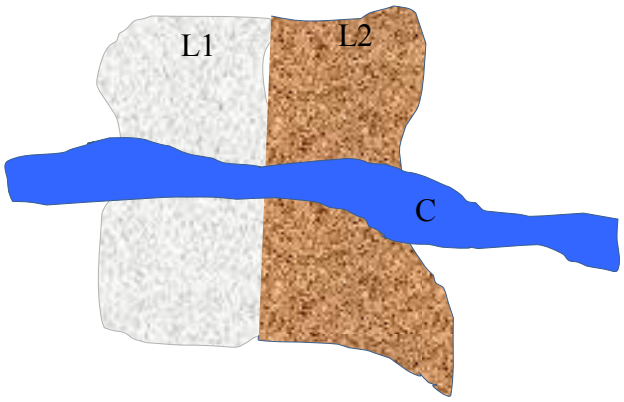

Figure 3.8 a. Scenario (t1:1990)

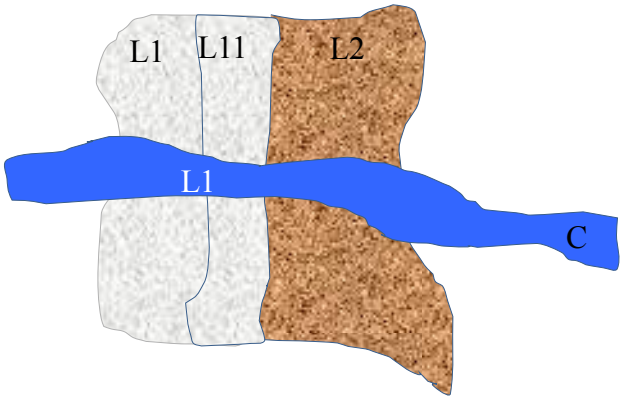

Figure 3.8 b. Scenario (t2:2000)

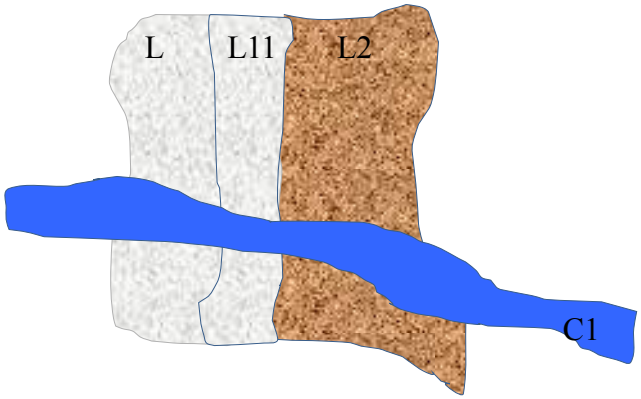

Figure 3.8 c. Scenario (t2:2010)

In order to deal with spatio-temporal database, good database models are required, enriched with efficient query language. However, today, there exist very few prototypes of complete systems, and very less products that provide effective support to the applications, tracking changes to spatial and non-spatial (or ordinary data) over time. Because the design and implementation of a complete spatio-temporal database is a

challenging undertaking, involving extensions to all aspects of a non-spatiotemporal architecture such as data model, query language, query optimizer, query evaluator, programming environment, storage manager, and indexes (Griffiths et al., 2001). In the past, the research in spatial and temporal data models and database systems has mostly been done independently. The spatial database research has focused on modeling and querying geometries associated with objects, while the temporal databases have focused on modeling and querying temporally evolving data. Nevertheless, many researchers have tried to combine the two areas because they all are dealing with dimensions and are closely related (Erwig, 1999). Since temporal and spatial data models have been intensively researched in the temporal and the spatial database communities, one may consider approaches to combine temporal and spatial data models to build spatio-temporal data. Spatio-temporal database models have proposed a variety of queries to analyze the spatio-temporal data. In this chapter, I summarize a few sample queries that may be asked on the spatio-temporal database.

Query 3.5 (Selection queries). In a spatio-temporal database of wireless network, in a given region *Q*, calculate the total number of mobile users at a time *t* for whole year *y*.

Query 3.6 (Nearest Neighbor -NN- queries). In a given county *C*, find the food corner nearest to a given location *l* in the year *y*.

Query 3.7 (Aggregate queries). In a spatio-temporal database of wireless network, in a given region *Q* calculate any instant *t* of time which reported the maximum number of mobile users over the whole year *y*.

NC-94 Spatio-Temporal Dataset. In this section, I discuss one of the most complete spatio-temporal dataset, NC-94. This is the dataset, which is used as the test bed in this dissertation to validate the prototype. The North Central Region in the United States is one of the most cultivated areas for row crop production. It comprises many states and Iowa is one of them, also known as the "Food Capital of the World." Over more than 50 years, North Central Regional Association of Agricultural Experiment Station (NCRA) in the United States is verifying, developing and validating agricultural databases. NCRA has collected a good number of useful agricultural datasets and NC-94 is one of them and is considered as one of the most important, internally consistent dataset. It is used intensively to facilitate crop and risk analysis, pest management and forecasting. As far as outreach is concerned, it is available to the public through the internet and can answer very interesting questions related to agriculture and thus helps in decision making. For scientific purpose, many scientific data formats of NC-94 are used with software packages to store and process it for environmental development. Despite the advantages of the scientific data formats, it is not easy for the public to directly access the rich dataset stored in scientific data formats, because they are not database management systems which directly support ad-hoc queries.

NC-94 is a compilation of 30 years Climate-Crop-Soil data, focused on producing a continuous high quality county-level data of commonly measured quantities of air

temperature (maximum and minimum), precipitation, crop yield and soils data. It is an important resource in the agricultural community and analyzing the dataset would yield invaluable understanding for scientists, public, farmers, planners, and policy makers to improve crop practices and yields, undertake scientific studies, and develop policy. In their research work (Gadia, Gutowski, Al-Kaisi, Taylor, and Herzmann, 2004), the authors mention that IEM (Iowa Environmental Mesonet) has collected many important datasets of evolving atmospheric, soil and hydrologic fields for analysis and for dissemination. NC-94 consists of both the geospatial and aspatial (non-spatial) attributes. Primarily, the geospatial attribute is available in GML format, which consist of the boundary coordinates for all the counties in the North Central Region. In their published work (Sripada et al., 2004), authors present a detail study on Geography Markup Language (GML), and issues that arise from using GML for spatial databases and solutions that have been proposed. The geospatial attribute is also available in shape file format in case needed. Aspatial (non-spatial) data are available in excel sheet for every county in the whole North Central Region and is separated in three classes – Climate, Crop, and Soil. Each class has different set of attributes, but for each of them, the spatial geometry is same and stationary. The Climatic dataset has the following attributes- state FIPS (Federal Information Processing Standards), county FIPS, year, day, maximum temperature, minimum temperature, humidity, precipitation, etc. The temporal granularity for the climatic data is day, which means, the climatic data has been collected on the daily basis for every county in the entire North Central Region. The Crop dataset is collected on annual basis, thus, the temporal granularity for this dataset is year. The other attributes for this dataset are – state FIPS, county FIPS, year, commodity, yield, yield unit,

production, production unit, etc. The soil characteristics do not change so often, so the temporal granularity for this dataset is 30 years. The other non-spatial attributes for Soil dataset are – plasticity, bulk density, field capacity, porosity, available H_2O etc. This data is collected at various depth points – below 25, 50, 100, and 250 cm.

Big Data. The fast-paced growth of the modern data- spatial, temporal or spatio-temporal- raises the doubts on the computing capacity of the existing database models. In last few years, due to the internet sites, the sources of data collection have been increased. The data is bombarded from many potential sources such as sensors - gathering climatic information, power meter readings, social media sites, videos and digital pictures, traffic data and many more. These sources are the responsible for rapid growth of voluminous data - about 2.5 quintillion bytes every day (IBM, 2012). Because of the increasing of the data with such a fast velocity, the size of data is a centrally moving target today. In 2012, in a single dataset, the size of the data likely to increase from few dozen terabytes to many petabytes and about 90% of the whole data in world today is produced in last two years only (IBM, 2012). Not only from storage capacity, but also from processing aspects, the existing database models fall short to handle them. This gives rise to a new definition of data in the database base system, called “Big Data”. The concept of big data is relatively new and has large scope of further research. This chapter highlights some important aspects about the big data. Primarily, the big data is described to have four concerns: volume of the data, velocity of the data, variety of the data and veracity of the data. All of the four concerns are briefly described here.

Volume of the data. Today, the industries are inundated with voluminous data of all types. The data is growing easily from terabytes to even petabytes. Tweets from the social network sites alone are responsible for up to 12 terabytes of data every day. On annual basis the power meters contribute to a huge amount of data – 350 billion readings.

Velocity of the data. The data, getting collected at the enterprises is at a very fast pace. Sometimes even the delay of a minute may cause the discrepancy in the analyzed output. For especially time-sensitive systems such as fraud detection, the big data must be analyzed as it comes into the enterprises to obtain the best results such as scanning though the 5 million trade transactions, produced every day, to capture the potential frauds.

Varity of the data. As per the description about big data, it can be of any type regardless of it structured or unstructured nature. The data type majorly includes – text, audio and video data, sensor data, log files etc. The reason enterprises like to work with big data is the finding of new insights when analyzing all these data types together. The examples include – monitoring live video feeds from surveillance cameras in order to narrow down to the point of interest; to analyze the bulk data growth in the videos, documents and images to enhance the customer satisfaction.

Veracity of the data. In the business world, the leaders do not completely trust on the information extracted from big data. Though, they exploit this information to obtain best decision making. If someone does not trust on the input, how she/ he can trust on the

output and changes that one will act of such outputs are next to negligible. As the big data is growing in size and variety every day, instating the confidence in big data poses greater challenges.

The complete description about big data is beyond the scope of the dissertation. The readers interested to have complete knowledge about big data are encouraged to look for other resources about big data.

CHAPTER 4. PARAMETRIC DATA MODEL

4.1 OVERVIEW

The Parametric Data Model has been studied since the mid-1980s in the literature. It is capable of handling the multi-dimensional data in a uniform way. In this chapter, an overview of the Parametric Data Model for dimensional data alongwith its query language - ParaSQL is provided. The new challenges, taken up in this research work to further enhance the prototype of the model are also discussed in significant details.

4.1.1 Introduction

The Parametric Data Model is one of data models for dimensional data that has the capability of handling heterogeneous dimensions. It was introduced by Gadia (Gadia and Nair, 1993) to model ordinary, temporal, and spatio-temporal data in a uniform way. One of many potential features of the Parametric Data Model is scalability as it can be extended to absorb another dimension, whereas the most of other data models focus only on one particular form of dimensional data.

In the Parametric Data Model, there is an underlying *parametric space* that is simply imagined as a set of points. The data model defines an attribute as a function over the parametric space that makes it possible to capture an object in the real world in a single tuple in a database. Therefore, it can sustain one-to-one correspondence between tuples and objects in the real world.

Domains of values in the Parametric Data Model are represented by parametric elements, which are the subsets of the parametric space. It is important to note that the set of all parametric elements should be closed under set theoretic operations such as *union*, *intersection*, and *complementation*. Satisfying the closure property for the set operations articulates ParaSQL in more precise term. It naturally minimizes the complexity of ParaSQL by mapping union, intersection, and complementation to *or*, *and*, and *not* in natural languages. One must note that the Parametric Data Model is orthogonal to database paradigms such as relational, objet-oriented, and XML. Based on the Parametric Data Model, one can implement databases called *parametric databases*. Parametric databases can be seen as temporal databases, spatial databases, or spatio-temporal databases depending on dimensions - time, space or both - handled by the databases.

4.1.2 Parametric Elements

In order to model an object in the real world in a single tuple as naturally as possible, *parametric elements* are introduced in Parametric Data Model. Parametric elements are subset of the parametric space. Representing parametric elements is left open, but they should satisfy the closure property of union ($\cup$), intersection ($\cap$), and complementation ($\neg$). One may call parametric elements, *temporal elements* in the temporal context, *spatial elements* in the spatial context, and *spatio-temporal elements* in the spatio-temporal context. It is worth noting, the space and time dimensions are intertwined in the spatio-temporal context, so that the spatio-temporal elements in Parametric Data Model can be constructed by combining temporal elements with spatial elements.

A. Temporal Elements

Time intervals express their inadequacy in modeling the history of an entire object in a single tuple, and they require the query languages that are too arduous for expressing in natural languages (Gadia and Nair, 1993). To derive the timestamps, closed under the set theoretic operations - union, intersection and complementation - the notion of *temporal elements* is conceived in Parametric Data Model (Gadia and Vaishnav, 1985; Gadia and Nair, 1993; Tansel et al., 1993). Under its conceptualized ideas, the Parametric Data Model considers that there exists a universe of time, contains an interval [0, NOW] of instants with a linear order on it. In this interval, NOW represents the current instant of the time. In the Parametric Data Model, a temporal element is described as a finite union of various time intervals. This signifies that an interval is an obvious temporal element and an instant *t* can be defined with the interval [*t, t*], respecting the structure of temporal element. A few examples of temporal elements include [12, 70] and [0, 12] ∪ [40, 60]. It is worth noticing once again that the set of all temporal elements should respect the closure property i.e. closed under union, intersection and complementation.

B. Spatial Elements

In spatial database context, the Parametric Data Model considers an underlying universal region R, but a user thinks of it as a set of points. Let us consider a set of subsets of R, and define it as *REG and* it is closed under union, intersection, and complementation. A *spatial element* is an element of *REG*. The constitution of R is not based upon any special

assumptions. The region R can be considered as - an *n*-dimensional Euclidean space, surface of a sphere, portion of a plane, a curve etc. The regions in *REG* should respect a potential hypothesis i.e. the regions should have some reasonable description about them (Gadia and Chopra, 1993).

C. Spatio-Temporal Elements

As per the description of spatial and temporal universe – defined above, their convolution may help in obtaining the spatio-temporal universe R x T = R x [0, NOW]. In both the cases – space and time, our interests lie in those spatial and temporal elements that obey the closure properties i.e. they should be closed under union, intersection, and complementation that are needed for seamless querying. To preserve this seamlessness, a *spatio-temporal element* is defined of the form: $reg_1 \times \mu_1 \cup reg_2 \times \mu_2 \ldots \cup reg_3 \times \mu_3$; where for each i; $i \leq 1 \leq n$, reg_i and μ_i are any spatial and temporal elements respectively. This form of spatio-temporal elements is clearly closed under the three set operations (Gadia, Chopra and Tim, 1993).

4.1.3 Parametric Relations

Similar to the classical databases, in the Parametric Data Model, a *parametric tuple* is constituted by the concatenation of values. The only sole difference is size of the values i.e. the parametric values can be very large. In parametric databases, a parametric relation

can informally be defined as a set of parametric tuples. In this section, I discuss a few examples on temporal, spatial and spatio-temporal relations.

A. Temporal Relations

Figure 4.1 is an example of a parametric temporal relation – Emp that maintains the history of employees. The attributes of this relation are Name, Salary, and DName (department name).

Name	Salary	DName
[11,60] John	[11, 49] 50K [50, 60] 55K [55, 60] 60K	[11, 44] R&D [45, 60] Test
[0, 20] Tom ∪ [41, 51]	[0, 20] 45K [41, 51] 50K	[0, 20] Sales ∪ [41, 51]

Figure 4.1 Emp temporal relation

In order to capture the evolving value of an attribute, a *temporal value* of attribute *A* is defined as a function from a temporal element into the domain of *A*. One example - derived from the figure 4.1 - of such temporal value of the attribute - DName includes ‹ [11, 44] R&D, [45, 60] Test ›. The attribute value is described as - an employee works in department of R&D for the duration 11 to 44 and is associated with Test department for the time interval 45 to 60. If ξ is the representation of a temporal value then [[ξ]] delivers its domain. Based upon this, the domain of [[‹ [11, 44] R&D, [45, 60] Test › = [11, 60]. The semantic of the expression ξ ↓ μ is the restriction of ξ to the temporal element μ and one such example is that - if μ = [28, 55], ξ ↓ μ = ‹ [28, 44] R&D, [45, 55] Test ›.

A relation is assigned a key that uniquely identifies an object by values that remain invariant in the parametric domain of the object. Formally, a relation r over a scheme R, with $K \subseteq R$ as the key of r, is a finite set of tuples such that no key attribute value in a tuple changes from one point in its domain to another, and no two tuples should have the same key value. In the realm of the Parametric Data Model, these keys are pronounced as *parametrically invariant*. An object should have only one key and the Parametric Data Model establishes the one-to-one relationship between the tuples in the database and objects in the real world.

B. Spatial Relations

Figure 4.2 is an example that exhibits a parametric spatial relation –County. The attributes of this relation are CName (county name) and Crop. This County relation contains spatial and crop information that indicates, which crops are being cultivated in which county. It should be noted that $creg_1$, ... , $creg_6$ can be extremely complex, each consisting of multiple disjoint regions having complex shapes.

CName	Crop
$creg_1 \cup creg_2$ Story	$creg_1$ wheat $creg_2$ corn
$creg_3 \cup creg_4 \cup creg_5$ Orange	$creg3$ wheat $creg4$ barley $creg5$ rice
$creg_6$ Polk	$creg_6$ wheat

Figure 4.2 County spatial relation (Gadia and Chopra, 1993)

A *spatial tuple* is constituted by spatial values that have same spatial domains. A *spatial value* is a function from a spatial element into a domain of an attribute. One good example of a spatial value of attribute Crop may be - ‹ *reg*1 wheat, *reg*2 corn ›. The semantics of this example can be understood as - wheat and corn are being cultivated in *reg*1 and *reg*2 respectively; and the domain of this spatial value is [[‹ reg_1 wheat, reg_2 corn ›]] = $reg_1 \cup reg_2$.

C. Spatio-temporal Relations

Figure 4.3 is an example of a parametric spatio-temporal relation - *Well*. The relation consists of information about the different concentrations of two chemicals in the well. The concentrations are collected at different time points. In the example, the columns UGConc and DGConc are correspond to up-gradient and down-gradient wells, which is primarily, depends upon the direction of ground water flow (Gadia and Nair, 1993). In the relation p1 and p2 are the representations of two wells.

ChemName	UGConc	DGConc
p_1 x [0, NOW] Atrazine $\cup p_2$ x [0, NOW]	p_1 x [0, NOW] 1.0 p_2 x [0, 5] 1.5 p_2 x [6, NOW] 3.5	p_1 x [0, NOW] 0.9 p_2 x [0, 10] 1.4 p_2 x [11, NOW] 2.9
p_1 x [0, NOW] Simazine	p_1 x [0, 9] 10.0 p_1 x [10, NOW] 12.2	p_1 x [0, NOW] 9.2

Figure 4.3 Well spatio-temporal relation (Gadia and Nair, 1993)

A *spatio-temporal value* can be defined as the spatio-temporal assignment to an attribute that helps in capturing an attribute value, varying with space and time. A spatio-temporal

value of attribute *A* is a function from a spatio-temporal element into the domain of *A*. In this Well relation, the value of attribute UGConc of the first tuple is ‹ p_1 x [0, NOW] 1.0, p_2 x [0, 5] 1.5, *p*2 x [6, NOW] 3.5›. This can be described as – for the time duration 0 to NOW (current time) the concentration of Atrazine in well *p*1 is 1.0; and in the well p2, that changes to 1.5 for the time duration 0 to 5, and becomes 3.5 thereafter, from time instant 6 to NOW. The domain of this value can be extracted as: [[‹ p_1 x [0, NOW] 1.0, *p2* x [0, 5]1.5, p_2 x [6, NOW] 3.5›]] = p_1 x [0, NOW] ∪ p_2 x [0, NOW]

4.1.4 Parametric Structured Query Language

Similar to the other classical models, the Parametric Data Model governs its own SQL–like query language that is popularly known as Parametric Structured Query Language (ParaSQL). It is constituted by three mutually recursive expressions - relational expression, domain expression, and boolean expression – that help in evaluating to relations, parametric elements, and boolean values respectively. This section discusses the simplified Backus Naur Form (BNF) of all three expressions. The user-friendliness of ParaSQL and some interesting examples are also documented in this section. Readers interested in learning the algebra leading to semantics of ParaSQL in terms of the relational operations are encouraged to review the work of Gadia (Gadia and Nair, 1998).

A. Relational Expressions

```
<relational expression>  :=  <select statement> |
<relational expression> {UNION <relational expression>} +
<relational expression> {DIFFERENCE <relational expression>}+
<relational expression> {INTERSECTION <relational expression>}+

<select statement> := SELECT <attribute list>
                      [RESTRICTED TO <domain expression>]
                      FROM <relation list>
                      [WHERE <boolean expression>]
```

Figure 4.4 BNF of relational expression

In a database model a relational expression returns a relation - a set of tuples. It can be expressed using union, intersection, difference, or select statement. Figure 4.4 expresses the simplified BNF of the relational expression of ParaSQL. In this section the scope of our discussion is limited to the select statement only as it claims to be the most interesting relational expression. A ParaSQL select statement is shaped up, similar to a classical SQL-styled select statement, but extended with a new clause *RESTRICTED TO* that restricts the domain of tuples qualified by *WHERE* clause. The *WHERE* clause in ParaSQL functions in the same way as that of a classical SQL. That means, a tuple is returned if a boolean expression gets satisfied.

In the relational expression, it must be noted that *RESTRICTED TO* and *WHERE* clauses are optional and can be omitted if not needed. However, a ParaSQL parser recovers them as given here.

SELECT <attribute list>
FROM <relation list>

⇒

SELECT <attribute list>
RESTRICTED TO *R* x *T*
FROM <relation list>
WHERE TRUE

B. **Domain Expressions**

The functionality of a domain expression is to return the domain of - a tuple, an attribute or a relation. Figure 4.5 is the simplified BNF of domain expression.

```
<domain expression> :=
                        <atm dom exp> |
                       ( <domain expression> ) |
                        <domain expression> { + <domain expression> |
                                              * <domain expression> |
                                               - <domain expression>
                                            }* |
                                              ~ <domain expression>
<atm dom exp> :=
                    [[<attribute>]] |
                    [[<attribute> <op> <attribute>]] |
                    [[<attribute> <op> <value>]] |
                    [[<relational expression>]] |
                      <parametric element>
```

Figure 4.5 BNF of domain expression

The atomic domain expression [[<attribute>]] retrieves the domain (spatial or temporal) of a specified attribute. The domain expression, [[<attribute><op><attribute>]] collects the domain of two attributes, having <op> relationship. A related example may be – the domain of an expression that evaluates – the salary of one employee E1 is greater than the salary of another E2 can be expressed as [[E1.Salary > E2.Salary]].

The domain expression, [[<attribute><op><value>]] collects the domain of an attribute and a constant, satisfying the <op> relationship. An example includes - domain that the salary of an employee is greater than $80,000 is expressed as [[E.Salary > 80,000]]. The domain expression, [[<relational expression>]] conglomerates all subdomains of qualified tuples returned by the relational expression. Finally in the figure, the last kind of domain expression is a parametric element as a domain expression returns a parametric element. The domain expressions in ParaSQL have the scope to be operated using +, *, -, or ~ that correspond to union (∪), intersection (∩), difference (—), and complementation (¬) respectively.

C. Boolean Expressions

A boolean expression evaluates whether a prescribed boolean condition is true or false. Figure 4.6 is the simplified BNF of boolean expressions.

A boolean expression in ParaSQL functions in the same ways as in classical SQL and either qualifies or disqualifies a tuple. It is distinguished from classical SQL in the sense that in its constitution it can use domain expressions with set operators. Let us consider an example - a user needs to return the information of Sales department. In a ParaSQL query within the *WHERE* clause, it can be expressed as DName = “Sales”. Though, this expression can be viewed as an abbreviation of [[DName= “Sales”]] ≠ Φ. This can be described as – at one point of time the domain of an event where the department name is Sales is not empty. The boolean expressions can be connected using the boolean operators - AND, OR, and NOT.

```
<boolean expression> ::=
                    <atm bool exp> |
                    <boolean expression> { AND <boolean expression> |
                                           OR <boolean expression>
                                           }*  |
                                           NOT <boolean expression>
<atm bool exp> ::=
                    <domain expression> <set op> <domain expression> |
                    <attribute> <op> <attribute> |
                    <attribute> <op> <value>
```

Figure 4.6 BNF of boolean expression

4.1.5 User-Friendliness

As mentioned previously, the Parametric Data Model views a real world entity as a single tuple. It means that the whole object can be described in a single tuple unlike the classical models that require multiple tuples for the same purpose. This virtue of ParaSQL query helps reducing its complexity at the user level as it does not need to invoke the very expansive unnecessary self-joins that collect the scattered object fragments sprawled in multiple tuples. In his previous work, Gadia (Gadia and Chopra, 1993; Gadia and Nair, 1993) has shown that ParaSQL is capable to represent English queries in a very simple way. Let us consider an example - a user changes *or* condition to *and* condition in an English query. In ParaSQL, this change can be expressed in the similar way as the English query, by simply changing *or* condition to *and* condition in *WHERE* clause. To understand it better, consider the following two queries:

1. *Retrieve complete information about employees who worked in R&D or Sales department.*

2. *Retrieve complete information about employees who worked in R&D and Sales department.*

These two English queries are different from each other in their boolean condition. However, to express these queries in those models, which describe an object in multiple tuples and rigorously use self-joins, is difficult for their query language. In ParaSQL, the boolean expressions of the above queries are expressed as follows:

[[DName= “R&D”]] ≠ Φ OR [[DName= “Sales”]] ≠ Φ

[[DName= “R&D”]] ≠ Φ AND [[DName= “Sales”]] ≠ Φ

The meaning of the domain expression, [[DName= “R&D”]] ≠ Φ can be understood as – at one point of time the name of a department was R&D. In *WHERE* clause these two expressions can be abbreviated as follows:

DName= “R&D” OR DName= “Sales”

DName= “R&D” AND DName= “Sales”

In addition to this, ParaSQL eases the expressing of queries where *an event did not happen*. Let us consider the following query to have better understanding.

3. *Retrieve complete information about employees while they did not work in R&D or Sales department.*

In ParaSQL, the restriction is expressed by domain expression as follows:

(~ [[DName= “R&D”]]) ∪ (~ [[DName= “Sales”]])

It is worth to notice that the complementation (~) as well as unions (∪) of parametric elements is a parametric element.

4.1.6 ParaSQL Examples

For the sake of further illustration of ParaSQL, let us study the following English queries that assume that the underlying relations are temporal, spatial, and spatio-temporal relations, respectively.

4. *Retrieve the details from the Emp relation about employees while they worked in either R&D or Sales and did not earn more than $80, 000.*

```
SELECT *
RESTRICTED TO (         [[E.DName = "R&D"]]
                    +     [[E.DName="Sales"]]
                )    * (~ [[E.Salary > 80000]] )
FROM Emp E
```

This ParaSQL query retrieves the tuples from the Emp relation (a parametric temporal relation). For each qualified tuple, it restricts it to a temporal domain in such a way that an employee worked in R&D or Sales department and the salary of the employee less than $80, 000.

5. *Retrieve the complete information about counties whose land acres are greater than 100, 000 and which grow wheat and corn.*

```
SELECT *
  FROM County C
WHERE  Area ( [[ C.CName ]] ) > 100000
           AND C.Crop = "wheat" AND C.Crop = "corn"
```

In this query, County relation is a parametric spatial relation that consists of crop information with spatial regions where crop is currently cultivated. In this ParaSQL, in the *WHERE* clause, the condition - if the area of a county is greater than 100, 000 acres and the county grows wheat and corn - is checked.

The expression [[C.CName]] returns a spatial element and the spatial function - Area ([[C.CName]]) - retrieves the area of the domain of a county.

5. *Retrieve the current information about up-gradient wells which contain the concentration of Atrazine less than 1.5 and corn is being cultivated in a region to which the wells belong.*

```
SELECT *
        RESTRICTED TO   [[SELECT *
                          FROM County C WHERE C.Crop = "wheat"]]
                        * [[W.UGCons < 1.5]] * NOW
FROM Well W
WHERE W.ChemName = "Atrazine" AND W.UGCons < 1.5
```

This ParaSQL query operates on the spatio-temporal relation – *Well* and retrieves the tuples in such a way that the chemical name is Atrazine and up-gradient concentration is less than 1.5. The qualified tuple - satisfied the boolean condition - is restricted to the spatio-temporal element returned by the domain expression in the *RESTRICTED TO* clause. The domain expression intersects three parametric elements: 1) the first parametric element is a spatial element which is the domain of counties that grow wheat; 2) the second parametric element is a spatio-temporal element which is the domain of the up-gradient well which contains Atrazine with concentration less than 1.5; 3) the last parametric element is the temporal element NOW.

4.2 CONCEPT OF SUBSET

In any data model, the concept of subset and subquery is of paramount importance. The process of subsetting and subquerying helps in extracting a smaller set of data - out of a hugely complex database - as the desired dataset, called subset. The slicing and dicing of this smaller subset is extremely easy as compared to the original mammoth dataset and oftentimes reveals important insights that lead to efficient decision making. In this chapter, the consideration is given to the concept of subset and subquery in Parametric Data Model. The new challenges: implementation of the concept of subset, implementation of the concept of subquery, and further expansion of ParaSQL, taken up in this research work, are extensively discussed in this chapter.

4.2.1 Introduction

In any database model, the analysis of the data can be eased by extracting the smaller set of the data of interest, called subset from the mammoth original dataset. Thus subset helps enhancing the performance of system by avoiding the iteration through the huge parental data in further analysis. A subset, its specification, or the formal process for its extraction can be complex. In database community subsets are extracted through SQL-like query. An SQL query can be a composition of subqueries. Each subquery can be seen as an iterative step toward the extraction of the desired subset. For this to work, subqueries should result into relations that have the same structure as relations in a given

data model. In the previous versions of the prototype of the Parametric Data Model (Narayanan, 2009), if a ParaSQL query is executed on a spatio-temporal dataset such as NC-94 (NCRA, 2004), the result set is a stream of raw text data. It is worth mentioning here once again, the NC-94 dataset is used as a test bed in this entire dissertation and chapter 3 provides a detailed description about it, including its three fragments – climate, crop, and soil. The resultant can either be directed to an output console such as DOS (Disk Operating System) for display or dumped to a static text file for later use, dismantling the relational structure in both cases. This may contribute to wide variety of applications, but falls short from various perspectives such as - the streamed out data cannot be queried further, unlike classical databases, where subset provides additional strength to the database management system and the parametric prototype lacks this potential functionality.

In this study, the implementation of the Parametric Data Model has been enhanced further. In this current version, it accepts relational dataset as input and generates the relational dataset as output and thus preserves the relation's integrity, where further subqueries can be applied on the generated dataset. As mentioned earlier, the NC-94 dataset is used to validate the functionality of the prototype. It is a compilation of 30 years crop-climate-soil database, focused on producing a continuous high quality county-level data of commonly measured quantities of air temperature, precipitation, crop yield and soils data.

4.2.2 Data Retrieval in Parametric Data Model

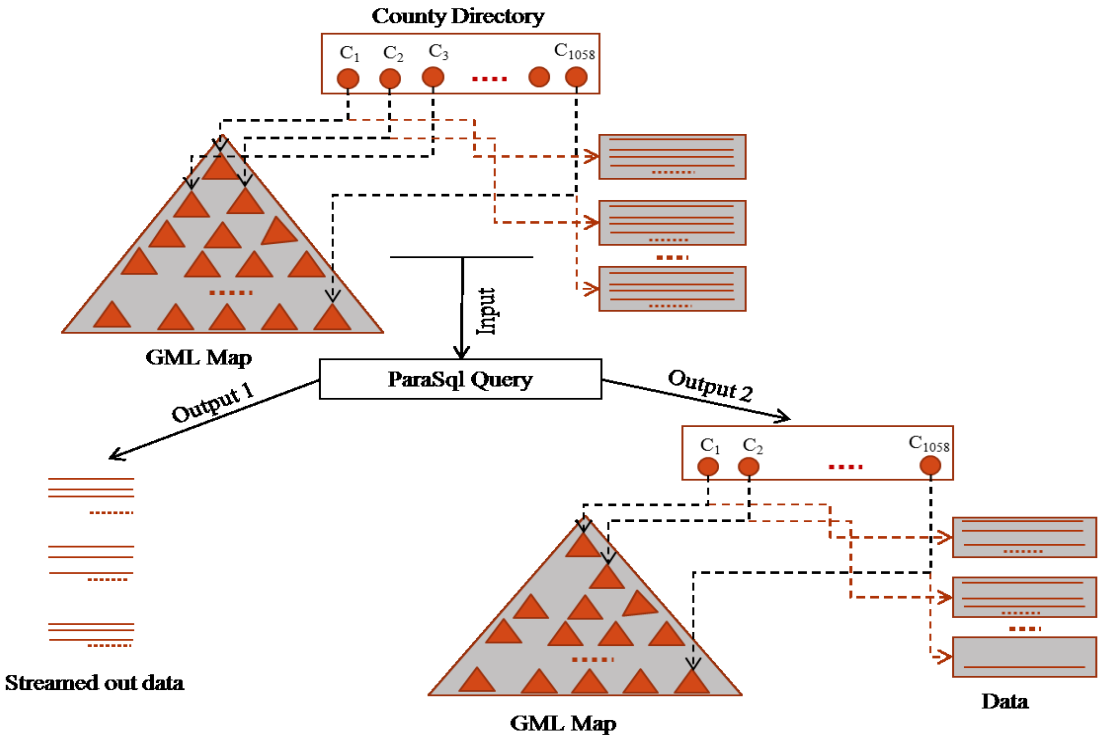

Figure 4.7 ParaSQL query execution output

One of the important motivations of this work is to further develop the existing prototype of the Parametric Data Model to accommodate the concept of subquery. This means, if the input of ParaSQL query is a relational dataset, the resultant output should also be a relational dataset, which must have the scope for further subquerying. The existing prototype does not support this concept and imposes restrictions in case of subquery.

This encourages the further development of the prototype in such a way that could start preserving the relational structure of the output as well, as happens in classical models. The following sections elaborate the existing prototype and enhanced prototype more clearly from output point of view.

4.2.3 Existing Prototype of Parametric Data Model

The existing prototype of the Parametric Data Model, validated using NC-94 dataset, has been extensively used in different domains including the database lab of Iowa State University, where it is used for various purposes e.g. data analysis, teaching, and further research.

When a ParaSQL query is applied to the relational dataset, the resultant (qualified) dataset is no longer a relational dataset rather it is streaming out as raw text which can be either seen on DOS console or can be stored into a text file. If someone wants to apply a ParaSQL subquery further on the resultant data, she/he cannot do it as the resultant streamed out data fails to preserve the structure of a relation (Figure 4.7 output 1).

4.2.4 Enhanced Prototype of Parametric Data Model

The enhanced prototype type is able to produce relational subset. Figure 4.7 (output 2) is the pictorial representation of the relational dataset utilized as input to a ParaSQL query and the resultant dataset after query execution. It can be noted that the output of the applied ParaSQL query is a subset of the parent dataset.

In the best case scenario, a subset can consists of the equal number of the counties as it was in the parent dataset but in average case scenario there is possibility that few counties will be disappeared as they could not qualify the condition. A county will be treated as the qualified county (survivor) if it has atleast one qualified tuple (tuple satisfies the given condition), otherwise, a victimized county. If a county fails to register any of the

tuples as the qualified tuple, the existence of that victimized county will vanish from the GML map (OGC, 2000) and dataset but the set of survivor counties is good to be utilized further for subquery thus preserves the structure of the relation. It can be noted that county 3 (C_3) is vanished from the county directory after the execution of a ParaSQL query.

4.2.5 Query Set for Subset

```
SELECT *
  INTO SubClimate
  RESTRICTED TO [[(C.MaxTemp +C.MinTemp)/2<0]
FROM Climate C
```

Figure 4.8 (a) Sample ParaSQL query for subset concept

This is one of the two important ParaSQL queries, used in the concept of subset implementation for the Parametric Data Model. This helps to store the qualified data into the new dataset from the old dataset. As shown in the sample query (Figure 4.8(a)), the data is read from the old dataset, *Climate* and only the qualified data which satisfy the *RESTRICTED TO* condition will be stored into the new dataset *SubClimate*. In this work

```
SELECT *
FROM SubClimate C
```

Figure 4.8 (b) Sample ParaSQL query for subset concept

the climate dataset is stored with county as the lowest spatial granularity. The expression [[(C.MaxTemp +C.MinTemp)/2<0]] returns the temporal element (Gadia and Vaishnav,

1985) when the average temperature was less than 0. The second sample ParaSQL query (Figure 4.8(b)) retrieves the data from new relation *SubClimate* and thus helps in validating the newly implemented functionality.

4.3 CONCEPT OF SUBQUERY

The implementation of the concept of subset in Parametric Data Model encourages another interesting concept to be implemented in the prototype of the model, the concept of subquery. The preserved relational structure of the output of a ParaSQL query offers the scope for further subquerying. In a ParaSQL query, a query (inner) can be embedded inside another query (outer) and this complex query is applied to the relation(s), conforming the concept of the classical database paradigm, which involves inner and outer query. In this study, the development of subquery concept in the Parametric Data Model is another important motivation, where a set of parametric queries (*Para*SQL): outer query & inner query operates on parametric data. The resultset of inner query restricts the domain of execution for outer query. This enhanced functionality is validated using spatio-temporal NC-94 climatic dataset.

4.3.1 Query Set for Subquery

A sample ParaSQL subquery is shown in figure 4.9. In this complex query, the subquery (inner query) is executed first and returns the results from the relation, *ClimateSub*.

```
SELECT *
  RESTRICTED TO [[SELECT * FROM ClimateSub C1]]
FROM Climate C
```

Figure 4.9 Sample ParaSQL query for subquery concept

Then the outer query is executed, retrieving the results from relation, *Climate*, but the output is restricted to the temporal element of the result of the inner query. This clause returns the temporal domain of the results returned by inner query and restricts the domain of execution of outer query. The subquery concept is a baby step in this direction in the Parametric Data Model.

4.4 PARASQL EXTENSION

4.4.1 Where Clause in ParaSQL

In this study, the use of the new design of the storage technology is taken into consideration. In the redesign of the storage format, the idea is to make the technology more generic, hide the complex details from the user and to provide clear configuration. In the previous design, only the page header contains the useful information: current page Id, next page Id, and number of tuplets in the page. On the execution of a query, always the first tuple is resulted out. In order to fetch a particular tuple, iteration will go over all the previous tuples (hence tuplets), before the tuple of interest results out. In the new design the concept of tuple header is conceived. In the storage, every tuple contains tuple

Id and number of its tuplets stored in the storage. As soon as the dataset is loaded into the storage, an index will be generated consisting of tuple Id and its header address as key-value pair. This helps in avoiding the unwanted iteration to reach to the tuple of interest. Reading of the tuple header, also allows the restricted display of tuplets as number of tuplets in a tuple is already known from the header. The new storage format is shown in figure 4.10. As shown in figure file header will contain tuple's header size. The values in this header will be applicable at the file level. Once a user has access the tuple header, the exact number of its tuplets will be known and can be access easily using some kind of programming loop-such as *for*. The technology is for the general purpose use for the spatio-temporal objects where spatial component is always stationary. In order to load the data in storage, a general purpose loader is developed. This works for any relation of this type such as climate, crop, soil of NC-94 dataset and their subset. It hides many details about the relation such as- whether a relational has a single tuple or multiple tuples or the attribute key constituted by a single attribute or multiple attributes and hides geometry too. The relation will be sorted based on the parametric key before being stored.

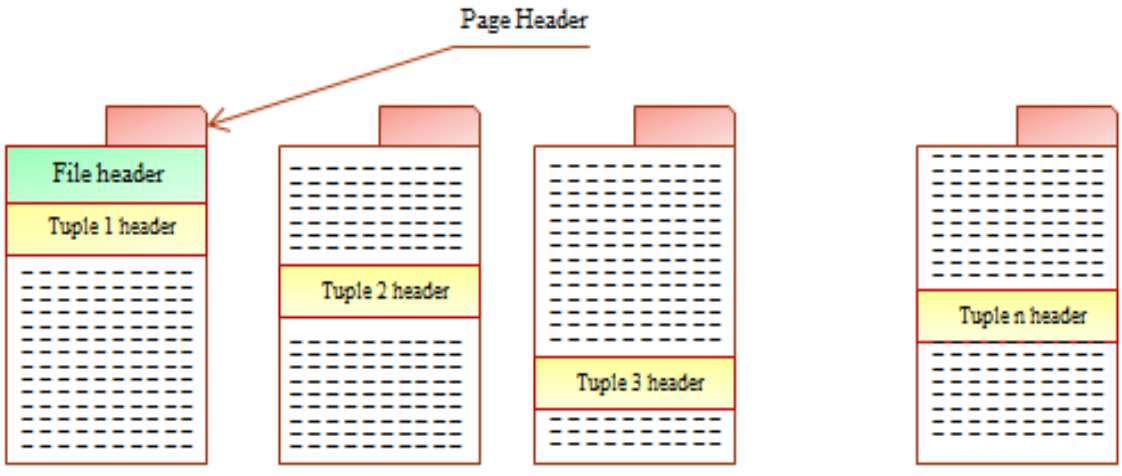

Figure 4.10 New storage format

DName	MName
[11,44] Toys [45,60] Shoes	[11,60] John
[45,49] Toys	[45,49] Leu
[41,47] Clothing	[41,47] Tom
[71,NOW] Clothing	[71,NOW] Inga

Figure 4.11 (a) The Dept1 relation

DName	MName	Start	End
Toys	John	11	44
Toys	Leu	45	49
Clothing	Tom	41	47
Clothing	Inga	71	NOW
Shoes	John	45	60

Figure 4.11 (b) The Dept2 relation

To have better understanding of *where* clause in Parametric Data Model, let us consider the two representations of Dept relation as an example. The representations are parametric (Dept1) and non-parametric (Dept2) relations and shown in figures 4.11 (a) and 4.11(b) respectively. In Dept1 relation, DName is the key. A key in the relation provides a persistent identity to an object. At every instant of the time both the relations have the same information. Let us consider a simple query (Example 1) and allow it to answer from both the relations.

Example1. *Give information about managers who were managers atleast during [11, 50]*.

For the Parametric Data Model, the above query is written like Q1 and result is shown in figure 4.11 (c). For non-parametric relation, the query expressed in the same way (Q2) results empty relation (Figure 4.11(d)).

```
Q1:  SELECT x.MName
       FROM Dept1
     WHERE [[x.MName]] ⊇ [11, 50]
```

MName
[11,60] John

Figure 4.11 (c) Output of Q1

```
Q2:  SELECT x.MName
       FROM Dept2
     WHERE [[x.MName]] ⊇ [11, 50]
```

MName

Figure 4.11 (d) Output of Q2

It is not claimed that the query cannot be expressed in the second representation – Dept2 (non-parametric), but it should be more complicated. This hints, the Parametric Data Model offers the best structure leading to perhaps the best query language.

Thus, in Parametric Data Model, the *where* clause looks at the whole object view unlike other database model, where it looks at the fragment of the object. So the Parametric Data Model, it can do much sophisticated internal navigation.

```
SELECT Precipitation
  FROM Climate C
WHERE NEIGHBOR (C.FIPS, 19169) = TRUE
```

Figure 4.12 (a) Sample ParaSQL query tested for *where* clause expansion

```
SELECT AvgPrecipitation
   FROM Climate C
WHERE NEIGHBOR (C.FIPS, 19169) = TRUE
```

Figure 4.12 (b) Sample ParaSQL query tested for *where* clause expansion

Therefore, the *where* clause is a strong operator. In this research work, the *where* clause is expanded further to harness its potential.

In the previous implementation of the prototype, in order to evaluate the boolean condition in the *where* clause, regardless of status of the outcome, the object was navigated internally. The iterator used to iterate through each individual tuplet in the tuple. In a *where* clause, if an object did not satisfy the given condition, iterating through inside it added the extra overhead on the system's functionality that caused significant delay in reporting the output to the end user.

The intelligent thoughts are given on how the lengthy internal navigation through an unqualified object can be avoided with the perception that it will enhance the performance of the system significantly. In the expanded version of the *where* clause, there are two iterators employed to carry out the task. One iterator - say external iterator - works at the object level, whereas the other one - say internal iterator - navigates through the object itself. So in a query, when a boolean condition in *where* clause is encountered, the external iterator starts iterating over the objects, one at a time. If an object qualifies the given condition, it is further exposed to the internal iterator that navigates though it. Once the navigation though the qualified object is completed, the external iterator moves to the next object and same process is repeated for all other object in the dataset. The unqualified objects are discarded at the external iteration only - diminishing any scope of internal navigation - and external iterator moves to the next objects. This improves the system performance in significant way as the internal iteration though any unqualified object is avoided. Figures 4.12 (a), and 4.12 (b) are two sample queries used for the testing of the expanded version of the *where* clause. The queries operate on the NC-94,

Climate dataset. In both the queries, the number 19169 uniquely identifies Story County in Iowa. The expression ‘C.FIPS’ is a numeric value that represents a unique spatial location - in this case a county (United States Federal Government, 1994). The semantic of the boolean condition is that a county that is considered to be qualified for further processing should be a neighbor of Story County. For the qualified counties, the first query - 4.12(a) – reports the simple precipitation whereas the second query - 4.12 (b) - retrieves the average precipitation.

CHAPTER 5. VISUALIZATION OF LARGE SPATIO-TEMPORAL DATA

5.1 NEED FOR VISUALIZATION OF SPATIO-TEMPORAL DATASETS

In the scientific community, significant research is going on to make the visualization matured (Wijk, 2005). A potential use of visualization has been experienced in numerous distinguish research areas and applications. Visualization has helped solving many problems, and is continued to offer aids to new directions.

The result of the rapid growth in Geographical Information System (GIS) is the collection of mammoth heterogeneous data, collected by various agencies available today. Huge amount of potentially important data is stored at various distributed locations and in heterogeneous formats. According to the statistics available in database literature (Franklin, 1992), about 80% of all the stored data are spatial databases. Lately large spatial, temporal, and spatio-temporal databases have attained significant momentum in academic, research and industry. In order to have their full understanding, spatially and temporarily, visualization of such datasets is utmost necessary.

The exploratory visual analysis of such large multifaceted datasets offers a helping hand in preliminary investigation (Gahegan, 2005; Kraak, 2006), where especially, spatial artifact of such datasets is of paramount importance. The spatial components in heterogeneous spatio-temporal databases may be in different formats and each format does not guarantee the easy acquisition and sharing of the data using off-the-shelf visual analytic GIS software systems such as ESRI ArcGIS, and GeoMedia etc., available commercially or in research community. It is very unlikely that all the software systems

use the same format (Tarnoff, 1998) as each vendor has its own proprietary design, data model, and storage technology. To facilitate the exchange, sharing and visualization of the information among various heterogeneous spatio-temporal database systems, conversion tools- from one format to another - play vital role. Most of the scientific visualization systems allow the same spatio-temporal data to be examined from different perspectives, helping in intuitive understanding about the data. In many scientific systems which undertake spatio-temporal visualization such as (Shahabi et al., 2010; Sharma et al., 2011; Huang et al., 1999; Huang et al., 2001; MacEachren et al., 1999; Gahegan, 2005; Kraak, 2006; Takatsuka et al., 2002; Demiryurek et al., 2010) the visualized object is selectable which offers better retrieval of more intrigue information and allow new queries to be applied on the selectable output. Along with intuitive data analysis visualization provides valuable assistant in in intelligent decision making (Tory and Moller, 2004), however, from human perspective, the interaction with a visualization tool can strongly influence the understanding of the data. In order to develop intelligent and efficient choices, a good understanding of the purpose and meaning of the spatio-temporal visualization is needed.

5.2 VISUALIZATION IN PARAMETRIC DATA MODEL

Based on the previous discussion on the visualization and spatio-temporal databases, it is worth mentioning that a spatio-temporal data model should be equipped with visualization capability in order to perform intuitive analysis on the dataset and making intelligent decisions. Though the previous version of the Parametric Data Model is able to

handle the spatial and temporal domains of a spatio-temporal dataset but completely unaware of the potential of visualization.

Another significant contribution of this study is the integration of the concept of data visualization in Parametric Data Model. In this data model, the visualization works along with subset, but in this section the emphasis is given only on visualization. A modular implementation for visualization is done in the Parametric Data Model and the ParaSQL query is further enhanced which invokes the visualization. A sample ParaSQL query is given in figure 5.1. The query is validated using NC-94 climatic data as a use case.

```
SELECT *
   INTO ClimateSub
   DISPLAY Precipitation VIA MapCountyRanges.xml
   RESTRICTED TO [[(C.MaxTemp + C.MinTemp) < 0]]
FROM Climate C;
```

Figure 5.1 Sample ParaSQL query for visualization concept

As it could be noticed, this ParaSQL query embraces new clauses in this implementation such as *DISPLAY*, and *VIA*. The first clause is responsible for the display of that attribute which follows it in the query. In this sample query, the attribute-Precipitation will be displayed. The *VIA* clause is responsible to facilitate the display with suitable color coding scheme, narrated in an XML configuration file, which is supplied as input in the query. This sample query portraits the dual roles. It retrieves the data from the relation *Climate*, and stores back to another relation *ClimateSub*. Only those records are stored which qualify the *RESTRCIED TO* condition (explained earlier). In the qualified results the query displays one attribute, called Precipitation (on-the-fly) with a

color coding scheme written in MapCountyRange.xml file (refer to figure 4.5). The validation of this implementation has been done using spatio-temporal NC-94 dataset.

One of the applications of this is to calculate and analyze the degree of coldness (Sharma and Gadia, 2010) in the North Central Region of the United States. For the sake of simplicity, the region of interest is restricted to Iowa only. The application is described below in detail.

5.2.1 To Calculate and Analyze the Degree of Coldness

State of Iowa is an agricultural rich state in North Central Region and is divided into ninety nine counties. North Central Regional Association (NCRA) (NCRA, 2004) in the United States maintains agricultural databases to facilitate crop and risk analysis, pest management and forecasting. NC-94 (NCRA, 2004) is one such dataset, which is intensively used and is available for public use through many sources to process and analyze to get future predictions about agriculture. In this work, the cumulative degree of coldness is calculated in Iowa with spatial granularity as county in last thirty years. To demonstrate the degree of coldness, a base color is chosen and counties are rendered with different shades of that color based on the degree of coldness. Higher intensity of the color reflects the higher coldness, whereas the lower intensity corresponds to lower coldness.

On the execution of the query, the first thing happens is the query parsing and a parse tree is generated. This parse tree is taken as input in generating the expression tree. These are the natural phenomenon of any database management system. The xml format of the parse tree and expression tree of this sample query is shown in figures 5.2(a) and 5.2(b).

```
- <ParseTree>
  - <QueryExpression>
    - <SelectStatement>
      - <SelectClause>
        + <AttributeList>
        </SelectClause>
      - <IntoClause>
          <RelationName>ClimateSub</RelationName>
        </IntoClause>
      - <DisplayClause>
          <AttributeName>Precipitation</AttributeName>
        </DisplayClause>
      - <ViaClause>
          <AttributeName>MapCountyRanges</AttributeName>
        </ViaClause>
      - <RestrictClause>
        - <UnaryDomainExpression>
          - <SimpleExpOpSimpleExp>
            + <AtomicSimpleExpression>
              <ROperation>lessThan</ROperation>
            - <UnaryExpression>
                <Number>0</Number>
              </UnaryExpression>
            </SimpleExpOpSimpleExp>
          </UnaryDomainExpression>
        </RestrictClause>
      - <FromClause>
        - <RelationList>
          - <Relation>
              <RelationName>Climate</RelationName>
              <NickName>C</NickName>
            </Relation>
          </RelationList>
        </FromClause>
        <WhereClause />
      </SelectStatement>
    </QueryExpression>
  </ParseTree>
```

Figure 5.2 (a) Parse tree (xml view)

```
- <ExpressionTree>
  - <Iterator type="relationscan" relname="Climate">
    - <Projection isAll="true">
        <Annotation />
      </Projection>
    - <Intostatement hasRelation="true">
        <RelationName relName="ClimateSub" />
      </Intostatement>
    - <Displaystatement hasArgument="true">
        <Attribute attrName="Precipitation" />
      </Displaystatement>
    - <Viastatment hasArgument="true">
        <Attribute attrName="MapCountyRanges" />
      </Viastatment>
    - <Restriction existDomainExp="true">
      - <DomainExp opType="unary">
        - <ArithOpConst opType="lessThan">
          - <Arithmatic opType="plus">
              <Attribute attrName="MaxTemp" attrPos="6" type="float" />
              <Attribute attrName="MinTemp" attrPos="7" type="float" />
            </Arithmatic>
            <Const value="0" type="float" />
          </ArithOpConst>
        </DomainExp>
      </Restriction>
    - <WhereCond>
        <Condition isTrue="false" />
      </WhereCond>
    </Iterator>
  </ExpressionTree>
```

Figure 5.2 (b) Expression tree (xml view)

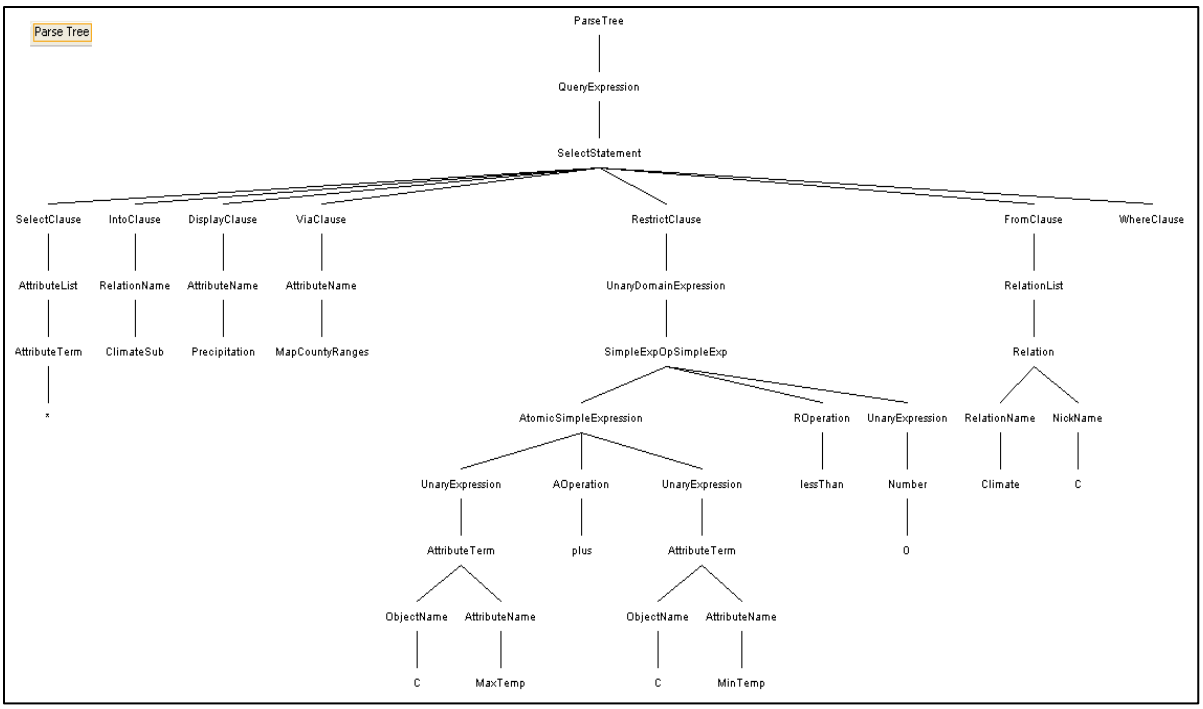

Figure 5.3 (a) Parse tree (graphical view)

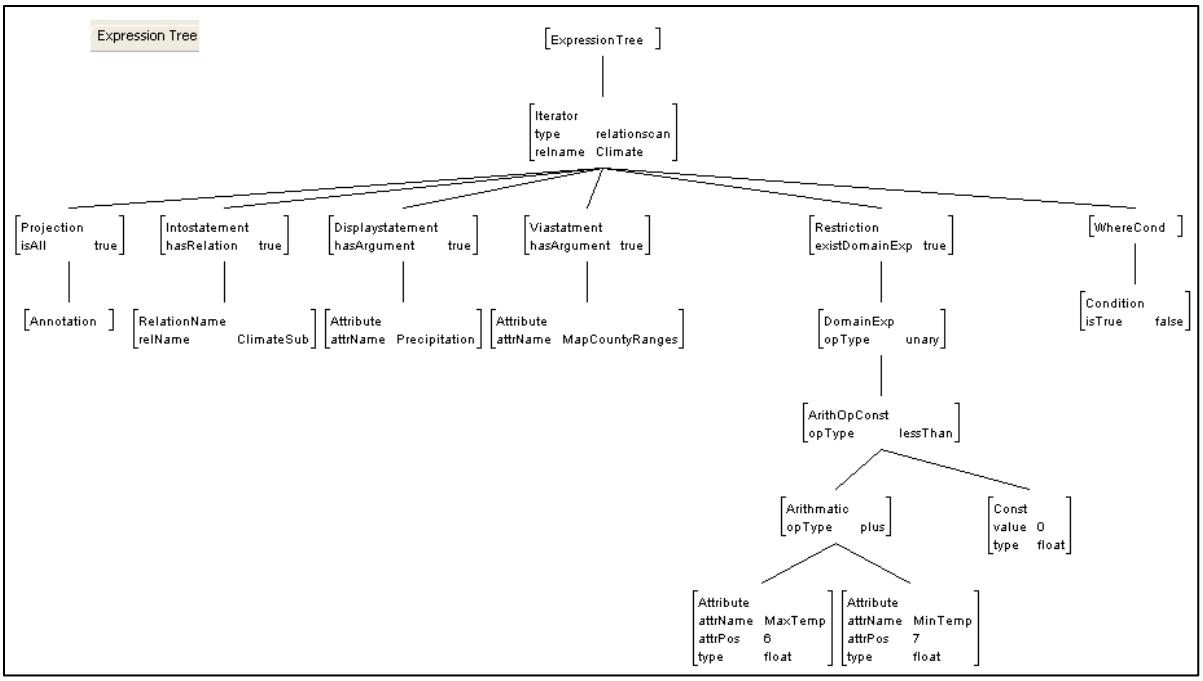

Figure 5.3 (b) Expression tree (graphical view)

The graphical formats of the same are depicted in figures 5.3(a) and 5.3(b) respectively. An iterator, which is responsible to retrieve the data from the disk into buffer(s), reads the expression tree to learn about the attributes being used in this current query and the expression tree in figure 5.2 (b) clearly demonstrates that.

To offer better understanding of the parse tree and the expression tree of the input ParaSQL query, the graphical versions of parse tree and expression tree are also generated and are shown in figures 5.3 (a) and 5.3 (b) respectively. The resultant of the execution of the ParaSQL query consists of many attributes. A few of the attributes out of them are very important in order to calculate the degree of coldness in an individual county. In a county, those days are considered colder, which have average temperature less than 0 and the summation of all such cold days, divided by total number of days, produces to degree of coldness in an individual county. In order to display the coldness in a graphical map, a color coding scheme is used, which paints the colder counties with the darker shades and warmer counties with lighter shades of a color (blue in this application) on a graphical map, reflecting a better understanding about the coldness of counties. The configuration file (xml file) provides the opportunity to the users to customize the number of ranges a base color (blue here) can possesses.

In this application, for the sake of clear and distinct visualization of the coldness the display is restricted to the counties of Iowa only and ignored for the rest of the states in North Central Region.

5.2.1.1 Architecture

As mentioned earlier, the prototype of the Parametric Data Model is developed using java programming language, to offer similarity in the technology, this application of coldness calculation is developed using java programming. The figure 5.4 shows architecture (after subset processing), that mainly focuses on the visualization of the application. It assumes, the collection of {county, No of cold days} pair is generated by the subset process invoked by INTO clause (Figure 5.1).

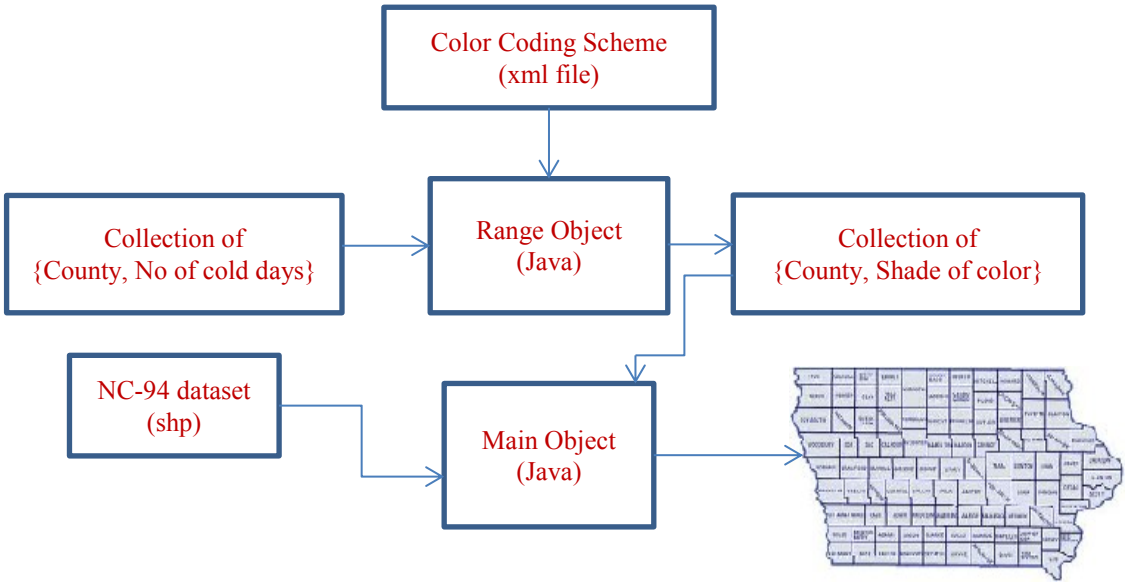

Figure 5.4 Architecture of coldness display in Iowa

A user defined java object – *Range.java* accepts this collection as input along with a color coding scheme defined in an xml file - MapCountyRange.xml (in this case). This xml file (Figure 5.5) consists of quite a few interesting values such as base color, number of ranges of the coldness and correspondingly the shades of base color are divided. Based

upon the range division, the *Range Object* assigns a shade of the base color to a county and a collection of {County, Shade of color} pairs is produced.

```
<?xml version="1.0"?>

<Color_Coding_Scheme>

        <Base>000000</Base>
        <Number_of_Range>5</Number_of_Range>
        <State_Fips>19</State_Fips>
        <State_Name>Iowa</State_Name>
        <Total_Number_Days>11000</Total_Number_Days>

</Color_Coding_Scheme>
```

Figure 5.5 Configuration file - MapCountyRange.xml

There is another user defined java object –*Main.java*, which is responsible for the display of the coldness on the geographical map. The object takes the collection of pairs as input along with the spatial component of NC-94 dataset (in shape format) and renders the coldness on a geographical map with spatial footprint as county.

5.2.1.2 Algorithms

This section describes the pseudocode to calculate the degree of coldness in a county. Step 1 helps to store the set of states extracted from NC-94 dataset into an array. As the region of interest for this application is confined within Iowa only so the array in step 1 is iterated to extract the spatial data in Iowa only and variable *S* in step 4 temporarily holds the state name. Step 5 helps to check, whether the stored state in variable *S* is Iowa. Once

step 5 returns true, all counties of Iowa are extracted and stored in another array *CC* as shown in step 7. This array is iterated to calculate the number of days, in which the average of maximum temperature and minimum temperature is less than zero and step 10 in this pseudocode invokes that operation.

ALGORITHM: Pseudocode for Degree of Coldness Calculation

```
1: Array SS ← {set of all states in North Central Region in NC-94 dataset }
2:  WHILE (SS. length >=0)
3:        BEGIN
4:         S ← Extracted state from array
5:           IF (S equals Iowa)
6:            BEGIN
7:             Array CC ← set of counties of Iowa
8:               FOR (int i=0; i <=CC.length; i++)
9:                 BEGIN
10:                  Number of cold days: n ← { ( (Temp .MAX + Temp .MIN) / 2 ) ≤ 0 }
11:                  Number of day in 30 years: N
12:        Degree of coldness = n/N
13:       END
14:      END
15:    END
```

Once the numbers of cold days are calculated, thereafter, it is extremely straight forward to calculate the degree of coldness over a 30 years period and step 12 in pseudocode helps to return the degree of coldness of a particular county at a time. Thus, finally, when the *for* loop in step 8 is exhausted, and the degree of coldness for all counties in Iowa is calculated.

5.2.1.3 Coldness Rendition on Geographical Map

Figure 5.6 is a geographical map, showing all ninety nine counties in Iowa. To show the coldness, each county is painted with a shade of blue color. The shade with higher intensity reflects the higher degree of coldness and that with lower intensity reflects the lower degree of coldness in a particular county for duration of thirty years.

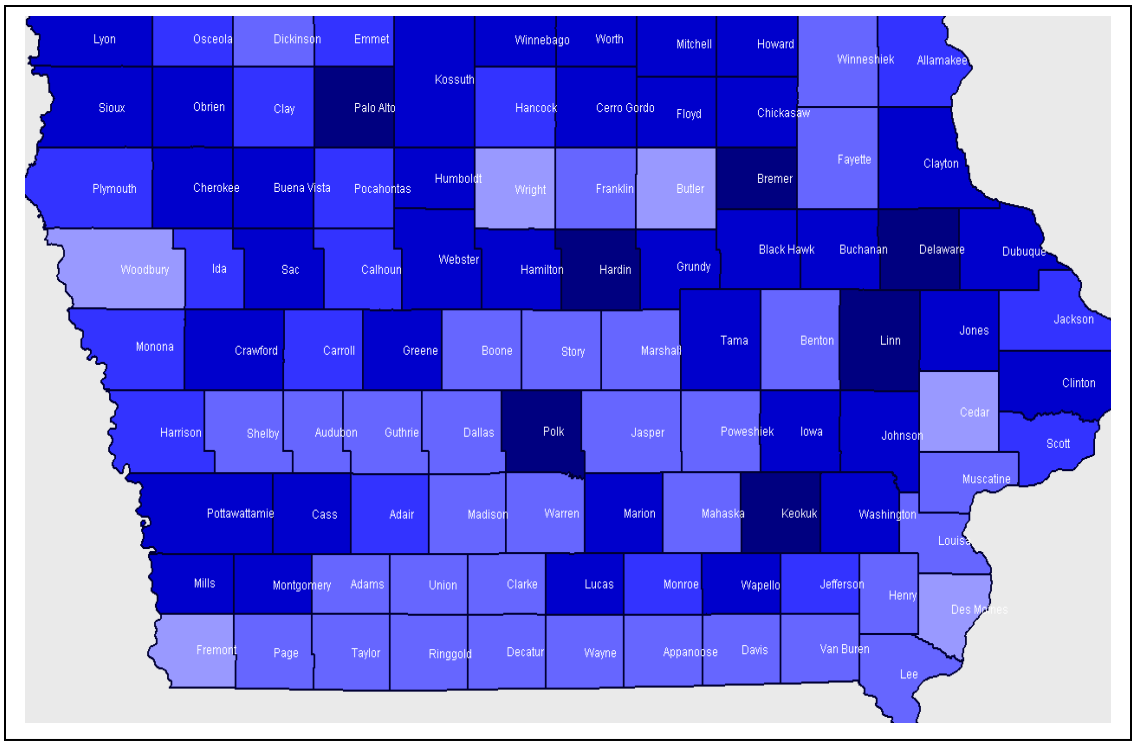

Figure 5.6 Display of coldness in all counties of Iowa

As shown in this geographical map, most of the counties at up north in Iowa have more intensified shades; whereas the counties at the down south in Iowa have lighter shades of blue color. From this, it can be inferred that the upper side (north) counties of Iowa have higher degree of coldness than that of down side (south). County Palo Alto is one of the

counties consisting of highest degree of coldness whereas Cedar belongs to that group of counties, which has lowest degree of coldness.

5.3 AUTOMATIC CONVERISON AND VISUALIZATION OF SPATIO-TEMPORAL QUERY DATA - AUTOCONVIZ

5.3.1 Introduction

With the rapid growth in Geographical Information System (GIS), the result of the data collection by various agencies is the mammoth heterogeneous data available today. Huge amount of potentially important data is stored at various distributed locations and in heterogeneous formats. Each format does not guarantee the easy acquisition and sharing of the data using off-the-shelf leading GIS software systems such as ESRI ArcGIS, and GeoMedia etc., available commercially or in research community.

It is very unlikely that all the software use the same format (Tarnoff, 1998) as each vendor has its own proprietary design, data model, and storage technology. In facilitating the exchange and sharing of the information among various heterogeneous spatio-temporal database systems, conversion tools- from one format to another - play vital role.

Table 5.1 NC-94 climate data sample

FIPS	StFIP	CoFIPS	Year	Day	Radiation	MaxTemp	MinTemp	Precipitation
19169	19	169	1992	7905	0.0	6.66	-7.94	-16.39

The development of a new custom tool from scratch which can accommodate the present spatial format of the artifact is difficult, time-consuming for every user and requires skills not easy to possess, leaving the many datasets underused consequently. Often times, the scientific community and practitioners, experience the need of the conversion of the spatial artifact of the dataset into a more suitable format easily accommodable into the domain of a proprietary or non-proprietary software tools for exploratory visualization. Though among the various spatial formats available today in spatial community, because of its *open source standard* nature, the Geographical Markup Language (GML) format overshadows the use of others and adheres to its central position and is of our interest in this work. It is an XML (Extensible Markup Language) format stores the geographical information that includes both the spatial and non-spatial properties of geographic features (Cox et al., 2003) and promises to support variety of sources, widely. The format of GML data is text; a universal format thus offers easy integration across a variety of resources and has the potential for real-time data sharing.

Due to the abrupt growing of the data, the analysis of huge amount of data is getting difficult, rather interesting subset of the vast data is extracted to do analysis. The smaller subset facilitates the desired analysis efficiently. In this work, the interest is in spatial/ spatio-temporal dataset which may exist in various formats. Let us consider an example of spatio-temporal dataset NC-94 (NCRA, 2006). This is a climate-crop-soil data for the duration 1971-2000, consisting of spatial and temporal component for North Central Region of the United States and for simplicity the focus is on climatic dataset only. The spatial and temporal artifacts are segregated into two separate groups, linked through their location references – FIPS code (USFG, 1994).

The spatial artifact is in Geographic Markup Language (GML) format (OGC, 2000) consisting of spatial and non-spatial attributes, whereas the temporal artifact is one of the attributes of the climatic tuples consisting with climatic attributes as – FIPS, State FIPS, County FIPS, Year, Day, Radiation, Maximum temperature (MaxTemp), Minimum temperature (MinTemp), and Precipitation. An example of attributes and a tuple is given in the table 5.1.

```
SELECT *
  RESTRICTED TO [[(C.MaxTemp + C.MinTemp)/2<0]]
FROM Climate C
```

Figure 5.7 ParaSQL query

Spatial/spatio-temporal queries are required to facilitate the driving of climatic results – possessing temporal aspect - from spatial/spatio-temporal datasets. The resultset produced by the query may be a subset of the original dataset. This research work is more interested in special kind of spatio-temporal queries - parametric queries, also known as ParaSQL queries (Noh, 2006; Gadia and Nair, 1993).

ParaSQL queries are useful to database community and consist of both the aspects-spatial and temporal. An example of ParaSQL query is given in Figure 5.7. This ParaSQL query, applied on NC-94 climatic dataset, results those days as output that possess the average temperature less than 0. In this work, it is assumed, the post execution spatial component is in GML format again and Geographical Information System (GIS) community (Franklin, 1992) is more interested to know about the intricate spatial and non-spatial attributes in much finer detail.

An attempt is made towards the development of a medium, where both - the database and GIS communities - can harness the capabilities of the system and that helps to bridge

the gap between database and GIS communities. If database community relies on the execution of the spatial or non-spatial queries to produces the results, GIS community is interested to provide visualization support for a better or full understanding of the resultant data, provided data is visualizable. A need is felt of visualization to explore and understand the intricate details about the geospatial component in any spatial or spatio-temporal artifact and this is the main focus of this work.

A software tool, called AutoConViz is developed to cater the spatial artifact of the spatial/spatio-temporal data by converting the original dataset into different format and helps in automatic visualization of it. AutoConViz takes spatial/spatio-temporal data as input in GML format and generates a shape file format of the corresponding data as output and renders it automatically to visualize what the shape of the spatial artifact of the data is in the AutoConViz window, equipped with the basic functionalities such as zooming, panning, and selection etc. If a user is further interested to do more intensive analysis on the data (shp format), the software leverages a way to open the shape file in ArcGIS (ESRI, 1990) from the current window with bare minimum efforts – on a single click. AutoConViz is a robust and efficient software tool and is validated successfully using various valid sample spatial dataset (in GML format), and but due to the space limitation, this paper could accommodate the validation only on two inputs - “places.gml” and “counties.gml” which are explained in significant detail later in the validation section.

The novelty comes from the fact that as soon as the shape file, from its counterpart gml file, is generated, a user can visualize its shape on-the-fly without a single bit of human intervention. Thus AutoConViz provides this additional functionality of visualization along with conversion. Upon visualization, a user can perform a few basic operations

such as zoom in and out, selection partially or fully, selection inversion etc, on the rendered shape file.

As far as functionality is concerned, AutoConViz works with conservative approach and offers enough basic functionality that can be useful to perform on the shape file for the mere visualization at the prima facie. If the user is further interested in the more intricate detail of that spatial dataset (in shape file format), AutoConViz, offers a bridge to navigate to fully functional classical ArGIS, which is reservoir of unlimited functionalities readily available to use.

ArcGIS (with its various flavors) from Environmental Systems Research Institute (Dangermond, 1969) is considered as one of the leading product for spatial visualization. An attempt is made to implement the visualization module of AutoConViz, using ArcGIS's arc object as underneath technology. This helps in seamless integration of new functionalities and incorporation with ArcGIS desktop in case of future development.

Thus the distinguish features of this research can be summarized as follows:

- On-the-fly visualization of spatial shape file format after conversion from corresponding input spatial GML format. This helps in fully understand the geometry of the spatial data.
- Conservative approach in functionalities exploration.
- Unified technology – arc object - in development for seamless integration with ArcGIS and easy future functionality extension.

Similar to other various visualization tools available today in literature, research or commercial arenas, AutoConViz can be an additional contribution to database and GIS communities.

5.3.2 Functional Architecture

This section describes the functional architecture of the software. The architecture is decomposed into subsections for better understanding.

(a) Large Spatio-Temporal Database. In the database world, the analysis of huge amount of data (Figure 5.8 (a)) not advisable from efficiency point of view, rather, interesting subset of the vast data is extracted to do interesting analysis. The smaller subset facilitates the desired analysis efficiently. There are numerous methodologies available to derive the smaller interesting subset from the vast chunk of the data in the database world. The explanation of those methodologies is out of the scope of this paper. In this work, I am interested in on such spatio-temporal dataset – NC-94 (NCRA, 2004) having spatial artifact in GML format. NC-94 dataset is a 30 years data consists of climate, crop, and soil data for North Central Region of United States. The well organization of spatial and temporal components of this NC-94 dataset draws our attention to further explore it. NC-94 dataset has the spatial and temporal granularities as county and day respectively.

(b) Spatio-Temporal Query. Spatial/ spatio-temporal query (Figure 5.8 (b)) facilitates the driving of results from spatial/ spatio-temporal datasets. The result set produced by the query may be a subset of the original dataset. In this work, I am more interested in special kind of spatio-temporal queries - parametric queries- also known as ParaSQL queries (Gadia and Nair, 1993). ParaSQL queries are useful for database community and

consist of both the aspects- spatial and temporal. Figure 5.7 is one such example of ParaSQL query applied to NC-94 climatic dataset (NCRA, 2004). The spatial and temporal footprints of the NC-94 dataset are county and day respectively. The dataset has various attributes (Table 1.1) and *MaxTemp* and *MinTemp* are two of them and used in the query. This ParaSQL query results those days as output that possesses the average temperature is less than 0. As stated earlier, the query results the spatial artifact in GML format, where GIS community is more interested to investigate the intricacy of spatial and non-spatial attributes in much finer detail.

(c) Spatial Artifact (GML format). This block consists of two artifacts: 1) Spatial (in GML format) and 2) Non-spatial. The latter is the adjacent peer small block (on the right) of this spatial artifact which is a set of tuples comprising the temporal aspects of the NC-94 dataset. The further use of these tuples is beyond the scope of this chapter, rather the spatial artifact (in GML format) is considered for the further operations. In order to start functioning, AutoConViz relies on a GML file as input. Any valid spatial artifact as GML file is supplied as input to AutoConViz (Figure 5.8 (c)) which produces the corresponding shape file format as output. The readers, interested to know more about GML can find voluminous information in literature such as Wikipedia on GML (OGC, 2000).

(d) GML - Shp Converter. This is one of the two main import components of AutoConViz (Figure 5.8(d)). This piece of functionality can be explained as- it takes any valid spatial artifact as GML file (e.g. xyx.gml) as input and produces the corresponding

shape file format (xyz.shp) along with the necessary other files – dbf and shx version of xyz (ESRI, 1990). Which are briefly explained in following section 5.8(e).

(e) Spatial Artifact (Shape file format). While producing the shape file format (e.g. xyz.shp) as output, AutoConViz produces, two more files also - xyz.dbf, and xyz.shx (ESRI, 1990) - along with that (Figure 5.8 (e)). The shape file (shp) consists of the geometrical part of that GML file. The dBase format (dbf) file contains the attributes for each geometry in shape file. The geometry in dbf is recognized by a unique identifier - FIPS (Federal Information Processing Standard) code (USFG, 1994). The shape index file (shx) preserves the positional index of the features in the geometry of the GML file. It basically provides the linking between the geometrical artifacts in the shape file (shp) to their respective non-geometrical artifacts in dbf file.

(f) Display Automate. As shown in figure 5.8(f), this is the additional functionality, AutoConViz offers along with conversion. It exploits the produced shape file (shp) as input and renders it in ArcGIS window with limited functionality (Figure 5.8(g)). The objective is to visualize and analyze the generated shape file format graphically to make more clear understanding about the spatial dataset. If the visualization of that shape file format introduces more curiosity to user, AutoConViz, offers navigation to the classical ArcGIS (Figure 5.8(h)) on a cost of one single button click, provided ArcGIS is preinstalled in the machine on which AutoConViz is running.

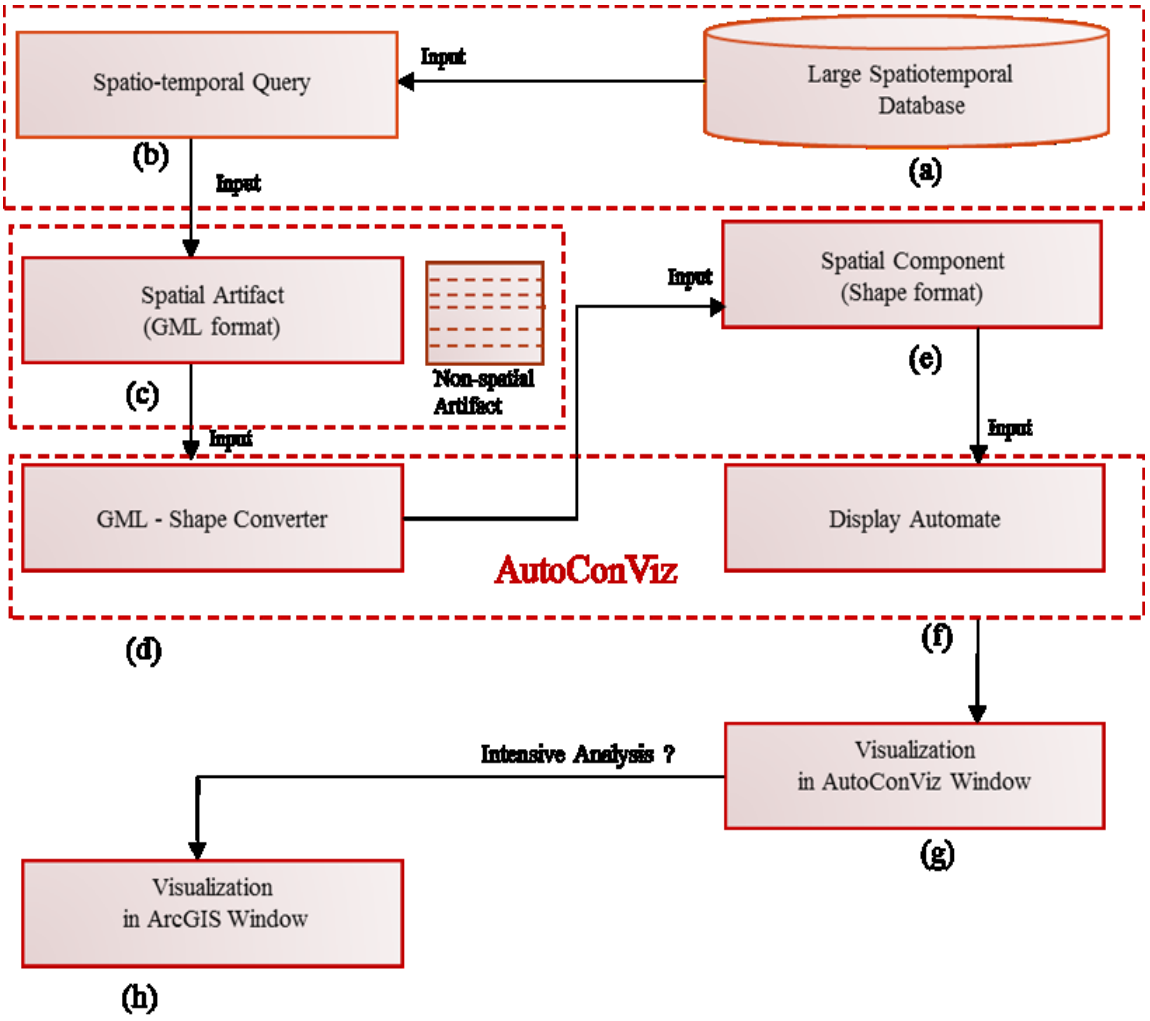

Figure 5.8 Functional architecture of AutoConViz

5.3.3 Prototype Implementation

The implementation of the software is divided into phases and all of them are explained below in sufficient detail.

A. Software and hardware requirement. The requirement of the software and hardware resources is the building block of any software development and considered as

one of the most critical stages. An intensive literature study is required to garner the knowledge about the suitable resources – both software and hardware. Following the same pattern, an intensive literature review of the resource documentation provided by ESRI (Dangermond, 1969) is done and it is found the following **software** resources needful to this research work.

- ArcGIS Runtime Engine 9.3
- ArcGIS Map 9.3
- ArcGIS Developer's Kit 9.3
- Eclipse Helios IDE (Integrated development environment)
- JDK 1.6

Hardware requirements comprise, a computer system- equipped with CPU, monitor and mouse. Though, operating system is not the limitation, but windows XP is used for this development.

B. Software development. To facilitate the software development in an organized way, Waterfall model (figure 4.10) is practiced up to a large extent in AutoConViz construction. It is worth to mention here clearly that the team size is three and most of the phases of this model are accommodated. The words *phase* and *stage* are interchangeably used in the context of waterfall model. The emphasis is given only on those stages of Waterfall model which are relevant to this work and are briefly described below.

I. Requirements. This phase broadly explore the needs regarding visualization persisting in database and GIS communities and required to be fulfilled soon. An intensive research is conducted about the existing software useful to handle similar kind of issues, and literature review and progressive brainstorming sessions are held to unify the direction about inputs and outputs expected from the constructed software.

In a nutshell, for AutoConViz, the input should a spatial/spatio-temporal data in GML format. The software system should generate a shape file format of the corresponding data as output and renders it in a window automatically to visualize what the shape of the data is. The window should be equipped with the basic functionalities such as zoom-in, zoom-out etc. If a user is further interested to do more analysis on the data (shp format), the software should leverage a way to open the shape file in ArcGIS map from the current window with bare minimum efforts by the user such as a single click. The software should save a copy of the generated shape file.

II. Design. A good design leads to good software product. After the successful completion of the stage 1, and passing through a rigorous thought process, a design layout plan is formulated both from theoretical and technical perspectives. At the abstract level, functionality is classified into modules to be engineered independently. Based on the functionality sought from AutoConViz, it turns out that there are two modules: *GML-Shape Converter* and *Display Automate (section 4.3.2)*, which can be developed independently and integrated later on to offer full functionality. All the three members of the team participate in this phase.

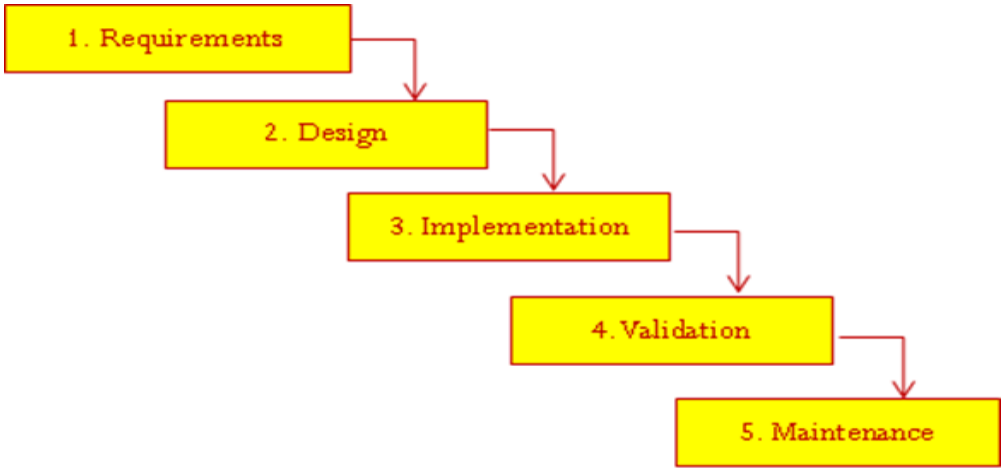

Figure 5.10 Waterfall model used in AutoConViz software development

III. Implementation. Once the modular design of the software is laid down, the actual code implementation in a computer programming language such as C, C++, Java etc., is started in this stage. The team of the engineers is divided into groups and different modules are assigned to them to develop. Based on the complexity of the module and size of the engineer's team, the implementation time varies.

The implementation (team size one) of AutoConViz is done in Java programming language. Both the modules are implemented independently. Underneath the implementation to develop the module: *GML-Shape Converter,* the java archive (jar files) available from http://www.deegree.org/ (Greve et al., 2001) is used. The implementation of visualization module: *Display Automate* is done using the java archive (*arcObject.jar*) from esri.com (Dangermond, 1969) as the required file.

IV. Validation. Under the software development standard, a dedicated team of tester (quality assurance) is assigned to test the functionality of the software implemented. For

complex softwares, the testing team can be divided into groups, testing the functionality at module level. Any error or bug is reported and assigned back to the responsible individual developer. The intensive validation of AutoConViz is described in section 4.4 in greater detail.

V. Maintenance. This phase is important and needed to ensure that the developed software system continue to perform with the same efficiency as desired. Unfortunately, this phase is not accommodated this this work.

5.3.4 Algorithm

ALGORITHM: Automatic conversation from GML to Shp and visualization

1. **BEGIN**
2. Spatio-temporal query runs on a large Spatio-temporal dataset *x* produces subset *y*
3. Spatial component of *y* is supplied as input to AutoConViz
4. AutoConViz produces corresponding data *z* as shape file format and stores it in a directory
5. AutoConViz reads that shape file and open visualization in a window
6. **IF** user needs more analysis
7. On click on ArcGIS Viz
8. Open new window of ArcGIS preloaded with the same shape file as layer
9. Do more analysis
10. **ENDIF**
11. **END**

This section describes the pseudocode for the functionality of AutoConViz and each step as self-explanatory.

5.3.5 Validation

In software development, a set of test beds (of size one or more) is required in order to validate the software functionality intensively. A good quality of software should be generic enough to attend various test cases of variety of inputs of same kind. AutoConViz is a robust and efficient software tool. Though AutoConViz has been validated successfully using various valid sample gml files, due to the space limitation of this paper, snapshots for only two inputs – “places.gml” and “counties.gml” discussed here.

places.gml and counties.gml

```
<gml:featureMember>
    <ogr:placept fid="F0">
      <ogr:geometryProperty>
                  <gml:Point>
                        <gml:coordinates>
                              -56.315090179443359,52.516880035400391,0
                        </gml:coordinates>
                  </gml:Point>
      </ogr:geometryProperty>
      <ogr:AREA>0.000</ogr:AREA>
      <ogr:PERIMETER>0.000</ogr:PERIMETER>
      <ogr:PACEL_>1</ogr:PACEL_>
      <ogr:PACEL_ID>1</ogr:PACEL_ID>
      <ogr:NAME>Port Hope Simpson</ogr:NAME>
      <ogr:REG_CODE>10</ogr:REG_CODE>
      <ogr:NTS50>013A09</ogr:NTS50>
      <ogr:LAT>523300</ogr:LAT>
      <ogr:LONG>561800</ogr:LONG>
      <ogr:POP91>614</ogr:POP91>
      <ogr:SGC_CODE>1010009</ogr:SGC_CODE>
      <ogr:CAPITAL>0</ogr:CAPITAL>
      <ogr:POP_RANGE>2</ogr:POP_RANGE>
    </ogr:placept>
  </gml:featureMember>
```

Figure 5.11 (a) Snapshot of a fraction of places.gml

```
<gml:featureMember>
 <ogr:counties fid="F0">
  <ogr:geometryProperty>
   <gml:MultiPolygon>
        <gml:polygonMember>
         <gml:Polygon>
          <gml:outerBoundaryIs>
           <gml:LinearRing>
                <gml:coordinates>
                -90.963155,46.961962999999997,
                 46.961962999999997 -90.964322999999993,
                  ...       ...       ...

                 46.157646999999997 -91.551282999999998,
                -90.963155,46.961962999999997
           </gml:coordinates>
          </gml:LinearRing>
         </gml:outerBoundaryIs>
        </gml:Polygon>
       </gml:polygonMember>
      </gml:MultiPolygon>
     </ogr:geometryProperty>
    </ogr:counties>
</gml:featureMember>
```

Figure 5.11(b) Snapshot of a fraction of counties.gml

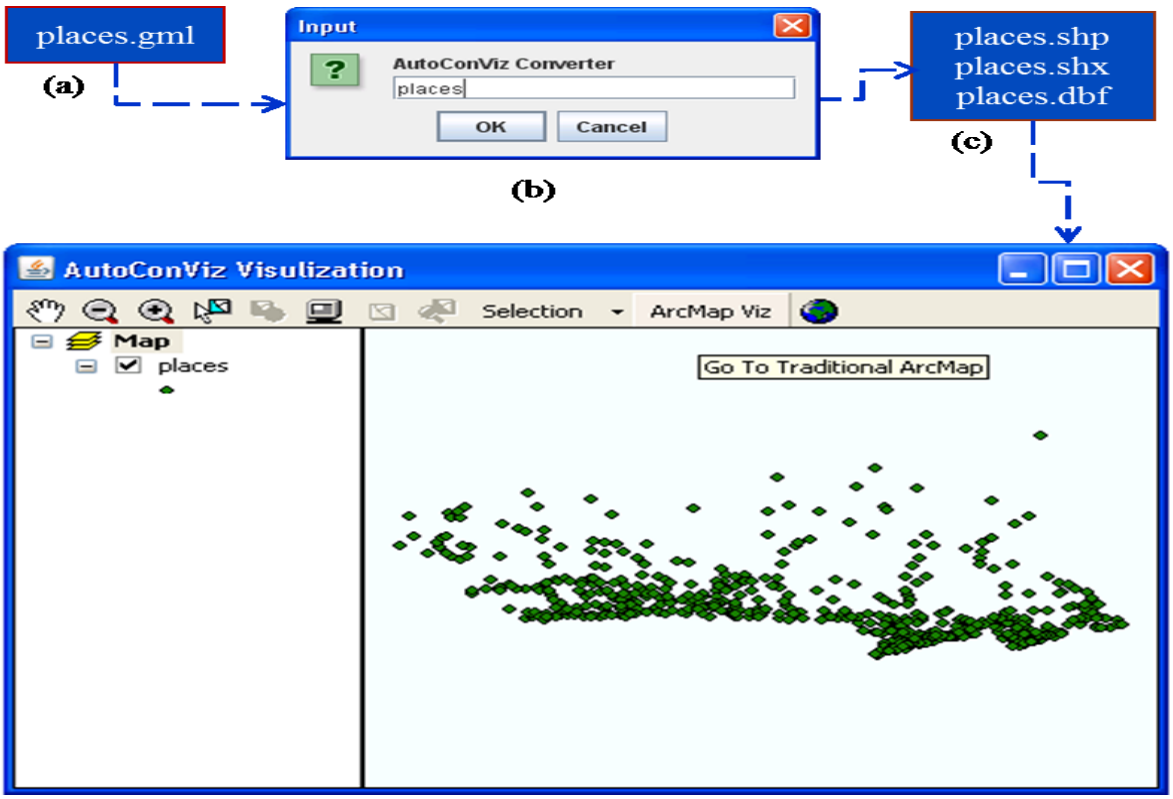

Figure 5.12(a) Interface for automatic visualization of places.shp

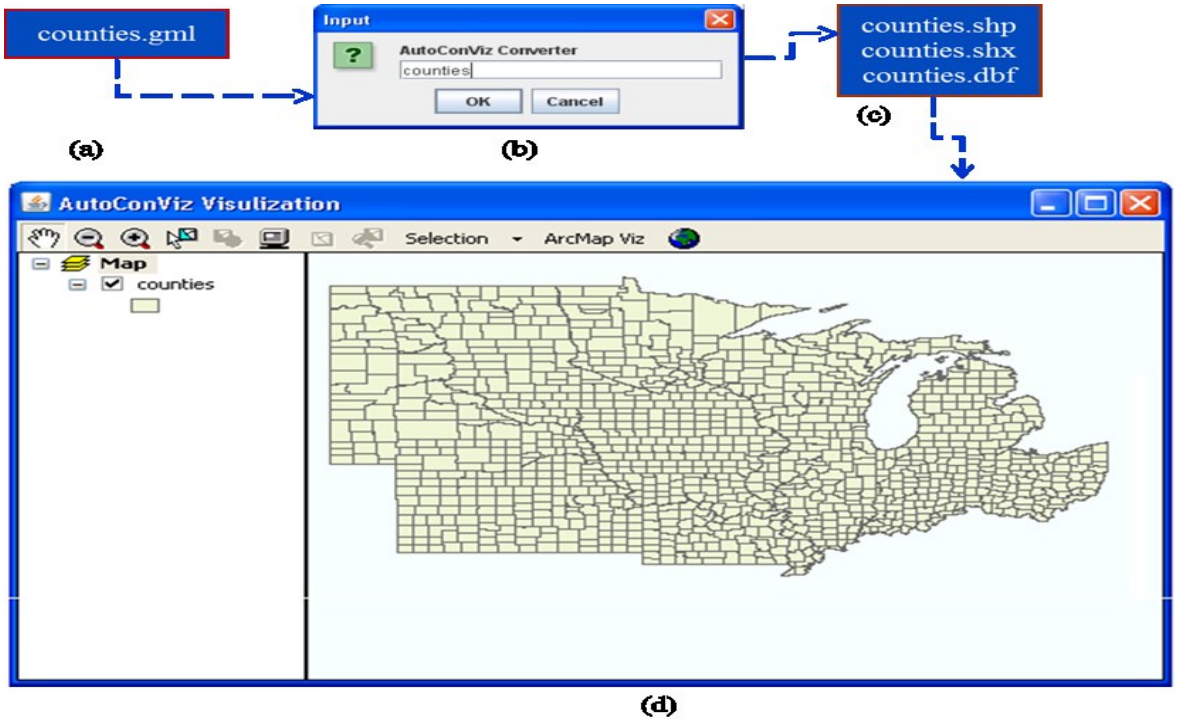

Figure 5.12(b) Interface for automatic visualization of counties.shp

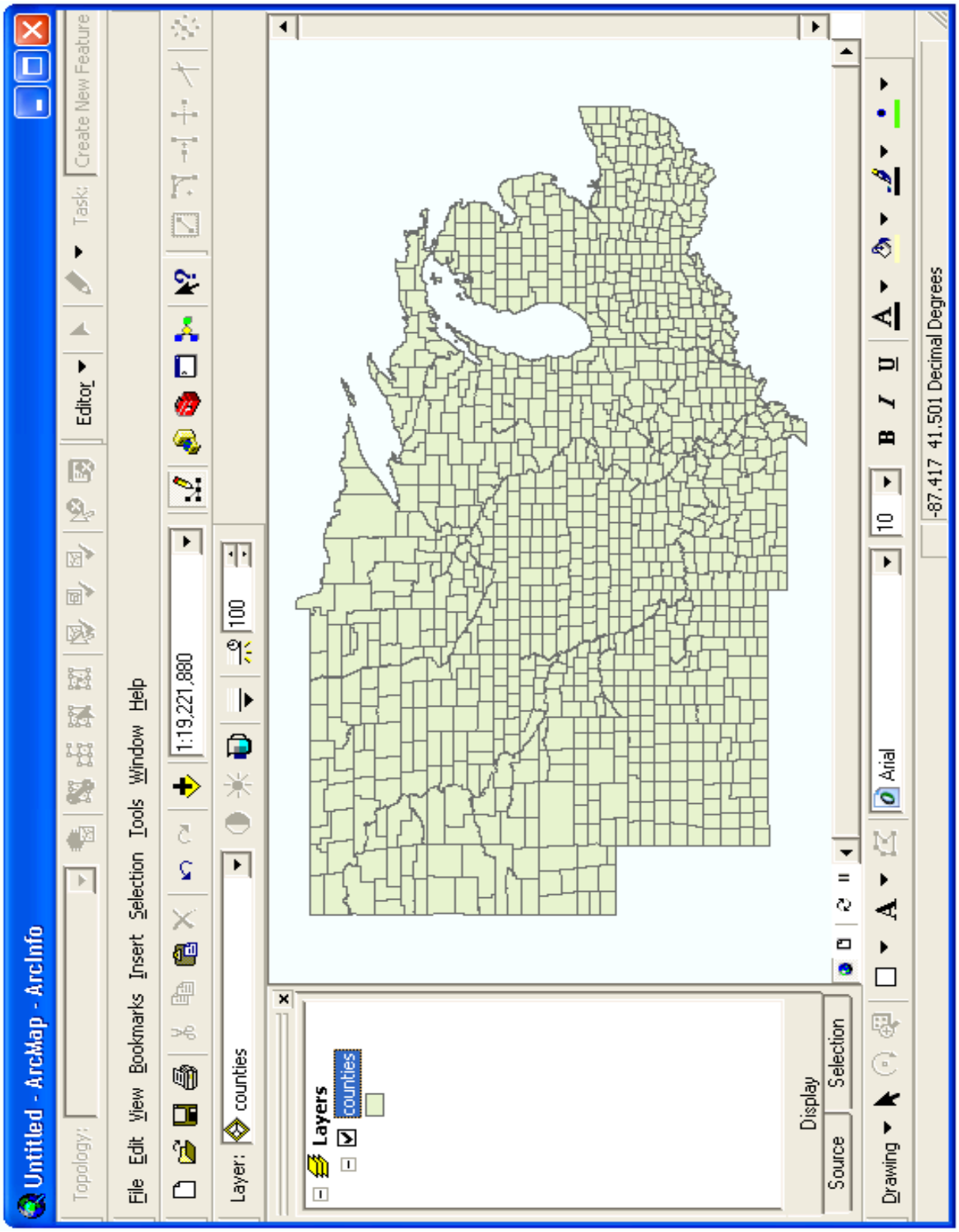

Figure 5.13 Visualization of counties.shp in ArcGIS

Due to the space limitation, only one feature member of each GML file is depicted in the snapshot of figures 5.11(a) and 5.11(b), with the 'geometry' property defined herein as a point and a polygon respectively.

Figures 5.12(a) and 5.12(b) are the automatic visualization of the converted shape files – "places.shp" and "counties.shp" respectively. As shown in the figures, the generated shape file for the former is a set of points whereas that for the latter is set of polygons. Though, in the *AutoConViz Visualization* window a user can perform a few operations on the rendered shape file such as panning, zooming, selection, full extent and navigation. The pan button – represented by hand - helps to move the displayed file anywhere inside the whole window. Zoom-in and out are self-descriptive operations. Selection button helps to select the displayed file partially or holistically. This command offers the opportunity – 1) to switch the selection in case of partially selection of the shape file image, 2) to zoom-in the selected features, 3) to clear the selection. Full extent command helps to bring the zoomed in or out image to the original size. A new command – *ArcMap Viz* – is introduced in the display window that navigates from this window to a traditional (classical) ArcGIS window. This feature facilities interested users to do more analysis with the same shape file (layer). On clicking this button (*ArcMap Viz*), the classical ArcGIS window opens up, pre-loaded with the same shape file (Figure 5.13). Reader is encouraged to visit ESRI website (Dangermond, 1969), if wishes to learn more about classical ArcGIS.

5.3.6 Efficiency

In software engineering, various factors are calculated to find out the efficiency of a software system, time is one of them and is of main emphasis in this section. The efficiency of AutoConViz is evaluated based on the time taken to visualize the spatial component (in shape file format). The visualization time of AutoConViz is compared with ESRI ArcGIS for both the sample spatial components (shape files) – "counties.shp", "places.shp" - used in this research work. Both the shape files are of relatively different sizes; the size of counties shape file is - **7.29 MB** and that of "places.shp" is **16.0 KB**. For AutoConViz, the conversion time of spatial component from gml format to shape file format is excluded.

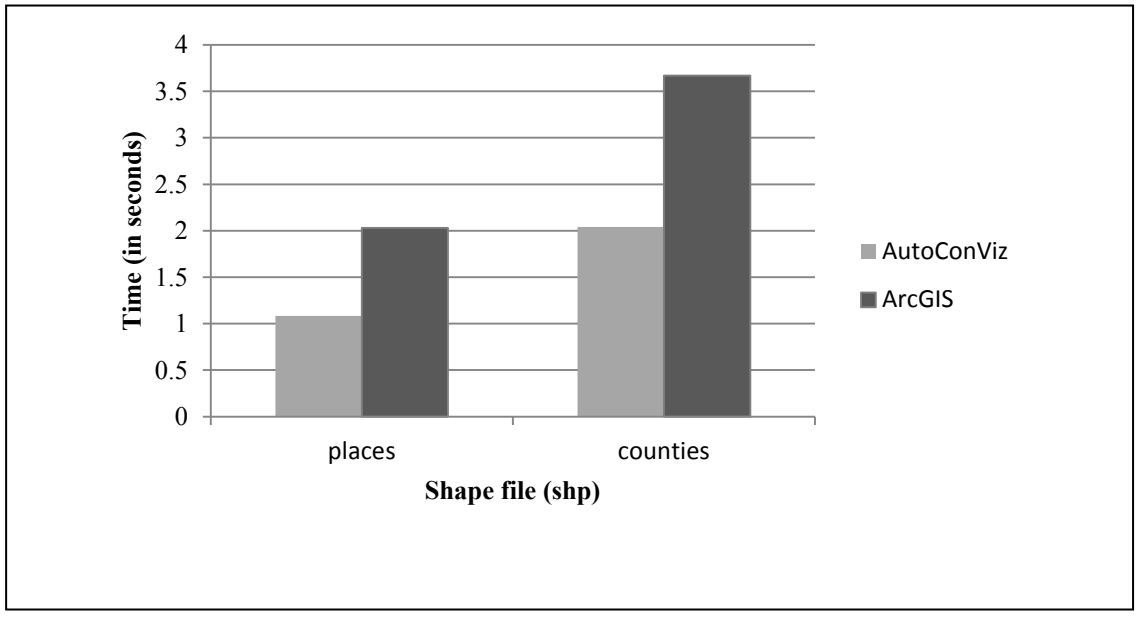

Figure 5.14 Time efficiency of AutoConViz vs. ArcGIS

The bar graph in figure 5.14 depicts the comparison between AutoConViz and ArcGIS in term of time efficiency. A time logger is used to capture the time consumed in

visualization for both the shape files, exploited in this research work. Below, the table 5.2 represents the concrete values of visualization time taken by AutoConViz and ArcGIS to render the shape files.

Table 5.2 Visualization time representation

S. No.	Name of file	Size of file	Visualization time	
			AutoConViz	ArcGIS
1	places.shp	16.0 KB	1.15 sec	2.05 sec
2	counties.shp	7.29 MB	2.09 sec	3.67 sec

Based on the statistics in table 5.2, the visualization time (say TViz) taken by AutoConviz as compared to ArcGIS is calculated for both the shape files as follows.

$$TViz_{\textbf{places.shp}} = \frac{2.05 - 1.15}{2.05} * 100 = 43.90\ \%$$

$$TViz_{\textbf{counties.shp}} = \frac{3.67 - 2.09}{3.67} * 100 = 43.05\ \%$$

Thus, it can be said that for these two sample shape files, AutoConViz takes almost 43% less time compared to ArcGIS in order to visualize a shape file. In both the case, it can be observed that AutoConViz turns out to be more time efficient.

In the Parametric Data Model, it is observed that the visualization of an attribute is a terminal visualization where no more functionality is possible except manual analysis. A minute change in the output needs either the whole process to be repeated or lots of implementation efforts are required with special skills which are not easy to possess for everybody. Thus, the visualization in the Parametric Data Model has limitations which are necessary to be addressed in order to have exploratory visualization. A consideration is given to overcome these visualization's limitations and after the rigorous thought process, it is expected that, AutoConViz can absorb the visualization limitations from the Parametric Data Model.

5.4 GEO-SPATIAL PATTERN VISUALIZATION AND ANALYSIS FOR SNAP

5.4.1 Overview of Supplemental Nutrition Assistance Program (SNAP)

Since a long period of time, USDA is running a food aid program to low or no-income population in United States. The program is widely known as FSP (even today) despite it is renamed as SNAP in last few years. In the earlier years, the qualified population used to receive paper stamp or coupons to buy prepackaged food. But in 21th century, in the world of computing, the paper stamp/coupons have been replaced by Electronic Benefit Transfer (EBT) card to distribute all the food-stamp benefits. Using the card, any edible food but prepackaged, (potential restriction) can be purchased. The food may include soft drinks and confectionery, and need not necessarily to be rich in nutrition factors. EBT card is a specialized debit card system used in late 1900s and issuance was administered

by private contractor for public-assistance welfare programs as well. The successful replacement of all paper stamps/ coupons by EBT cards, motivated the renaming the FSP as SNAP in October 2008.

In order to avail the SNAP benefits, the recipients must barely have near-poverty income. According to the June 2009 statistics, per person average monthly benefit was $133.12. The statistics also revealed the facts about the highest number of US population (39.68 million, in Feb 2010) sought SNAP benefits since 1960. In the section, the SNAP elaboration is subsectionized to deliver better understanding to the reader.

A. Who can qualify for SNAP

Under SNAP benefits, USDA helps in buying the healthy food to low income households where the time granularity is month. USDA sets the laws and regulations that determine - 1) the eligibility criteria, 2) the limit on income and resource ownership and 3) the maximum benefits per household size. Based on the regulations, one should meet one of the following criterions strongly to quality for the SNAP benefits- i) disabled or elderly living on a small income, ii) low wages work, iii) completely unemployed or part-time work, and iv) receiving no other assistance payments.

B. Application process for SNAP

Though, at the federal level, SNAP is administered by USDA, the application process at the state level may vary. This work explains the application process exclusively for Iowa

only as this region is the spatial interest in this study. Iowa Department of Human Services (Iowa DHS) works at county level that entertains the SNAP application from its households and facilitates the SNAP benefits. This helps a household to find a nearest DHS office to ease the whole process. For example, a SNAP applicant in Story County may approach to the nearest DSH, Story County Department of Human Services. DSH provides online application system and required documents can be submitted to local office. Alternatively, an application for SNAP benefits can be requested by phone, fax, or mail from the nearest DHS office or can be fetched in person. The total household gross monthly income should be less than the one, specified in regulations. For the whole household, under penalty of perjury a senior adult or authorized representative can sign the complete application. Upon receiving the application, the local DHS sets up an interview to verify the eligibility for SNAP benefits. There may be set of other required documents (please see below) that are needed to bring to the interview. Once the SNAP eligibility is verified, within thirty days of application submission, the households start to get benefits.

C. Required documents for interview. Though there may additional requirement of the documents, few of them are given below.

- Income and expenses verification
- Identity proof e.g. driver's license
- Social Security Numbers (SSN) -for all household members
- Bank statements
- Shelter costs statement

- Rent or mortgage payment
- Utility bills - heat, electricity, water/sewage/garbage, telephone, etc.

D. Household status changes. Households availing the SNAP benefits must report to local DHS office – 1) if the household income meets the required income level for the household size, 2) any changes in address to avoid problems. In case, if a household does not report the change and the extra SNAP benefits will have to be returned. The authorized person may be prosecuted up to severe penalty such as jailed in addition to repayment of the amount.

5.4.2 Snap Eligibility Calculation

SNAP, previously known as Food Stamp Program (FSP), administrated by the U.S. Department of Agriculture (Vilsack, 2012) is helping the low income or no income population since a long period of time. USDA coordinates with each state, which eventually distributes of the SNAP benefits to their eligible population. Earlier, FSP used to use colored paper stamps or coupons- brown, blue, and green – each carrying different price value which is one, five and ten dollars respectively. In today's modern world colored paper stamp/coupons are replaced by the cards to buy any food such as soft drinks and confectionery under the food-stamp program. There is no constraint whether to buy nutrition food or not but the food should be prepackaged. This study targets Iowa as the region of interest per-county basis, where the SNAP eligibility of the population is analyzed based on the fact that the population is 50% or more low income (185% of

federal poverty level). In the study, census tracts are considered as the finest granularity spatially where income data is easily available. The data is derived from two different online resources - 1) US Census Bureau, 2) ESRI (Dangermond, 1969). The downloaded data is the raw data and requires synthesis and analysis before being used further (Table 5.3).

Assuming that N is the total number of population to determine the poverty status and N_x is the total number of population with ratio of income to poverty level x, then

$N <_{1.85}$ (total population with ratio to poverty level less 1.85) $= \sum_{i=.50}^{1.75\ to\ 1.84} N_i$.

$\%\ N_{<1.85}$ (percentage of population with ratio to poverty level 1.85) $= \left(\frac{N_{<1.85}}{N}\right) * 100$.

Table 5.3 Collected data for SNAP (excel format)

Geography Identifier	Total population	Ratio of income to poverty level; Under .50	Ratio of income to poverty level; .50 to .74	Ratio of income to poverty level; .75 to .99	Ratio of income to poverty level; 1.00 to 1.24	Ratio of income to poverty level; 1.25 to 1.49	Ratio of income to poverty level; 1.50 to 1.74	Ratio of income to poverty level; 1.75 to 1.84	Total Population with ratio less than 1.85	% of population with ratio less than 1.85
19169000100	7086	64	145	31	105	110	160	46	661	9.328252893
19169000200	3696	124	66	31	74	109	95	50	549	14.8538961
19169000300	3284	169	87	116	82	141	39	24	658	20.0365408
19169000400	3036	207	38	92	69	66	87	41	600	19.76284585
19169000500	1426	191	144	67	153	192	156	32	935	65.56802244
19169000600	4621	300	254	188	154	224	153	39	1312	28.39212292
19169000700	3281	893	359	196	180	282	181	25	2116	64.49253276
19169000800	23	0	0	0	0	0	0	0	0	0
19169000900	4115	277	182	140	205	247	142	67	1260	30.61968408
19169001000	4135	575	500	374	361	205	333	167	2515	60.82224909
19169001100	3601	620	376	272	344	297	99	28	2036	56.53985004
19169001200	59	0	0	0	5	5	0	0	10	16.94915254
19169001301	4755	629	392	329	197	243	303	103	2196	46.1829653
19169001302	3561	122	64	73	79	48	122	26	534	14.9957877
19169010100	5986	97	88	118	103	155	183	87	831	13.88239225
19169010200	3964	70	56	96	127	116	82	73	620	15.6407669
19169010300	3996	91	70	65	80	118	120	66	610	15.26526527
19169010400	2881	72	42	29	155	44	122	32	496	17.21624436
19169010500	1961	27	30	24	52	125	146	39	443	22.59051504
19169010600	5087	77	74	108	99	199	155	116	828	16.27678396

The tracts can be classified on the basis of percentage of people who qualify for the SNAP program. The data is classified into two groups; less than 50% and 50% or above. Based on this classification the tracts in a county are plotted with two different visualization patterns - the SNAP qualified tracts are plotted by angled hatched lines.

5.4.3 Snap Eligible Visual Patterns

As stated above, this study considers Iowa as the region of interest to determine the spatial pattern eligibility for SNAP. The experiments are run on per county basis and each county is analyzed at the tract level. Out of all 99 counties in Iowa, a few of those counties are visualized using ArcMap, having at least one tract where population is SNAP eligible.

For the Black Hack county (figure 5.15 (f)) the tracts - 1, 3, 5, 9, and 23.02 are the qualified tracts. County Dubuque (FIPS-19061) has only one qualified tract (tract 1). Johnson county (figure 5.15 (c)) has three tracts – 11, 16, 21- eligible for SNAP program. Linn county (FIPS-19113) has two tracts - 21, and 27 eligible for SNAP. Polk county (figure 5.15(a)) has five qualified tracts -12, 26, 50, 51, and 52. Scott county (figure (5.15 (d)) has five tracts of low-income population, qualified to avail SNAP benefits. In Story county (figure 5.15(b)) the results are very interesting. There are four qualified tracts: 5, 7, 10, and 11. It can be noticed that tract 12 has rich population, but completely surrounded by tract 11- low income population. There is only one tract: 9605 in which the population is low-income/ no-income and qualified for SNAP program in Wapello

county with FIPS-19179. Webster county (FIPS -19187) also has only one qualified tract - tract Id as 7. Woodbury (figure 5.15 (e)) has four qualified tracts: 12, 15, 16, and 17.

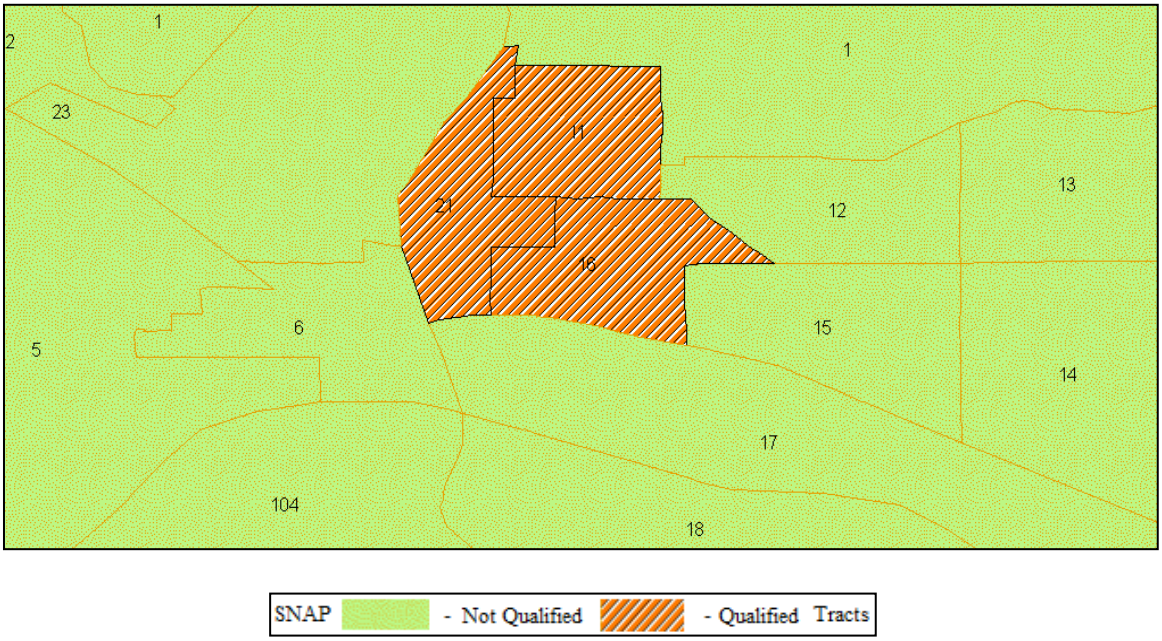

Figure 5.15 (a) POLK county (FIPS: 19153)

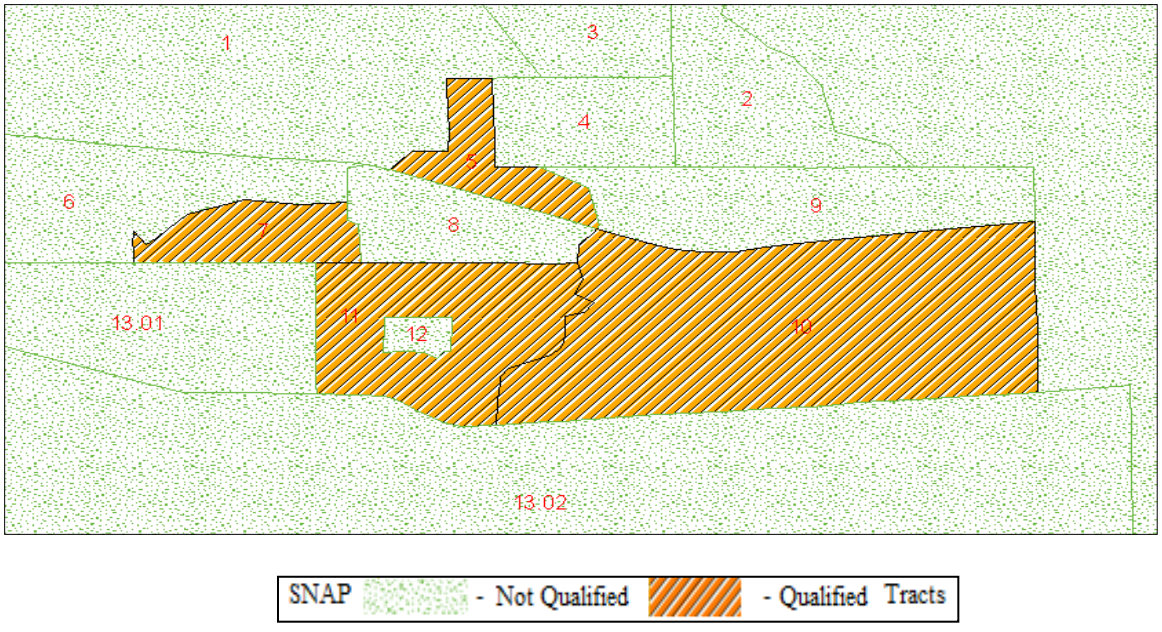

Figure 5.15 (b) STORY county (FIPS: 19169)

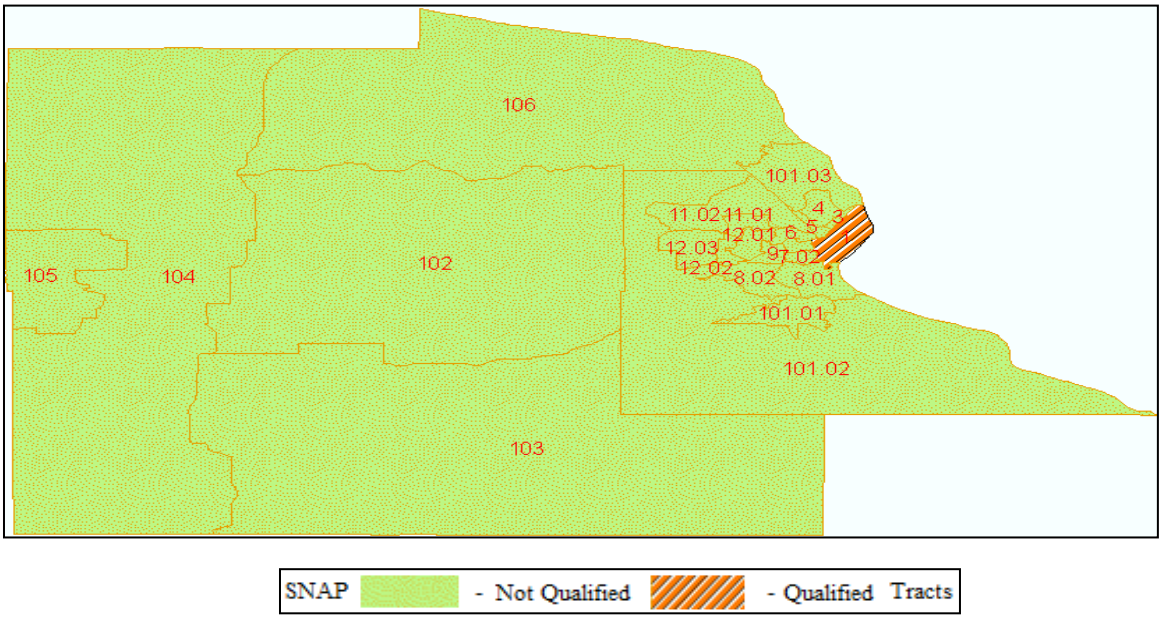

Figure 5.15 (c) JOHNSON county (FIPS: 19103)

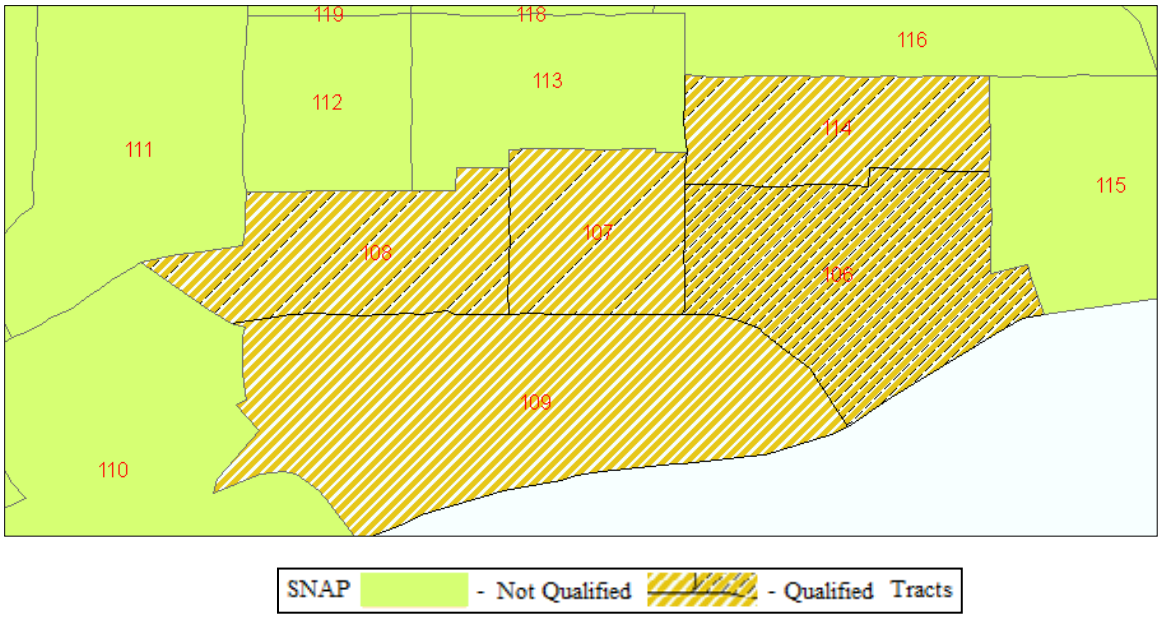

Figure 5.15 (d) SCOTT county (FIPS: 19163)

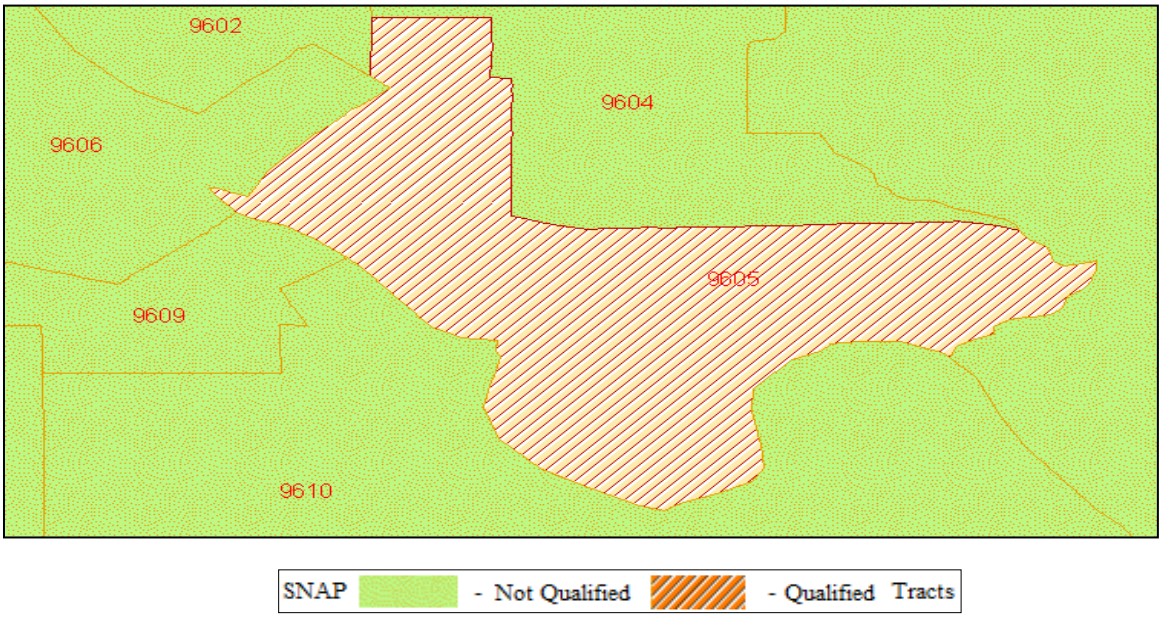

Figure 5.15 (e) WOODBURY county (FIPS: 19193)

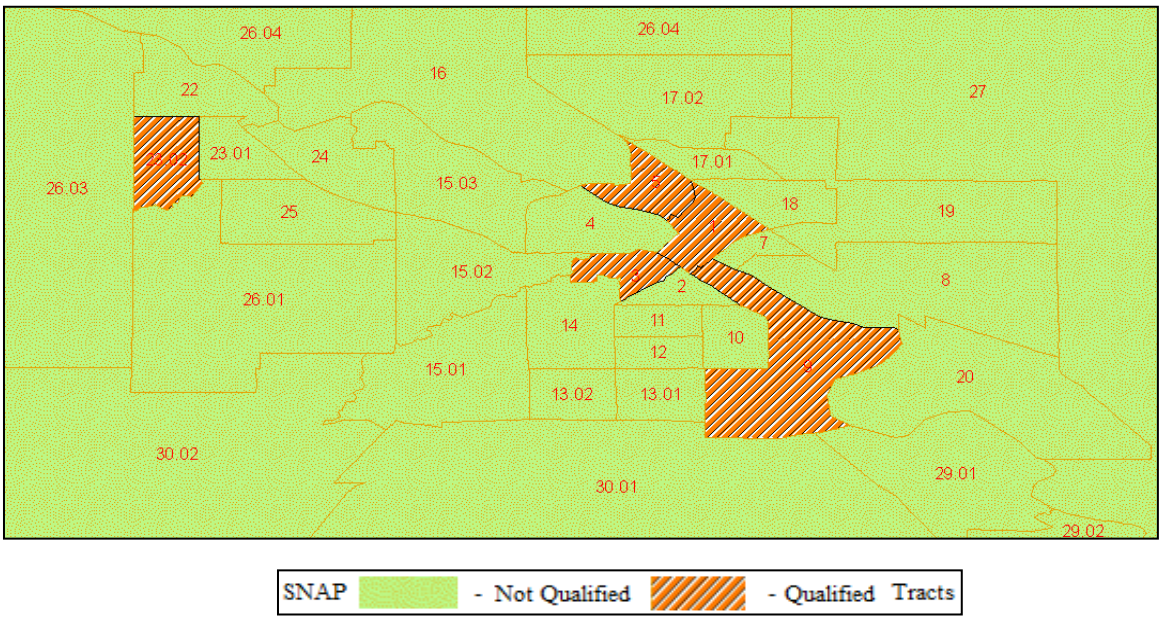

Figure 5.15 (f) Black Hawk county (FIPS: 19013)

5.4.4 Does SNAP Link to Racial or Ethnic Profiling

The research work is not completed on finding the SNAP qualified tracts only, rather expanded further to explore the socioeconomic aspects, keeping the qualifying tracts as base.

In the United States, there is a widespread stereotype that only minorities need SNAP - popularly known as food stamp - support. A remark from a republican US presidential candidate on Feb 25th, 2012 also links the food stamp to minorities only (http://www.cbsnews.com/video/watch/?id=7400038n). This work is intended to examine the hypothesis that- in the United States, only minorities avail SNAP benefits. State of Iowa is employed as the region of study and the census tract data is considered as the finest spatial granularity as the income data for individual participant is unavailable. GIS technology helps in differentiating SNAP eligible tracts from unqualified ones and displays the SNAP eligible diversity.

Hypothesis. In the United Stated, only the minority section of the population gets benefits from Supplemental Nutrition Assistant Program (SNAP).

In the study, once a SNAP qualified tract is found, the tract is further analyzed for the socioeconomic diversity. The total population in the tract is divided mainly into four classes based on the races: (1) White American, (2) African American, (3) Hispanic, and (4) Others. The last class, Others, includes all the other races except the above mentioned rest three, with a majority of Asians. In a tract, the total population of individual class is

divided by the total population of the tract in order to get the percentage (%) of population of that class and with the help of ArcMap, pie charts are drawn to represent that % of the socioeconomic diversity in the tract.

In this research work, the hypothesis - In the United States only minorities avail SNAP benefits- is examined. It is already mentioned, the population in divided into four diverse categories: (1) White, (2) African-American, (3) Hispanic, and (4) Others. The category – Others, is mainly dominated by Asian population live in the region. The same racial/ ethnic classification is used to verify the hypothesis. The results are collected county-basis, and tract in every county is considered as the smallest spatial granularity. It is previously mentioned, the study is conducted in all ninety-nine counties in the state of Iowa only.

Validation of Hypothesis. The total population is segregated into four sections: White, Black, Hispanic, and Others. The category – Others, circumvents mainly a majority of the Asian populous. As per the US federal poverty level, a tract is SNAP qualified if it reflects atleast 50% low income population. The hypothesis is validated by contradiction and has the following basis- if a SNAP qualified tract exhibits more than 50% of White populous occupancy in the entire tract; this indicates at-least a fraction of majority section also is nourished by SNAP and this contradicts the hypothesis.

Figures 5.16 (a) – 5.16 (c) are the pictorial representations, depicting the socioeconomic SNAP distribution to the population residing in three different SNAP qualified counties - Black Halk, Linn and Polk respectively.

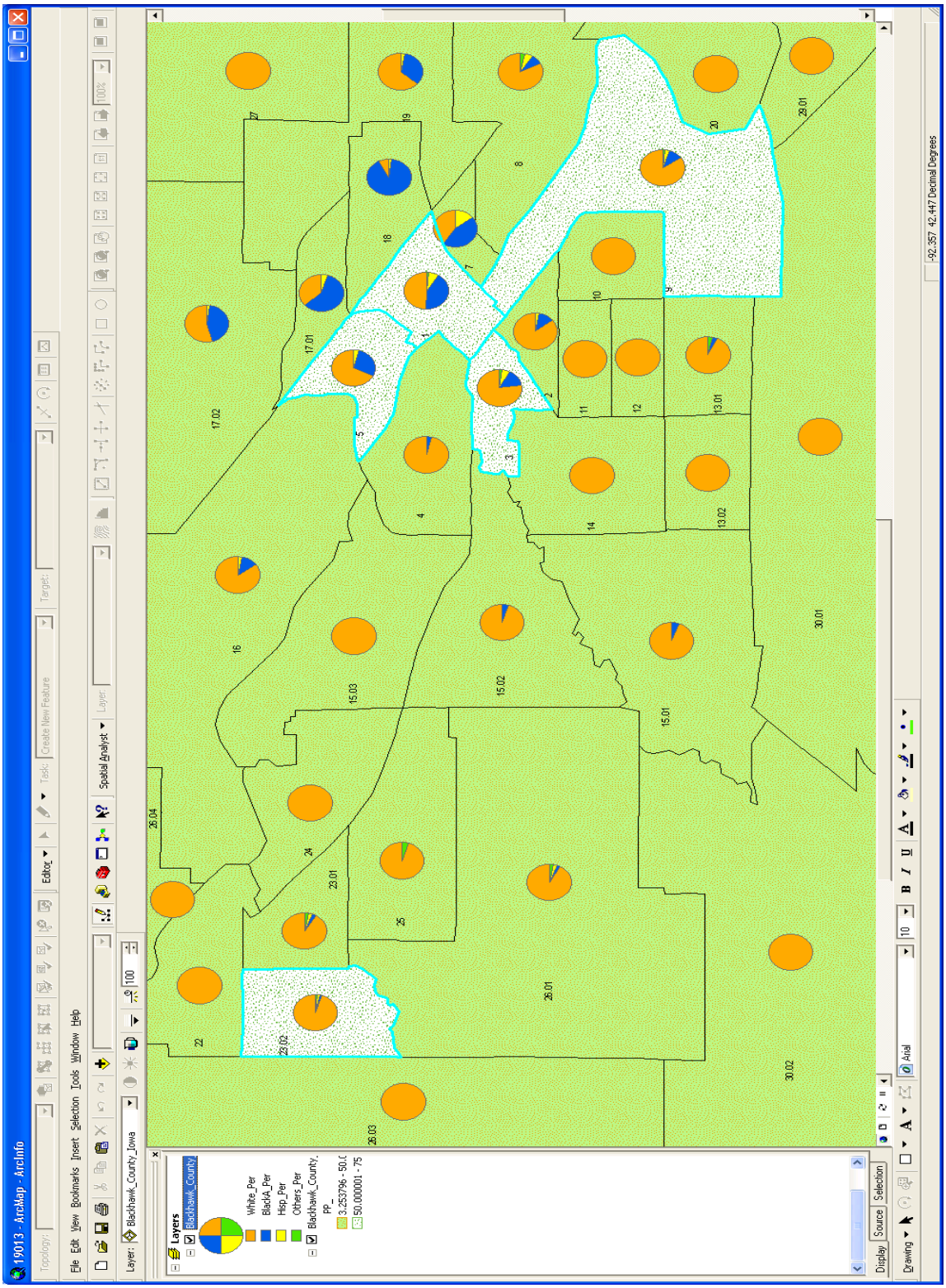

Figure 5.16 (a): Black Hawk county

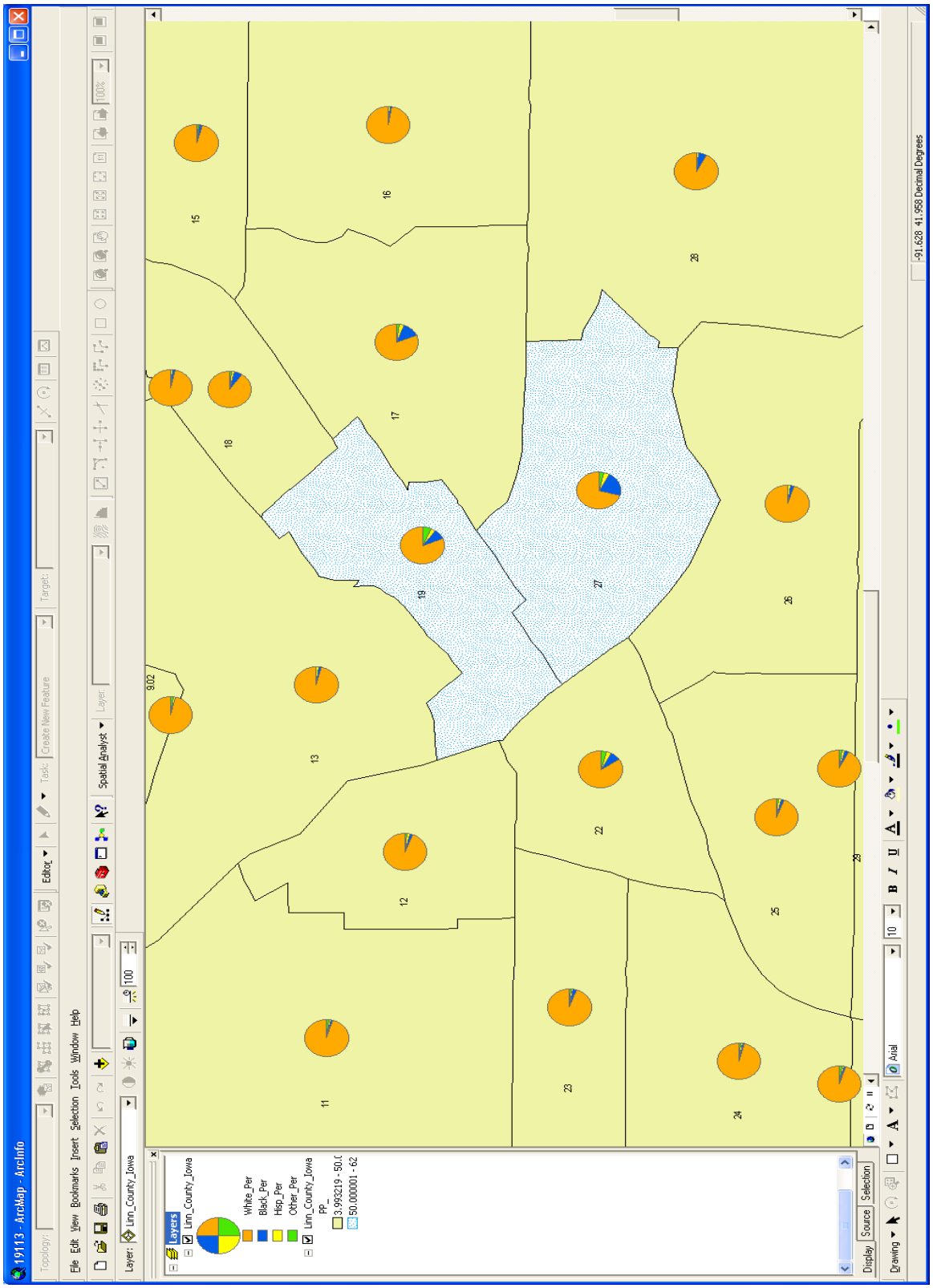

Figure 5.16(b): Linn county

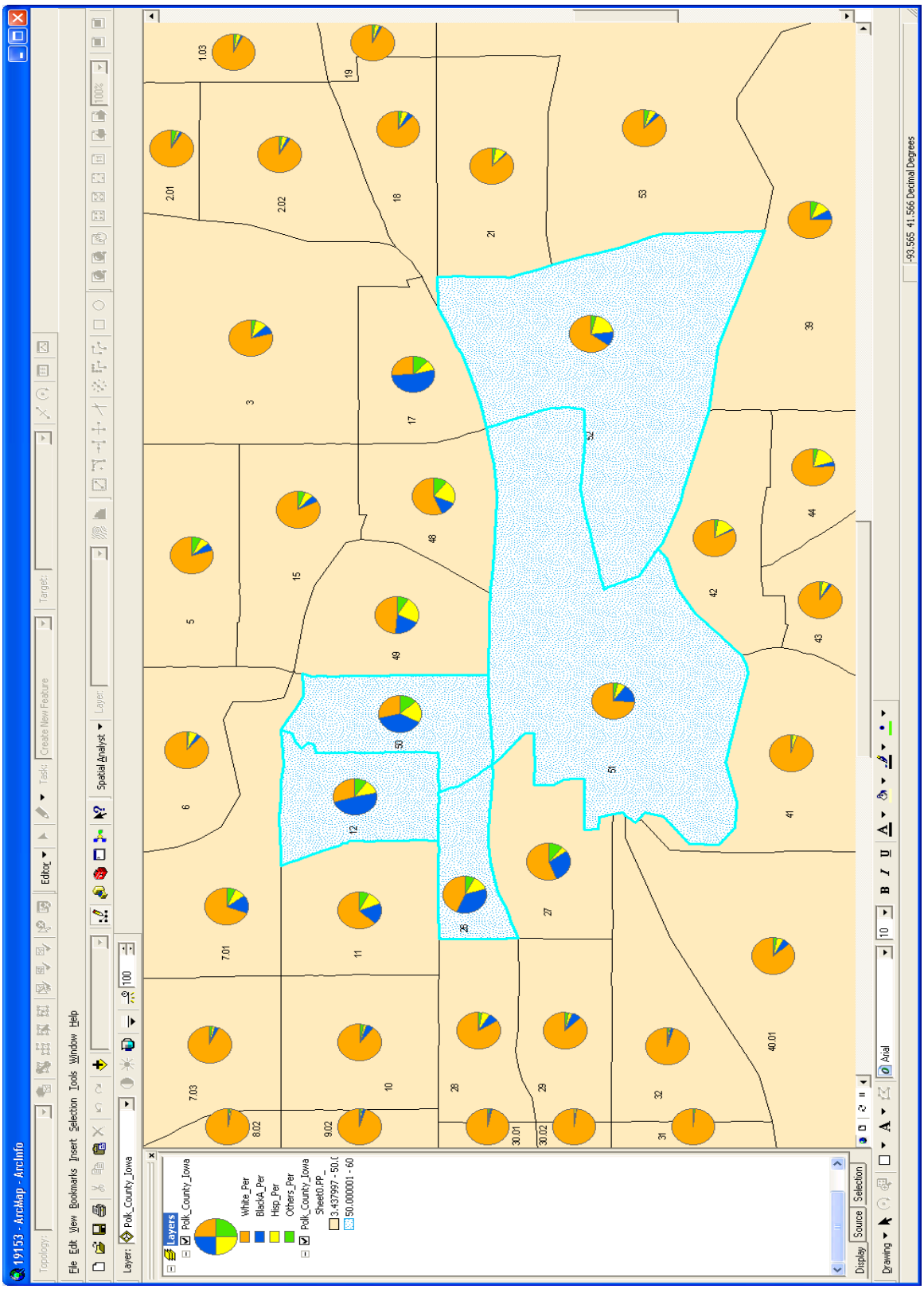

Figure 5.16(c): Polk county

Table 5.4 shows the qualified tracts in the qualified counties, where population is eligible for SNAP benefits.

Table 5.4 County and their SNAP qualified tracts

S. No.	County Name	SNAP Qualified Tracts Number
1	Black Hawk	1, 3, 5, 9, 23.02
2	Linn	19, 27
3	Polk	12, 26, 50, 51, 52

Hypothesis Testing Description. In this study, the results are collected on county-basis (Table 5.4), and a tract in every county is considered as the smallest spatial granularity. As it is mentioned already, the study is conducted for all ninety-nine counties in Iowa. The hypothesis is proved on this basis: If in a qualified tract there exist more than 50% or more, White in overall population, this indicates that at-least few White population is getting SNAP help. Figures 5.16 (d) -5.16 (i) show the percentage of diverse population in the SNAP qualified tracts. A red dashed horizontal line shows the level corresponds to 50% of the population. The bar graphs are demonstrated only for those tracts that exhibit (50 +X) % of White populous occupancy in the entire tract. Here 'X' is a number whose value ranges from 0 to 50. This guarantees that in SNAP qualified those tracts, there is atleast X % (such that $0 \leq X \leq 50$) of White population that avails the full benefits of USDA's SNAP support and this contradicts the hypothesis or the statement of the US republican.

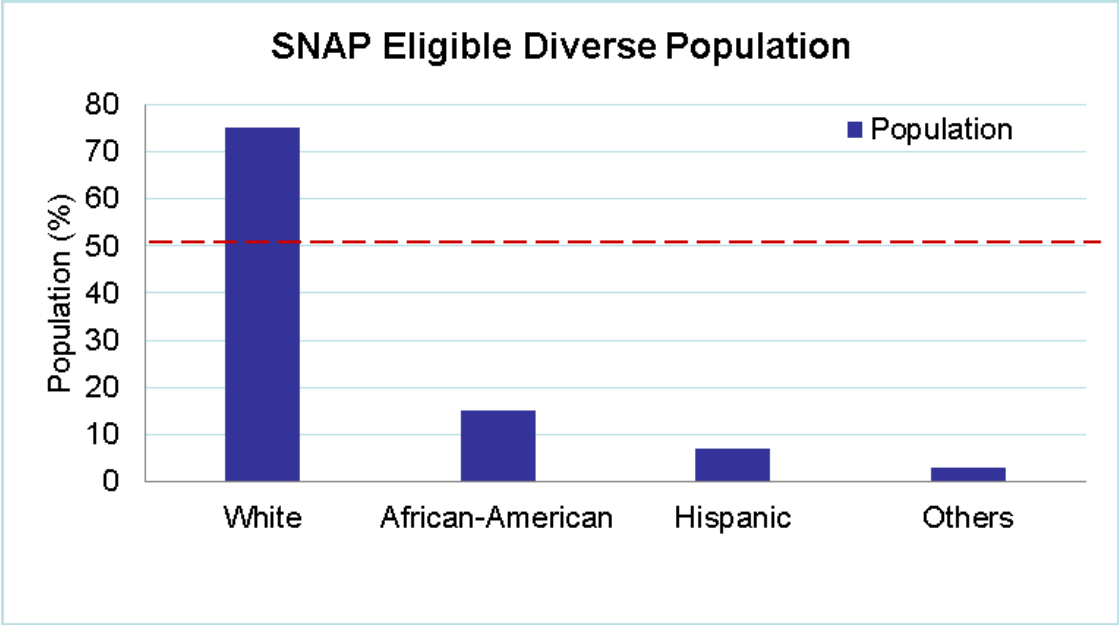

Figure 5.16(d): Black Hawk county (Tract 3)

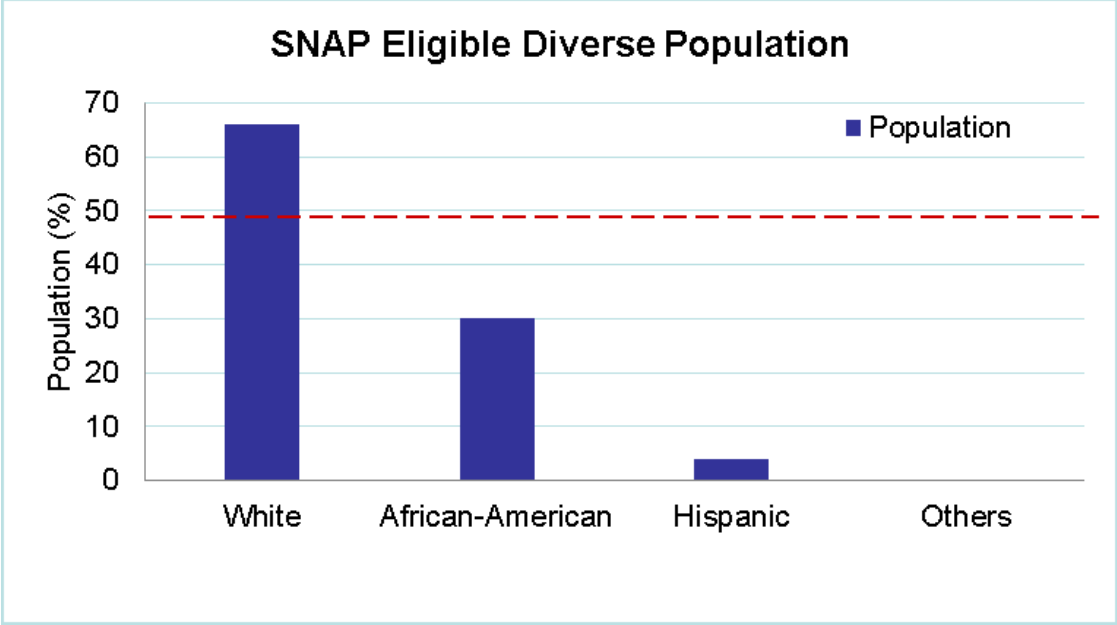

Figure 5.16(e): Black Hawk county (Tract 5)

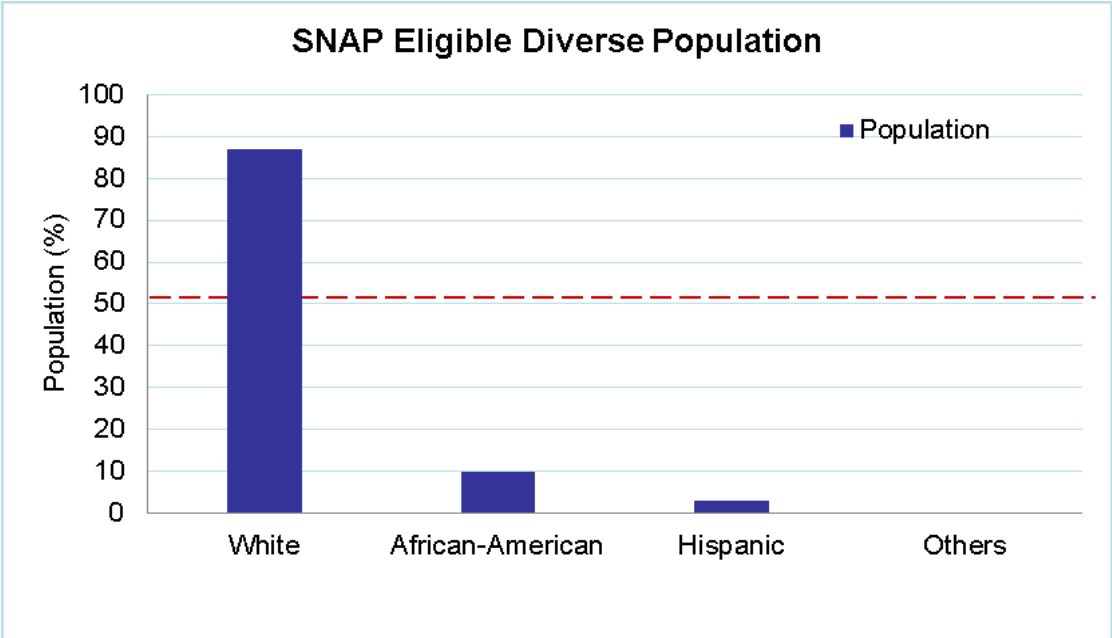

Figure 5.16(f): Black Hawk county (Tract 9)

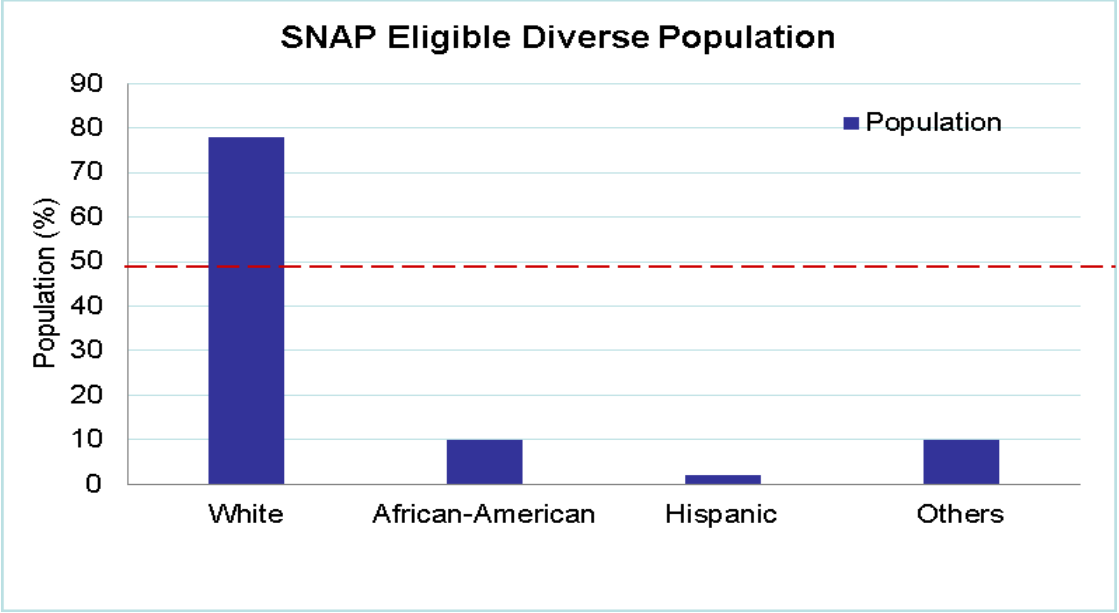

Figure 5.16(g): Linn county (Tract 19)

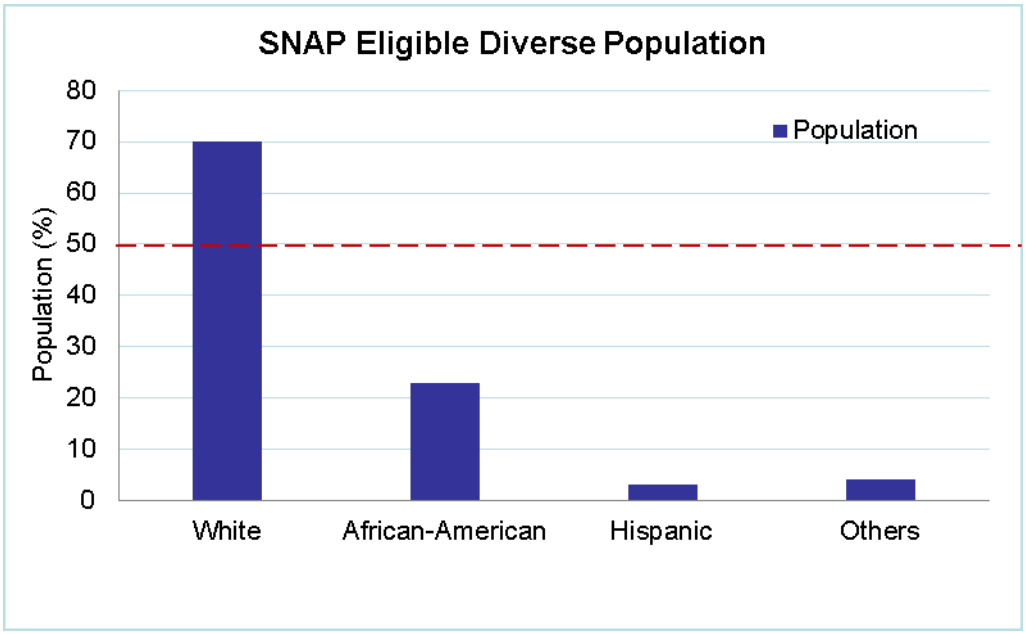

Figure 5.16(h): Linn county (Tract 27)

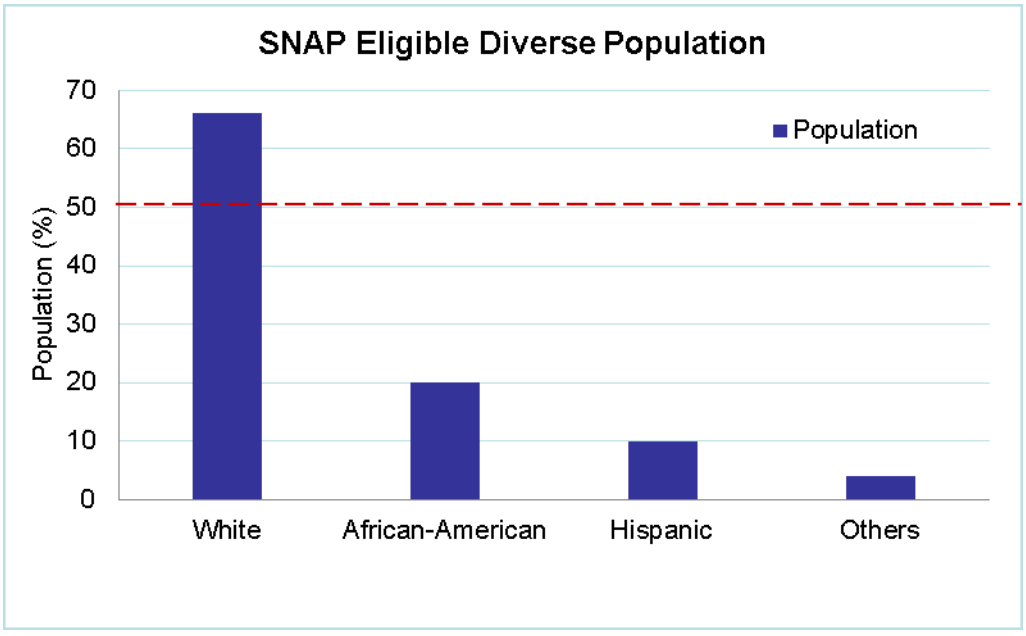

Figure 5.16(i): Polk county (Tract 51)

5.5 SUBSETTING THROUGH VISUALIZATION AND CONVERISON OF SPATIAL DATA - SUBVIZCON

5.5.1 Introduction

In the database community, ordinary, spatial, temporal or spatio-temporal, the concept of subset has been remained of paramount importance. This not only improves the efficiency of the database management system, but also provides the ease to the end user, in analyzing the data. Each community handles the subsetting process in their own fashion. The database community completely relies on SQL-like queries for subset extraction; whereas GIS community is not only dependent on SQL-like queries (attribute query and location query), but often extracts the subset through visualization also. GIS visualization offers click-based selection of a subset and its persistence. The extracted subset possesses the same format and structure as its superset, and is readily available for further use. GIS community exploits software tools to execute its plan. Though, there are numerous tools available in the research and commercial arenas but ESRI's (Dangermond, 1969) ArcGIS has been widely accepted in GIS community due it's functionality richness and easy to use nature. The process of subset extraction through visualization in GIS yields additional power to the user for data analysis. This obviously offers potential benefits to GIS community, at the same time, the database user also enjoys the privileges from it and SubVizCon (Subset through Visualization and Conversion) mediates it. SubVizCon is a software tool that helps to massage the extracted subset to a format, easily accommodated to the parametric database community, where the SQL-like queries can be applied to further subsetting to produce interesting outputs. The resultant can be analyzed through

visual exploration by infusing it to AutoConViz. Thus, a user acquires two-way subsetting capability. SubVizCon can be considered as an additional pillar in the ongoing efforts to strengthen the medium, which encourages the interaction between the database and GIS communities.

Below are a few interesting examples, where visual subset from GIS is proved to be a useful input to the Parametric Data Model and together produce interesting outputs that can lead to favorable decision making. The motivation is to harness the capabilities of both – GIS and the Parametric Data Model. The examples are based on crop and climate data from NC-94 dataset and help investigating the impact of climate on crop.

Example 1: In the year 1999, the crop yield for soybean crop in few counties of Iowa was **below** 43 BU (average crop yield). Report the accumulated GDD (Growing degree days) out in 1999 in those counties if the accumulated GDD is **above** 1500 (accumulated GDD required for soybean crop maturity).

GIS offers click-based interactive classification and classify the soybean crop yield in three classes: above average, average, and below average for the year 1999. The counties having the crop yield below average are selected, grouped together, extracted and stored as a new layer (.lyr), all with click based operations.

```
SELECT accGDD
  RECTRICTED TO [[C.Fips= x]] ∪ [[C.Fips= y]] ∪
                        [[C.Fips= z]] ∩[[ C. Year=1999]] ∩[[ C. accGDD>1500]]
FROM Climate C
```

The new layer is nothing but a subset extracted visually and that provides the spatial domain to restrict the output when used as input in the Parametric Data Model. Such subset classification and extraction are easy in GIS as human nature inclination towards visualization, but will be a cumbersome process in the Parametric Data Model. Let's assume there are only three counties in Iowa: x, y, z (Fips code), having below average soybean crop yield for the year 1990. Now, the accumulated GDD can easily calculated using the following ParaSQL query.

The deposited output (if any) reveals that accumulated GDD is not the only factor behind good crop yield. There are other important climate or soil parameters that impact crop yield. An interested user can further explore other parameters in order to know the actual cause behind the bad crop yield despite the good GDD.

Example 2: In the year 1999, the crop yield for soybean crop in a few counties of Iowa was **above** 43 BU (average crop yield) in 1999. Report the accumulated GDD (Growing degree days) out in 1999 in those counties if the accumulated GDD is **below** 1500.

Similarly in this example also, the following query will deposit the accumulated GDD as output if its value is below 1500 for those counties where in 1999, the soybean crop yield was good. The will be helpful to find out the other causes of good crop yield despite the bad GDD and which will be help in future decision making.

```
SELECT accGDD
  RECTRICTED TO  [[C.Fips= x]]  ∪ [[C.Fips= y]] ∪
                 [[C.Fips= z]] ∩[[ C. Year=1999]] ∩[[ C. accGDD<1500]]
FROM Climate C;
```

5.5.2 Functional Architecture

This section describes the functional architecture of the SubVizCon and the context, where this tool may be beneficial. The description is modularized according to the figure 5.17 for better understanding.

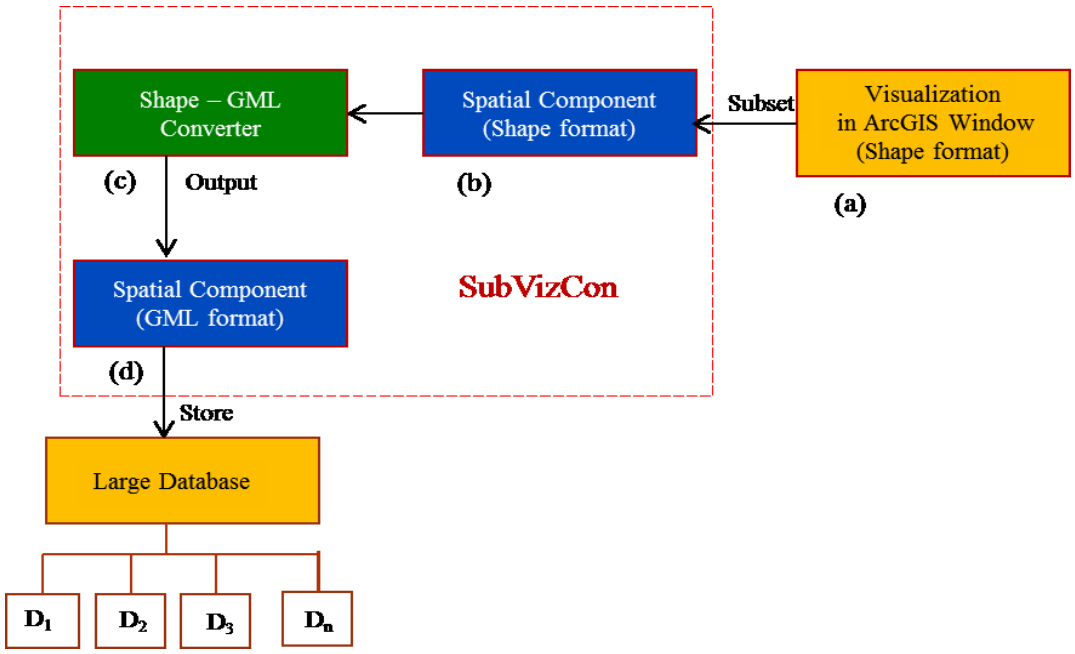

Figure 5.17 Functional architecture of SubVizCon

(a) Visualization in ArcGIS Window. As stated already that ESRI's ArcGIS (ArcMap) provides potential support to analyze the spatial data through visualization. The map window of ArcGIS visualizes the map. The visualization may be a single layered or an overlay of more than one layer. ArcGIS offers a huge pack of functionalities which can be applied to the visualized map to do further analysis. "*Selection*" is one of the important functionality among others and is of the interest in this work. This invokes the partial or full selection of the visualized artifact.

(b) Spatial Component (Shape format). In ArcGIS window, the visualized object (in the phase a) is selected partially and a subset is extracted and persisted for future use. The format of the subset (shape file format, say *x1.shp*) is preserved from its superset. The shape file (*x1.shp*, associated with *x1.dbf*, *x1.shx*) format of the spatial component is already discussed in greater detail (refer section 4.3.1 - c). This serves as an input to SubVizCon. As the subset is persisted and is a self-contained new layer, it can be visualized back using ArcGIS if needed.

(c) Shape-GML Converter (SubVizCon). The bare functionality of SubVizCon can be described as the conversion of a spatial artifact, being used in GIS community (in shape format) into another format (.gml) adaptable in database (especially parametric database) community. In the state-of-the art SubVizCon is conceived theoretically but not yet developed as a software tool.

(d) Spatial Component (GML format). Let's say *x1.gml* is the output of SubVizCon. The format of the output is GML (Geographical Markup Language). The readers who are interested to know more about GML can find voluminous information in literature such as Wikipedia on GML (Gahegan, 2005).

(e) Large Database. In the database world, the analysis of huge amount of data not advisable from efficiency point of view, rather, interesting subset of the vast data is extracted to do interesting analysis. The smaller subset facilitates the desired analysis efficiently. It is assumes that the database consists of multidimensional datasets. In the

figure, D_1, D_2, D_3 ...Dn represent the different dimensions – such as spatial, temporal, etc. - of the datasets.

5.5.3 Prototype Implementation

In the implementation of AutoConViz, I had a well-defined development environment that practiced a standard life-cycle model- water fall model- for software development. The implementation of SubVizCon is also performed majorly on the same existing software and hardware infrastructure and is briefly explained here.

As far as the software and hardware requirements are concerned, in the development of SubVizCon, the following software resources are required in this research work.

- Eclipse Helios IDE (Integrated development environment)
- JDK 1.6

As mentioned already, the implementation of SubVizCon borrows the same infrastructure used for AutoConViz development. The hardware used are: a computer system- equipped with CPU, monitor and mouse. Though, operating system used is windows XP, though it is not the limitation for the development. In this development also Waterfall model is used as a standard software development practice.

5.4.4 Algorithm

The functioning of SubVizCon software tool is relatively simpler and it mainly helps in GML file generation as output if a shape formatted (.shp) file is infused as input to the

software. The steps involved are compiled together in the algorithm in a very informal fashion that should be easy to understand for a laymen person also.

ALGORITHM: Visual subsetting and conversion from Shp to GML

1. **BEGIN**
2. Var x: Desired subset, derived from visual selection of a map in ArcMap window
   // x.shp, x.shx, x.dbf are generated
3. SubVizCon reads input x in its original shape file format (x.shp)
   // Input: x.shp, x.shx, x.dbf
4. SubVizCon produces its equivalent GML format file as output
5. Output: x.gml
   // This output is ready as input to Parametric Data Model for further subsetting
6. **END**

5.4.5 Validation

In order to validate SubVizCon, NC-94 dataset is exploited as test bed. Figure 5.18(a) shows a map of Iowa with spatial granularity as counties. It is worth to mention here once again, the map shown in this figure is the subset derived by executing ParaSQL query (database community) on whole NC-94 dataset, and subsequently visualized automatically with the assistance of AutoConViz.

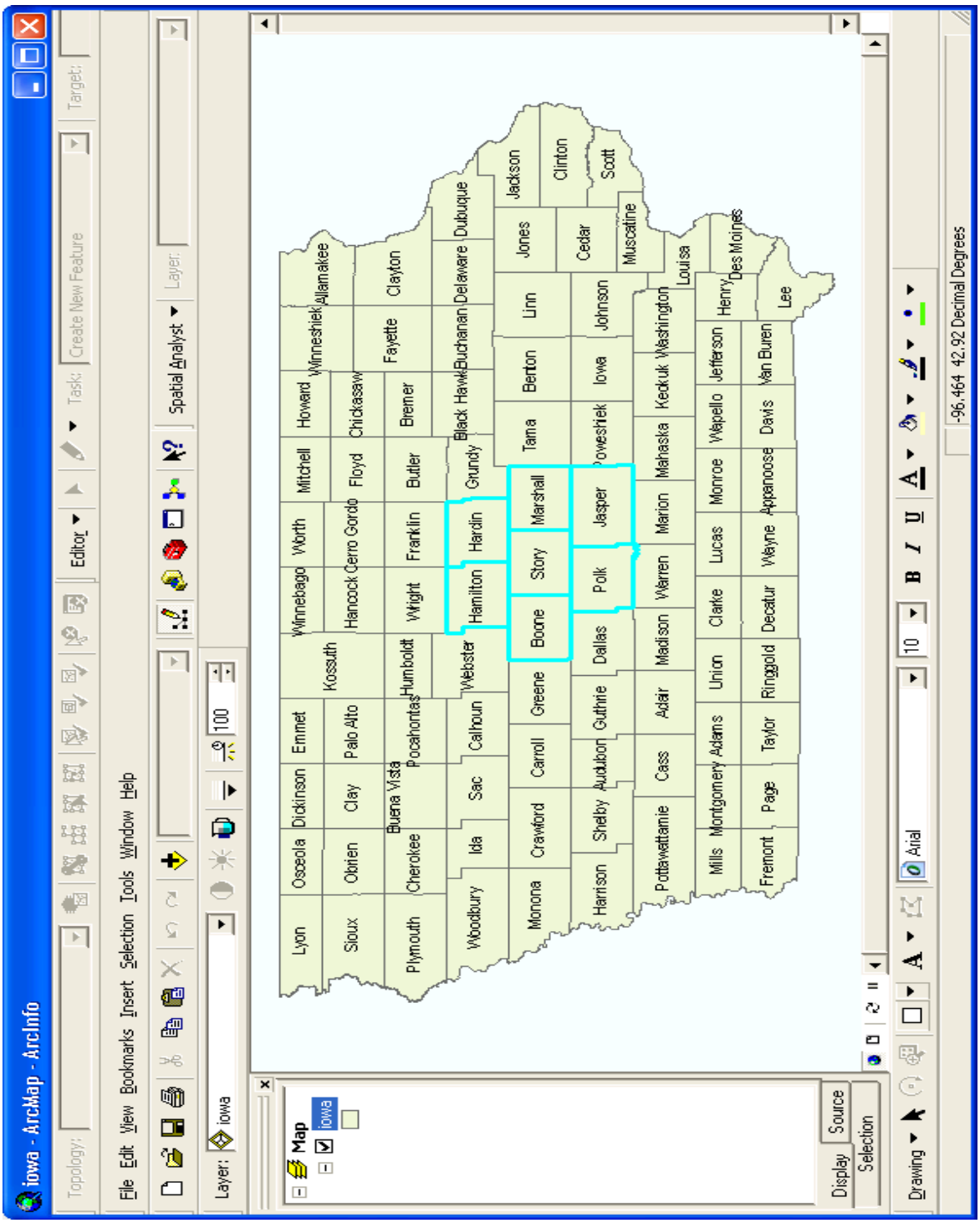

Figure 5.18 (a). SubVizCon - Visual subsetting (selection of interest)

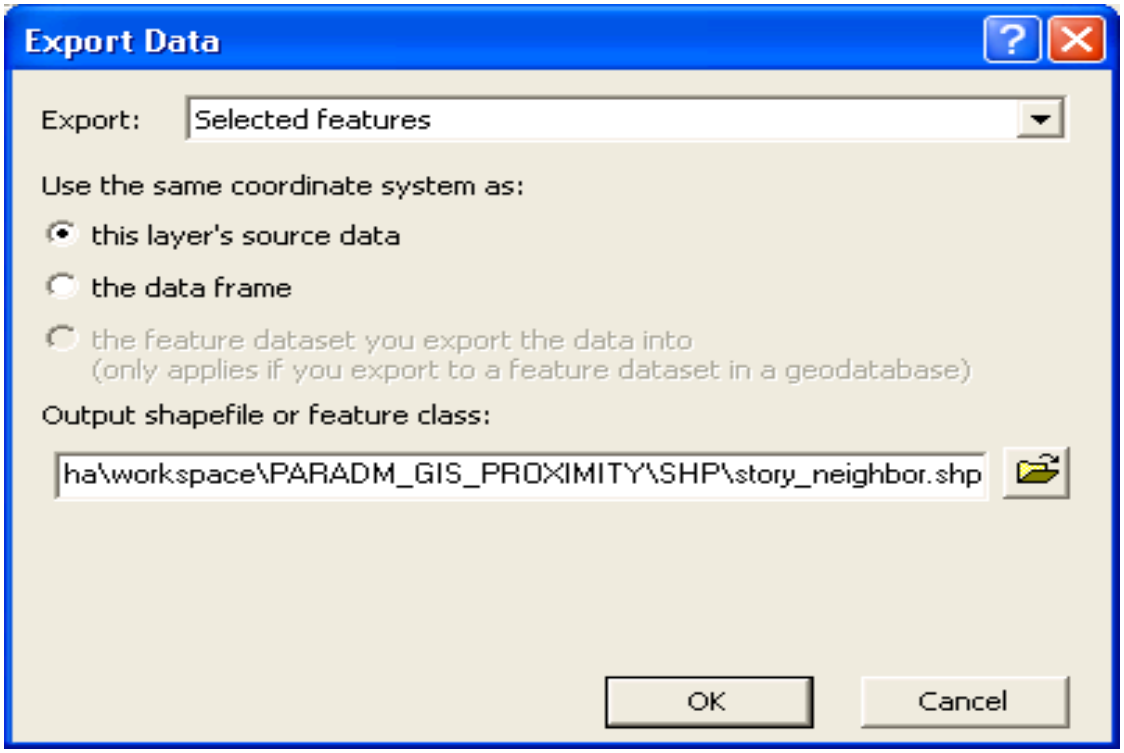

Figure 5.18(b). SubVizCon - Persisting selection of interest

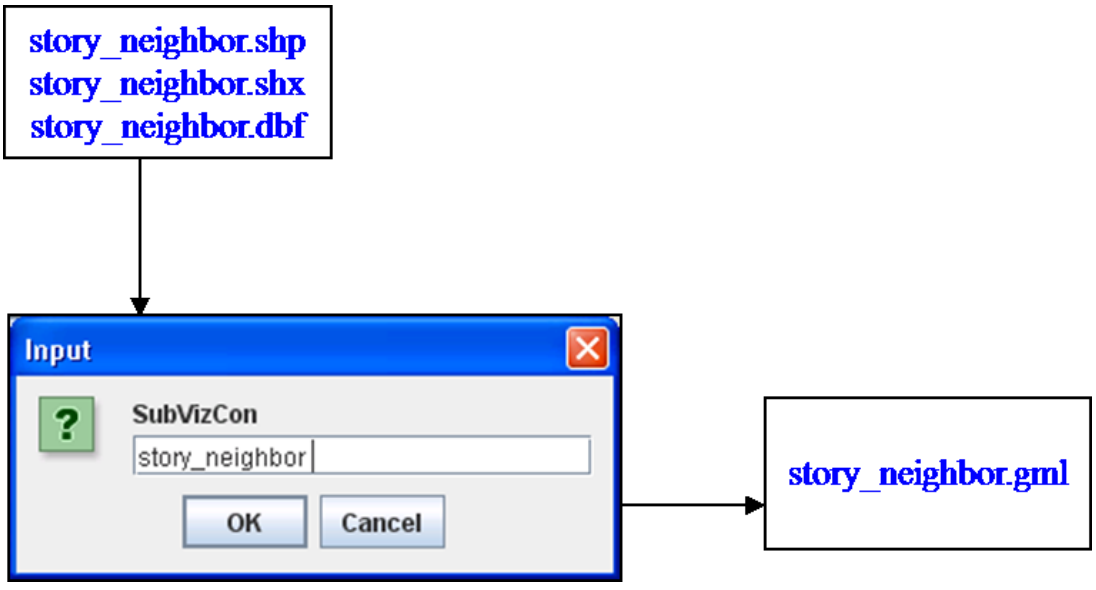

Figure 5.18(c). SubVizCon - Conversion

GIS community is very much familiar with ArcMap functioning. Instead of the entire Iowa, the GIS user is now interested smaller dataset and wants to investigate Story county and its neighbors only. In ArcMap, this smaller set can be extracted by selecting the desired space. The selection is persistent and a new name is assigned to it and figure 5.18(b) depicts this process. The visually extracted subset is renamed as story_neighbor.

The database community, who believes in query execution, wants to exploit this visually extracted dataset. The Parametric Data Model does not recognize the format of visually extracted subset, cannot directly operate on this dataset in its present state. SubVizCon helps alleviate such situation, and serves as a bridge between GIS and database communities. It plays a major role in fostering the output, released by GIS community, as input to database community. The tunneling behavior of SubVizCon is demonstrated in figure 5.18(c). It absorbs the output- released by GIS community through its visual extraction (shape file format) - as input, does its internal processing, and ejects the same in a format readily acceptable by the Parametric Data Model (GML file format), where ParaSQL queries can be executed in order to have interesting subsets further. Thus, SubVizCon helps in establishing one way communication between GIS and database communities (from GIS to database), which oftentimes work in their isolated environments. This coordination certainly offers more power to the user, when it comes to express database queries and will be considered a welcome addition in spatio-temporal databases.

CHAPTER 6. EXAMPLE APPLICATIONS

6.1 Case Study 1: Exploiting query and visualization technology for farm-level decision-making

It has been well documented that plant development, growth and yield depends on temperature. Indeed plants require a specific amount of heat to develop from one stage in the growing cycle to another such as from seeding to the four-leaf stage. A cool May, for example, can significantly delay a plant reaching the four-leaf stage, which could, in turn, optimum weed control strategies. On the other hand, two weeks of warmer than normal July temperatures can advance lentils from green pods to harvest-ready status. Thus, the ability to effectively manage the production system with different management schedule for pesticide application, fertility management, and harvest has become an important component of the farm production enterprises. This management ability has, in turn, depended on knowledge of heat degree days or growing degree days (GDD), an approach to assigning a heat value to each day during the crop growing season. GDD is calculated by: adding each day's maximum and minimum temperatures throughout the growing seasons, dividing that sum by two to obtain an average, and subtracting the "temperature base" assigned to the plant being monitored. The result is a "thermal time" that provides consistent prediction of when certain plant stage will occur and, in general, the "thermal calendar" for a specific crop. Indeed, the "thermal calendar" has become an important production variable and plant yield indicator.

Given that the growth, development , and yield of the crops is directly impacted by GDD, and having this information available can assist farmers make better informed crop management decisions, I explore a case study scenario that expands on and utilizes the

various data mining, software engineering, and spatial visualization capabilities developed in my research. This case study is presented herein as a vignette.

A farmer operating in Central Iowa has learned about the capability of NC-94 dataset in enhancing the planning and management of farm operations. He has also learned that an existing data management query system, ParaSQL, is available to help synthesize, query, and report crop-related information for management decisions. Specifically, he wants to obtain query results from 30 years of continuous Climatic data to plan for next years crop production using GDD as a predictor of management and yield.

To provide the information needed by the farmer, the steps in generating the information, synthesizing the resulting data, and integrating specific GDD data theme with other crop-related and geospatial data proceed as follows.

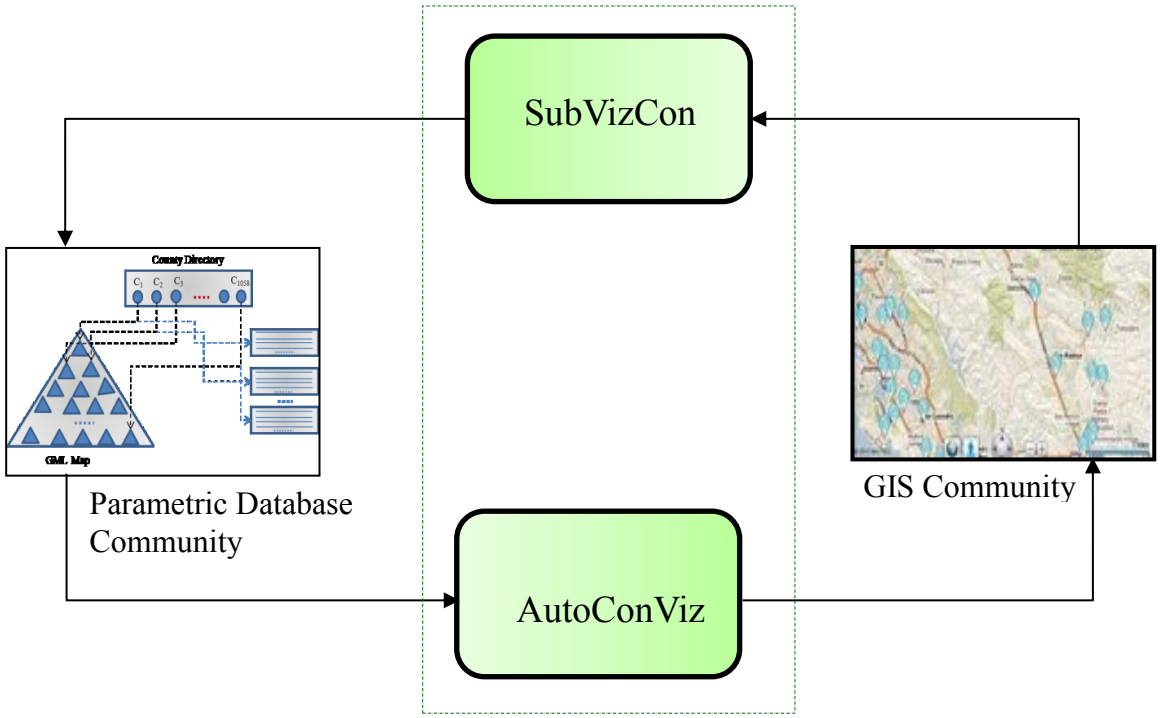

Figure 6.1 Composite framework

Figure 6.1 shows the composite framework of the research work. As it is already mentioned that the research in this dissertation is an attempt to bring the two powerful communities - parametric database and GIS - together for their mutual benefits up to an extent, where the results processed by one as output, should be acceptable by other as input for further processing. Generally these two communities have worked orthogonal to each other in their own territories.

Table 6.1 summarizes the various steps involved in the modeling of information generation, data synthesis, and integrating of GDD with other geospatial data. Though, GDD is not a native attribute in the NC-94 Climatic dataset, but the dataset consists of the two potential attributes (maximum and minimum temperature), which GDD primarily depends upon. An efficient ParaSQL query in the Parametric Data Model helps in deriving it from the native attributes. Figure 6.2 consists of three ParaSQL queries alongwith a command. The execution of the command (command 1) *Gdd 10* locks the parametric database system for GDD mode for a base temperature of 10^0C. Any ParaSQL query executed after the execution of command 1 will report GDD also as an appended attribute in the normal output. The combined output is stored in comma-separated value (CVS) file format. ParaSQL query 1 in figure 6.2 reports the GDD calculated on day-to-day basis and figure 6.3 (a) shows the output resulted on the execution of this query. It can be noticed that a new attribute -Gdd- is appended as new column, which is derived by using the two native attributes. This derived attribute has the potential use in the measurement of crop growth and in making prediction about the crop maturity.

Table 6.1 Steps involved in information generation, data synthesis and GDD integration with other geospatial datasets

Using the query interface: Generation of information and resulting data synthesis
Step 1. Load the NC-94 Climate dataset into database
[Storage for Parametric Data Model - CanStoreX]
Step 2. Execute the ParaSQL query to generate GDD from the Climate data
[Queries in figure 6.2 help in achieving it]
Step 3. User receives the output in CVS format
Data integration: Integration of GDD data with other geospatial data
Step 4. Spatial component of NC-94 data from ParaSQL is exposed to AutoConViz
Step 5. Open ArcMap from AutoConViz
[shp file will already added to opened ArcMap]
Step 6. Import CVS file-from step 3 into ArcMap
[Data as a whole from step 5 and 6 serves as Climate Layer in Arc Map]
Step 7. Add NC-94 Crop dataset to ArcMap
[This overlays on Climate Layer]
Step 9. In ArcMap, a smaller section of this overlay is selected and exported as new
[Newly extracted smaller layer has all attributes of both parental layers]
Step 10. New layer is exposed to SubVizCon and equivalent gml output is produced
Step 11. New output is loaded into storage
[Same as step 1]
Step 12. Another ParaSQL query is executed further on it
[An example ParaSQL query is in figure 6.4]

Command 1. Locking the system for growing degree days (Gdd) mode

```
Gdd 10
```

Query 1. Calculate daily basis gdd

```
SELECT *
FROM Climate C
```

Query 2. Calculate gdd only for contributing days

```
SELECT *
RESTRICTED TO [[(MaxTemp+MinTemp)/2>10]]
FROM Climate C
```

Query 3. Calculate gdd (contributing days) in neighboring counties of Polk County

```
SELECT *
RESTRICTED TO [[(MaxTemp+MinTemp)/2>10]]
FROM Climate C
WHERE NEIGHBOR (C.FIPS, 19153) = TRUE
```

Figure 6.2 ParaSQL queries to calculate Growing Degree Days (Gdd)

The ParaSQL query 2 reports GDD only for the contributing days. A day is considered a contributing day if it contributes atleast a minimal degree of heat, above the base temperature. In other words, a day is a contributing day if the average temperature of that day is atleast slightly greater than the base temperature. The screenshot in figure 6.3 (b) depicts the outcome of the execution of query 2. The ParaSQL query 3 provides an opportunity to the user to calculate the GDD in neighboring counties and figure 6.3 (c) shows the GDD calculated for the neighboring counties of Polk county in Iowa.

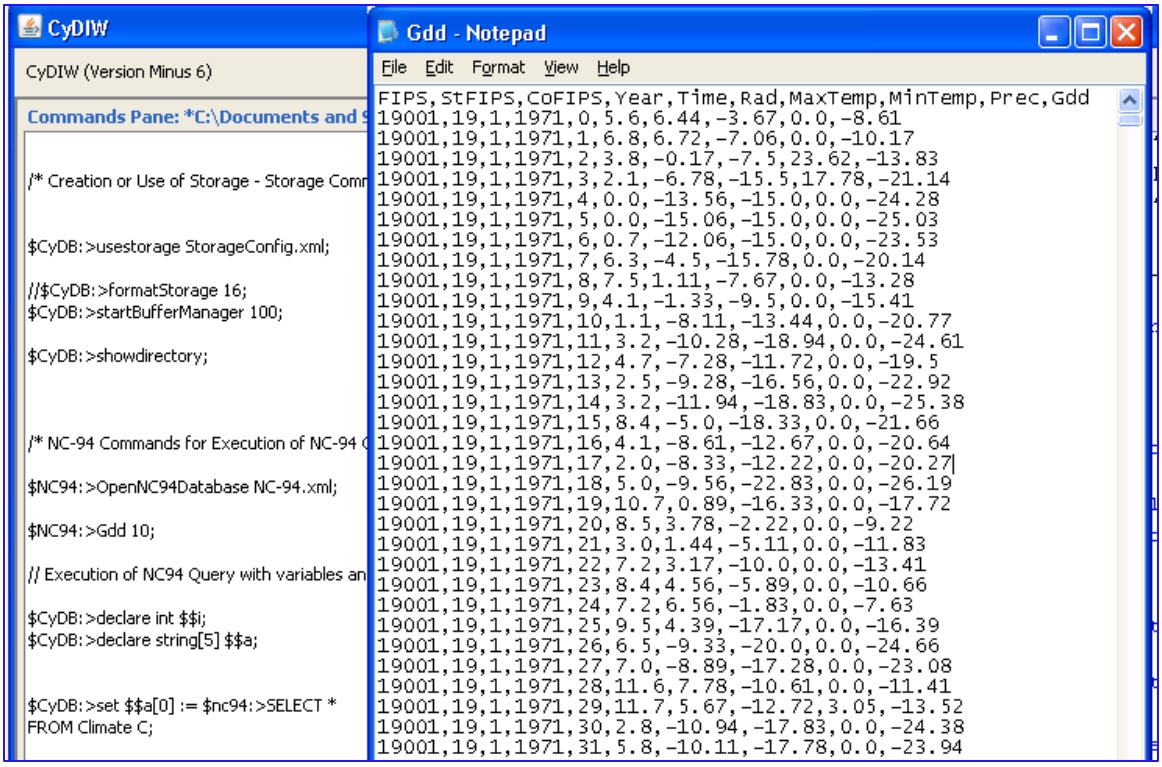

Figure 6.3 (a) Growing degree days (gdd) calculated on daily basis

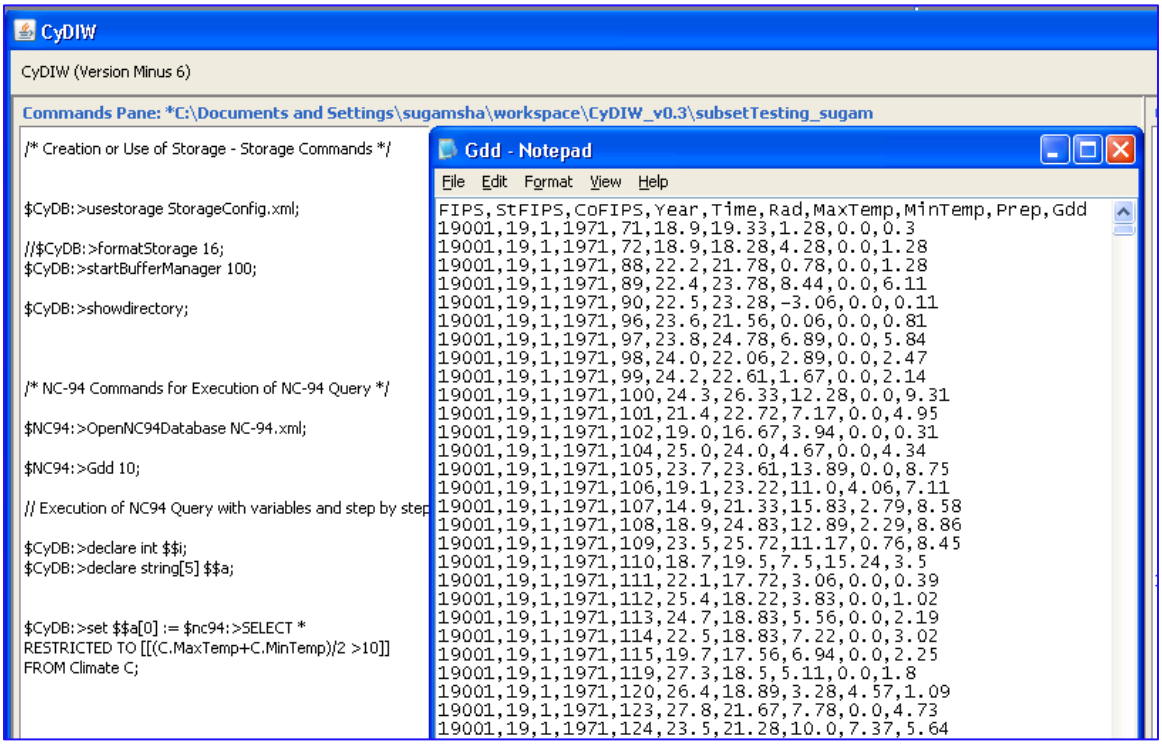

Figure 6.3 (b) Gdd calculated for contributing days only

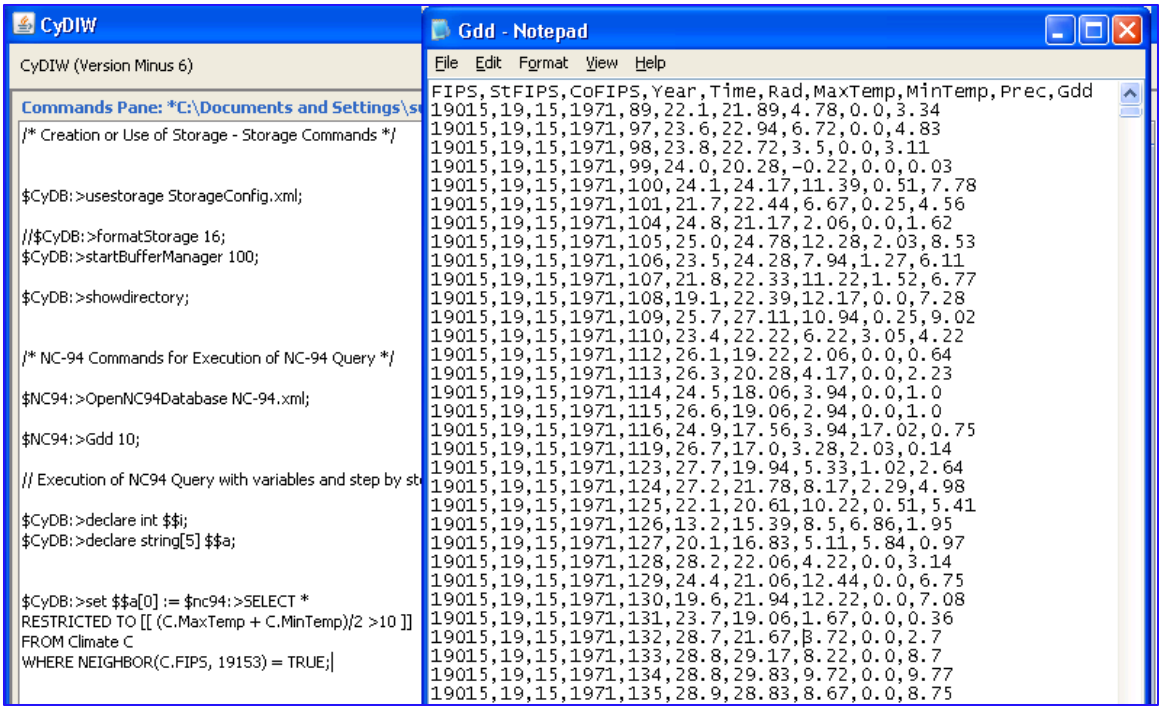

Figure 6.3 (c) Gdd (contributing days) calculated for the neighboring counties of Polk county

The aspatial outcome of the ParaSQL query in step 1 is stored in comma-separated values (CVS) format whereas the spatial component is passed through AutoConViz and subsequently both the artifacts are added to ArcMap. This will serves as a Climate layer. As in the scenario, the farmer has shown interest in crop production; the NC-94 Crop dataset is also imported to ArcMap and overlaid on the previously added Climatic layer. Import of the NC-94 Crop dataset into ArcMap is not direct and requires some intermediary steps as the dataset originally is available in MS Access that needs to be exported into CVS format first before importing into ArcMap. The *Selection by location* and *Selection by attribute* operations in ArcMap assist in analyzing the heat units and crop yields in different counties. Through, the visual analysis of heat accumulation and crop yield, the farmer narrows down his search to a smaller region – say LYON,

JASPER, LEE, POLK, JACKSON, and DALLAS Counties only. The region of interested is selected through the visualization and extracted as a persisted layer. This selection penetrates though both the layers so the extracted subset consists of all the attributes from Climate and Crop layers. The subset is passed to SubVizCon that subsequently produces the output (say SubReg) of the extracted region that is readily adaptable to the Parametric Data Model where ParaSQL queries can be further applied on it. The farmer further restricts his spatial choice and now just wants to focus on the neighbors of POLK county only. He is interested to have the yield for the soybean crop for the years- 1998, 1999, and 2000 when the cumulative heat accumulation was atleast 1200^{o}C. The ParaSQL query in figure 6.4 will report it.

```
SELECT Sr.FIPS, Sr.Year, Sr.Yield
  RESTRICT TO [[Sr. Commodity=soybean]]∩[[Sr. agg(Gdd)≥1200]] ∩
                              ([[Sr. Year=1998]] ∪[[Sr.Year=1999]] ∪[[Sr.Year=2000]])
  FROM SubReg Sr
WHERE NEIGHBOR (C.FIPS, 19153)=TRUE
```

Figure 6.4 ParaSQL query to calculate crop yield

6.2 Case Study 2: Near Real Time Obesity Analysis and Impact of Food Deserts

Obesity in humans is defined and calculated in term of Body Mass Index (BMI). An adult or older adult with BMI of 30 or higher is considered obese. It is a measure of an adult's weight in relation to his or her height, specifically the adult's weight in kilograms divided by the square of his or her height in meters, though the formula for calculating BMI in male and female population varies. Obesity is prevailing at a very fast rate across the United States and today, two-thirds of Americans are either overweight or obese.

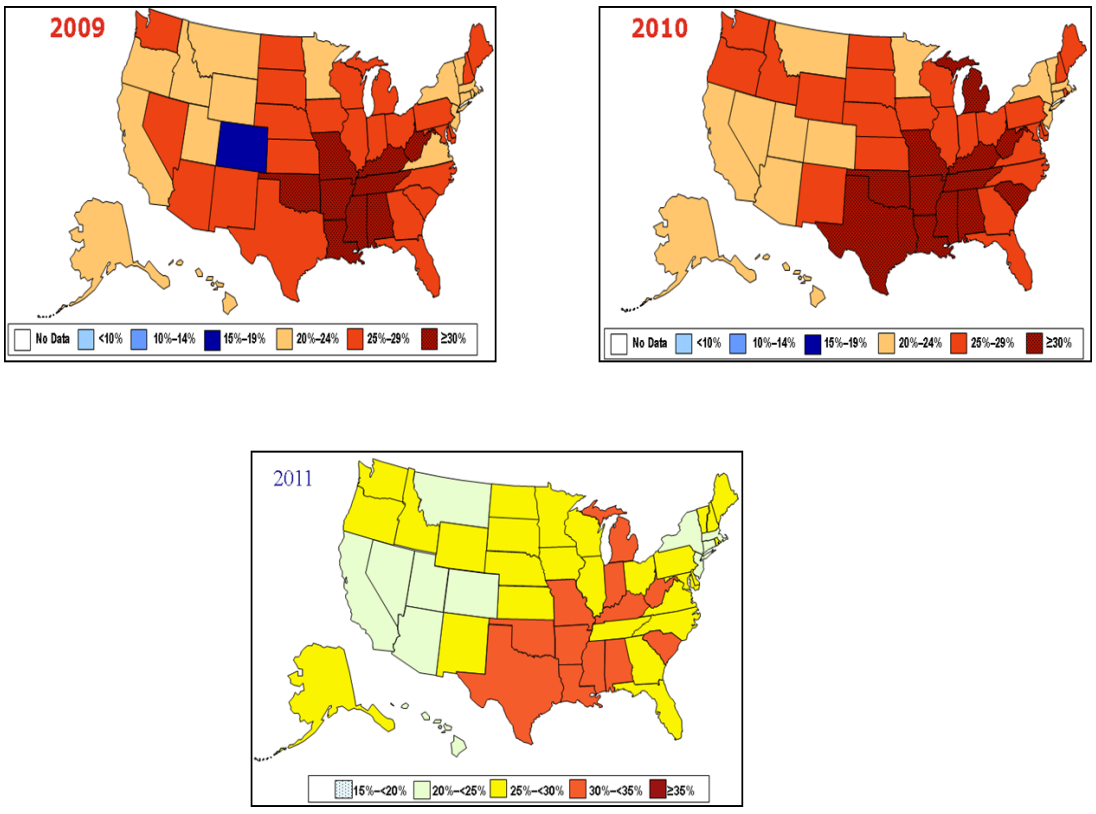

Figure 6.5 Obesity prevalence across US - 2009, 2010 & 2011 (Photo courtesy of CDC)

In the United States, of the total health spending, obesity alone dominates on 9% of it and today the spending only on obesity-impacted health touches $147 billion dollars every year. This amount has become just double if compared with what it used to be nearly a decade ago. The average annual health spending is $1400 more for a person with obesity as compared to a human of ideal weight. The obesity prevalence for the year 2009, 2010, and 2011 in the United States is shown in figure 6.5. As per the most updated information available from "Centers for Disease Control and Prevention," the minimum prevalence of obesity in the United States is 20%. There are thirty-six states that have the prevalence not less than 25%. There are twelve states where it has been measured 30% or higher, outlying Mississippi as the most obese state (35%).

A healthcare agency is interested in near real-time obesity analysis on a certain randomly selected population. The obesity data is getting stored in the repository on monthly basis throughout the year, through self-reported system. Lately, there have been continuous rattling stories that food desert fuels obesity. The agency is also interested to analyze the impact of food deserts on obesity. The geographical region of interest selected for this analysis is the state of Iowa, especially Story county.

Figure 6.6 is the architecture, used for the study that inherits the base from the composite framework in figure 6.1. It clearly shows that on the GIS side obesity layer, foodscape layer, and base layer for Iowa are added to ArcMap. The overlaid map shows classified foodscape –healthy vs. unhealthy food facilities, obese locations, and blocks in the Story county of Iowa.

Table 6.2 summarizes the steps required to model, this near real time obesity analysis study and the impact of food deserts on obesity.

Table 6.2 Steps involved to model near real time obesity analysis and impact of food desert on obesity

Step 1.	Add the Iowa base map layer to ArcMap
	[base map shows county and block levels]
Step 2.	Import obesity data table and POI data table into ArcMap
	[Add X, Y coordinates of both the tables -one at a time- into the map
Step 3.	Generate obesity layer and POI layer from the tables in step 2
Step 4.	Overlay the new layers in step 3 on base map layer in step 1
Step 5.	From POI layer using *Selection by attribute* select grocery data only
Step 6.	Classify the grocery data base on industry type –fast food and grocery store
	[fast food store- unhealthy food, grocery store – healthy food facility]
Step 7.	Perform *Spatial Join* operation to spatially join attributes of all three layers
	[ArcToolbox >Overlay >Spatial Join]
Step 8.	On map, zoom into Story County and select it
	[All the spatially joined attributes will be selected too]
Step 9.	Extract the selected region and exported it as a new layer
	[Newly extracted smaller layer has all attributes of all three parental layers]
Step 10.	New layer is exposed to SubVizCon and an equivalent gml output is produced
	[GML format has big scope of analysis in GIS science and readily available]

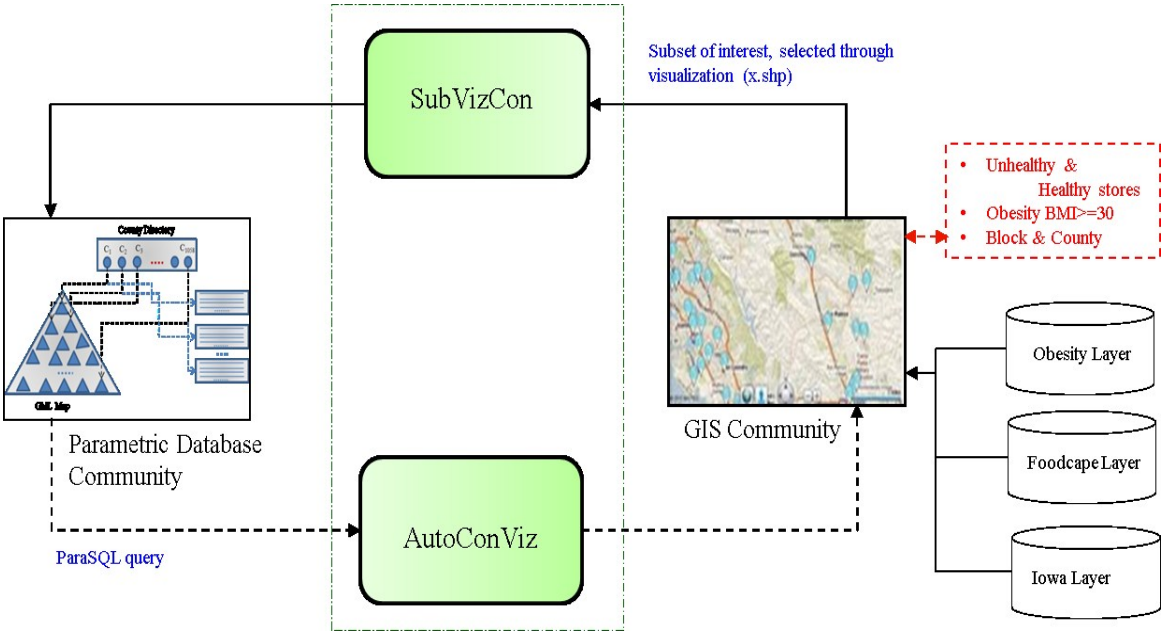

Figure 6.6 Architecture

It is assumed that the near real time obesity data (BMI) is continuously stored into repository on monthly basis. Due to the unavailability of the actual data, for this study, the synthetic obesity data is prepared, containing 12 months BMI of a random cluster of population in Iowa for the year 2003. The spatial location considered is the Story county in state of Iowa. Point of data (POI) data for the story county is downloaded. The POI data consists of foodscape (Grocery) data also alongwith other POIs too such as government buildings, healthcare facilities, etc. The base layer for the state of Iowa - county & block level - is downloaded from census bureau. All the three datasets are imported into ArcMap and overlaid. The foodscape data is classified using industry type and symbolized using variable size and color of the symbols.

The visual based analysis in ArcMap helps not only to locate the obese clusters in the neighborhood but the healthy vs. unhealthy food facilities also. In ArcMap, *Selection by location* and *Selection by attribute* operations greatly assist and enhance the analysis by slicing and dicing it.

To analyze the impact of food deserts on obesity, an obese cluster of population in Iowa (Story county) is selected through visualization as the only region of interest. Selection is penetrated through the overlay and used spatial joins to conglomerate attributes of all the layers (figure 6.7 and table 6.3). The selected region is zoomed in to clearly observe the geographical locations of classified food facilities. This provides visual base analysis of proximities of healthy vs. unhealthy stores from neighborhoods. Attribute query helps in counting the total number of healthy vs. unhealthy stores in a given region. It is observed that number of unhealthy food facilities is more than healthy food facilities and are in closer proximities of the neighborhood comparatively in the given potentially obese region. This selection is exported as a separate subset, consisting of attributes of base layer, obesity, and food facilities, which on exposing to SubVizCon produces an equivalent GML format that also has big potential scope of analysis in GIS science.

Table 6.3 Attributes of Story county, obesity, and foodscape layers after spatial joins

Attributes of Block_Obesity_Grocery_SpatialJoin

FID	Shape *	BLKIDFP	NAME	INTPTLAT	INTPTLON	Year	Age	Se	Educati	Occupati	Race_Et	Inco	BMI1	BMI2	BMI3	BMI5	BMI6	BMI8	BMI9	BMI12	Name_1
3440	Polygon	191690013021001D	Block 1001D	+42.0023184	-093.6100134	0	0					0	0	0	0	0	0	0	0	0	
3441	Polygon	191690009004015	Block 4015	+42.0273845	-093.6161662	0	0					0	0	0	0	0	0	0	0	0	Fareway
3442	Polygon	191690009004024	Block 4024	+42.0255090	-093.6145557	0	0					0	0	0	0	0	0	0	0	0	
3443	Polygon	191690010002013A	Block 2013A	+42.0209104	-093.6157466	0	0					0	0	0	0	0	0	0	0	0	
3444	Polygon	191690011004007D	Block 4007D	+42.0108365	-093.6461694	0	0					0	0	0	0	0	0	0	0	0	
3445	Polygon	191690009005001B	Block 5001B	+42.0298479	-093.6233183	0	0					0	0	0	0	0	0	0	0	0	Restaurant:Culver's
3446	Polygon	191690009005001B	Block 5001B	+42.0298479	-093.6233183	0	0					0	0	0	0	0	0	0	0	0	Fast Food Restaurant
3447	Polygon	191690009004001B	Block 4001B	+42.0295564	-093.6134274	0	0					0	0	0	0	0	0	0	0	0	
3448	Polygon	191690009005000B	Block 5000B	+42.0295026	-093.6212891	0	0					0	0	0	0	0	0	0	0	0	
3449	Polygon	191690009005005B	Block 5005B	+42.0276306	-093.6264211	0	0					0	0	0	0	0	0	0	0	0	
3450	Polygon	191690010002006B	Block 2006B	+42.0231453	-093.6132702	0	0					0	0	0	0	0	0	0	0	0	
3451	Polygon	191690010002011B	Block 2011B	+42.0226097	-093.6165488	0	0					0	0	0	0	0	0	0	0	0	
3452	Polygon	191690105001151	Block 1151	+42.0670263	-093.3911239	0	0					0	0	0	0	0	0	0	0	0	
3453	Polygon	191690105001086	Block 1086	+42.1149987	-093.4007550	2003	51	M	MS	Engineer	Hispanic	70	32	32.1	32.6	32.8	33	33.1	33.2	33.1	

Record: 0 Show: All Selected Records (0 out of 3686 Selected) Options

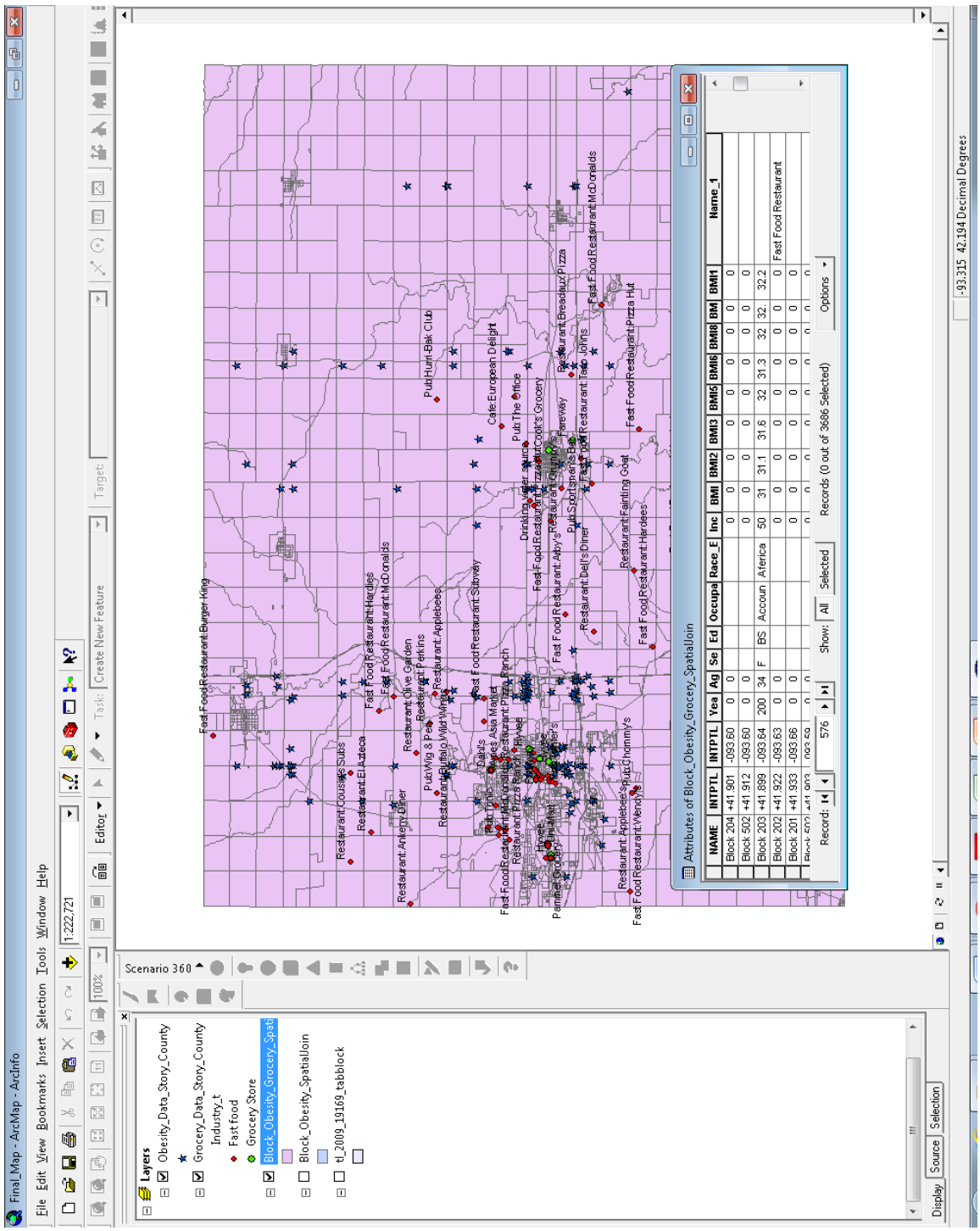

Figure 6.7 Story county base, obesity, and foodscape (grocery) layers in ArcMap

CHAPTER 7. SUMMARY AND FUTURE RESEARCH DIRECTIONS

7.1 SUMMARY

Within the database community, the Parametric Data Model has been studied since mid 1980s. It is known for its ability to handle multidimensional data with spatial, temporal and, spatio-temporal granularity. A working prototype of the Parametric Data Model already exists and is validated using one of the most complete spatio-temporal dataset, NC-94. Though the prototype is equipped with enormous functionalities, but still lacks few core functionalities: - (1) concept of subset, (2) subquery capability, and (3) visual support for the ParaSQL query results. The addition of these artifacts to the Parametric Data Model enhances it to a level, where it draws better attention from the database community and hopefully in its entirety, will be placed in the league of the other most popular database models available today. In this dissertation, these research challenges were addressed in five phases and are given below:

1. Implementation of the concept of subset in Parametric Data Model.
2. Addition of the subquery capability to the Parametric Data Model.
3. Expansion of ParaSQL.
4. Integration of the visualization to the Parametric Data Model.
5. Visual exploration of large spatial and spatio-temporal datasets.

7.1.1 Implementation of the concept of subset. Before this study was taken place, unlike other popular database models, there was no scope in the Parametric Data Model

to store the extracted subset back into the storage. In this dissertation, the concept of "subset" was successfully developed in the existing prototype of the Parametric Data Model. A new clause - INTO- was introduced in the ParaSQL. Though the behavior of this newly introduced INTO clause is kept optional but the presence of it in a ParaSQL query enforces the prototype to stores the extracted resultant (subset) back into the same storage. It is worth noticing here that the stored subset preserves the same structure and its parent object and this subset is readily available to be exploited by another ParaSQL query, similar to its parent. In this dissertation this enhanced functionality of the Parametric Data Model has been validated successfully using spatio-temporal NC-94 dataset as test bed.

7.1.2 Addition of subquery capability. The successful implementation of the subset concept in the Parametric Data Model opened up avenues for another useful concept, called "subquery". Unlike other popular database models, the Parametric Data Model had the deficiency of subquery execution. In a ParaSQL subquery concept, a ParaSQL query (inner) is embedded inside another ParaSQL query (outer) and this complex query is applied to the relation(s) as a whole. This conforms the concept of the classical database paradigm which involves inner and outer query. The concept of subquery is successfully added to the implementation of the prototype and is validated using the same spatial-temporal NC-94 dataset.

7.1.3 Expansion of ParaSQL. In a database management system *where* clause has its own merits in the query execution sequence as it filters out the unsatisfying tuples from

the top level only, thus avoids their further unnecessary processing and contributes to system performance. The concept of *where* clause in ParaSQL was very powerful but required further extension in order to harness it as before this study, only a simplistic flavor of it was implemented. In the Parametric Data Model, the *where* clause depends on the tuple (as whole object). A parametric tuple may contain multiple tuplets, which are the true records and contain attributes. In this dissertation, two newly developed iterators were used. One iterator iterated though the series of objects (tuples) and a qualified object was scanned though another iterator for internal navigation though that object. The new implementation exhibited a greater performance and the expansion cleaned the logic of *where* clause and made it more generic which can handle any similar kind of spatio-temporal data in which the geography does not change with time. In this work a general purpose technology was implemented for spatio-temporal object where geography does not change with time (stationary objects). In order to load a real dataset into parametric relation, now a generalized loader was used, which worked for any relation of this type including climate, crop, soil and their subset.

7.1.4 Visualization aid to Parametric Data Model. This study enriched the prototype of the Parametric Data Model with visualization capability, which was completely unavailable in the previous version of the prototype. A modular implementation for visualization is done in the Parametric Data Model and the ParaSQL query is further enhanced with new clauses – DISPLAY, VIA. The DISPLAY clause invokes the visualization whereas VIA clause facilitates the acceptance of an input configuration file

(in XML format) which provides color coding schemes alongwith other useful information.

In the current implementation of visualization concept in the Parametric Data Model, it is observed that the visualization of an attribute is a terminal visualization where no further operation is possible. A minute change in the output needs either the whole process to be repeated or lots of implementation efforts are required with special skills which are not easy to possess for everybody. Therefore, this initial visualization support in the Parametric Data Model was not matured enough and needed further development in order to have exploratory visualization.

Geographic Information System is widely used in geospatial visualization and is accepted as one of the most suitable technologies in spatial database research due to its visualization strength. In GIS community, ArcGIS is considered as the potential representative of GIS technology. In this dissertation, the comprehensive strength of ArcGIS visualization was integrated seamlessly into the Parametric Data Model to serve as the visualization platform. This enriched the Parametric Data Model with visualization capability that offered non-terminal visual outputs that had wide scope for further processing and analyzing using variety of operations such as sub-setting. The outputs were not localized to ArcGIS only, rather can be fed back to the Parametric Data Model as a fertile input, where ParaSQL operations were welcome further to operate on it. This entire circular process offered two-way communication between the Parametric Data Model and GIS and helps to bring them together. Their close proximity might evolve better ways to deal with large spatial, temporal or spatio-temporal databases and produce very interesting results that have better outreach and larger societal impact. The

establishment of this two-way communication was not direct, rather leveraged by two intermediary software tools, developed as a part of this research work and were called, AutoConViz (Automatic Conversion & Visualization) and SubVizCon (Subset from Visualization & Conversion). AutoConViz facilitated the query results visualization (from Parametric Data Model to GIS) whereas SubVizCon assisted the activities, being initiated at GIS side such as visual sub-setting in GIS and migrate towards the Parametric Data Model for further processing. This integration was validated using test cases with NC-94 dataset as test bed.

7.1.5 Visualization of large-scale spatio-temporal data. The above description about AutoConViz stated that it assisted the visualization of the large spatio-temporal dataset efficiently. It also offered navigation to ArcMap window. This strengthened its capability further as that allowed the fully exploratory visualization of large spatial or spatio-temporal data. In this dissertation, I availed its advantages and exploited other important large spatio-temporal datasets to obtain the potential insights in useful applications. One such important application successfully studied in this research work was to visualize and analyze the geo-spatial patterns for *Supplemental Nutrition Assistance Program (SNAP)* eligibility for Iowa. The study was conducted for all ninety-nine counties in Iowa and only those counties in which the population at the census tracts level (at least in one tract) was eligible for SNAP were shown as results in ArcGIS.

This research work was further extended beyond finding the SNAP qualified tracts only. The extended study piggybacked on the previous SNAP eligible patterns and the qualifying tracts were considered as the base layer to explore the socioeconomic diversity

of the SNAP eligible population. The total population in the tract was divided mainly into four classes based on the race\ ethnicity: (1) White American, (2) African American, (3) Hispanic, and (4) Others. The class "Others" included all the other races except the above mentioned races, with a majority of Asians. In ArcMap, pie charts were drawn to represent the percentage of the socioeconomic diversity in the tracts. The results revealed that in the United States, not only minorities availed SNAP benefits, but other sections of the society also depend on SNAP support.

7.2 FUTURE RESEARCH DIRECTIONS

At the abstract level, this research work contributes in the advancement of the science primarily of two individual research communities – 1) database community and, 2) GIS community. Each of the community consists of a great set of functionalities in their own rights and generally works in their own restricted domains. The database community is considered as the potential player when it comes to the engineering of the data. At the same time, the GIS community claims to be the potential player when the visualization and analysis of the data is the sole concern. Mixing both of them may offer a great deal of rich functionalities to the user who in turn can handle the much more complex scenarios related to data and the research challenges taken up in this dissertation attempt to bring them together to harness their mutual benefits.

The entire journey of this dissertation has significantly advanced my scientific and technical knowledge in the database implementation and GIS technology – especially

software development, visualization and exploratory analysis, enriched me with new skills and fostered the new research ideas.

During the last decade, the GIS technology has become a popular problem-solving and spatial analytical tool for researchers and organizations that wish to understand systemic and structured variables in data related to human-social systems. GIS has especially been used to visualize spatio-temporal relationships and to perform exploratory spatial analysis in numerous potential research areas. For example, in public health and epidemiologic research, GIS is used to visualize and to conduct exploratory spatial analysis on disease etiology (Cromley and McLafferty, 2011). It is also used for complex multivariate analyses to elucidate causative factors and establish relationships between exposure and health outcomes (Wakefield and Shaddick, 2006). This is an immediate and obvious research direction where I would like to explore my GIS knowledge further.

Today, “Big Data” (White, 2012; Robert, 2012) has evolved as a predominant research area where I also would like to exploit my data engineering capabilities. Big data is defined as the large, diverse, complex, and distributed data - structured or unstructured - generated from number of resources such as instruments, sensors, internet transactions, email, video, click streams etc. (Villars et al., 2011; IBM, 2012). The size of big data is a big research challenge today as in 2012 alone it ranges from a few dozen terabytes to many petabytes in a single data set (Watters, 2010). This continued rapid growth of data has imposed the need of new effective Knowledge Discovery and Data Mining (KDD) methodologies as the raw big data is not of direct use in the existing database management systems. The manual data analysis may be efficient if the total amount of data is relatively small, but is unacceptable in case of big data. The KDD helps to

automate the analysis the system and focus upon methodologies for extracting useful knowledge from data. That further assists the user to unearth the hidden facts and relations in the dataset to constitute wise decision that improve efficiency of the data analysis they perform. The revelation of such facts and relations can greatly improve the savings, efficiency, quality, and simplicity of a business.

Intelligent knowledge discovery from big data is a big challenge today that requires the federated endeavors of statistics, databases, pattern recognition, machine learning, data visualization, optimization, and high-performance computing in order to perform advanced analysis to enhance the business intelligence. It is observed that the scientific data oriented research is experiencing a potential transformation with the use of big data to obtain data-intensive decision-making at a new level of efficiency.

The White House also has recently announced a national "Big Data" initiative aiming to develop new core techniques and technologies to extract and use the knowledge from collections of large data sets to enhance the scientific research and innovation of managing, analyzing, visualizing, and extracting useful information from large, diverse, distributed and heterogeneous datasets (Weiss and Zgorski, 2012). This will increase the understanding of human and social processes and interactions and promote economic growth and improved health and quality of life.

APPENDIX

BATCH OF QUERIES EXECUTED ON NC-94 CLIMATE DATASET

```
/* Creation or Use of Storage - Storage Commands */

//$CyDB:>createrawstorage StorageConfig.xml;

$CyDB:>usestorage StorageConfig.xml;

//$CyDB:>formatStorage 16;

$CyDB:>startBufferManager 100;

$CyDB:>showdirectory;

$NC94:>OpenNC94Database NC-94.xml;

// Execution of NC94 Query with variables

$CyDB:>declare int $$i;

$CyDB:>declare string[5] $$a;


$CyDB:>set $$a[0] := $nc94:>SELECT *
RESTRICTED TO [[(C.MaxTemp + C.MinTemp)/2 < 0 ]]
FROM Climate C
WHERE NEIGHBOR(C.FIPS, 19169) = TRUE;


$CyDB:>set $$a[0] := $nc94:>SELECT *
RESTRICTED TO [[(C.MaxTemp + C.MinTemp)/2 < 0 ]]
FROM Climate C;



$CyDB:>set $$a[0] := $nc94:>SELECT *
INTO ClimateSub
DISPLAY Precipitation VIA MapCountyRanges.xml
RESTRICTED TO [[(C.MaxTemp+C.MinTemp)/2 < 0]]
FROM Climate C;
```

```
$CyDB:>set $$a[0] := $nc94:>SELECT *
INTO ClimateSub
DISPLAY MaxTemp VIA MapCountyRanges.xml
RESTRICTED TO [[(C.MaxTemp+C.MinTemp) > 1]]
FROM Climate C;



$CyDB:>set $$a[0] :=nc94:>SELECT *
RESTRICTED TO [[SELECT * FROM ClimateSub C1]]
FROM Climate C;

$NC94:>ParseQuery a[0];
$NC94:>DisplayParseTree a[0] xmlview;
$NC94:>DisplayParseTree a[0] graphicalview;


$NC94:>BuildExpressionTree a[0];
$NC94:>DisplayExpressionTree a[0] xmlview;
$NC94:>DisplayExpressionTree a[0] graphicalview;


$NC94:>ShowIterators;
$NC94:>OpenIterator a[0];
$NC94:>GetNextTuple a[0];
```

BIBLIOGRAPHY

Abraham T and Roddick J F, 1999. Survey of spatio-temporal databases. *GeoInformatica*, 3(1):61-99.

Alibrandi M, and Sarnoff M H, 2006. Using GIS to Answer the 'Whys' of 'Where' in Social Studies. *Social Education* 70(3): 138-143.

Bhargava G, 1989. A 2-dimensional temporal relational database model for querying errors and updates, and for achieving zero information-loss. *PhD thesis, Department of Computer Science, Iowa State University, Ames, Iowa.*

Chen C X, 2001. Data Models and Query Languages of Spatio-Temporal Information, PhD thesis, University of California, Los Angeles.

Chen C X, Kong J, and Zaniolo C, 2003. Design and implementation of a temporal extension of SQL, *In Proceedings of the 19th International Conference on Data Engineering*, 689-691, Bangalore, India.

Cox S, CuthBert A, Lake R, Martell R, 2003. Geography Markup Language Implementation Specification. *Open GIS Consortium Inc. http://www.opengis.org/techno/documents/02-023r4.pdf*

Cromley E K, McLafferty S L, 2011. GIS and public health. *Guilford Press*.

Dangermond J, 1969. Environmental Systems Research Institute [OL]. *http://www.esri.com/*

Demiryurek U, Banaei-Kashani F, Shahbi C, 2010. TransDec: A spatiotemporal query processing framework for transportation systems, *IEEE 26th International Conference on Data Engineering (ICDE, PP. 1197 -1200.*

ESRI, 1990. Shapefile [OL]. http://en.wikipedia.org/wiki/Shapefile

Erwig M, GÄuting R H, Schneider M, and Vazirgiannis M, 1999. Spatio-temporal data types: An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3):269-296, 1999.

Franklin C, 1992. An Introduction to Geographic Information Systems: Linking Maps to Databases. *Database*, 15(2), 12-21.

Gadia S K and Chopra V, 1993. A relational model and SQL-like query language for spatial databases. *In Advanced Database Systems, volume 759 of Lecture Notes in Computer Science*, pages 213–225. Springer-verlag.

Gadia S K, Chopra V, and Tim U S, 1993. An sql-like seamless query language for spatio-temporal data. *In Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, pages Q1-Q20, Arlington, Texas, USA.

Gadia S K., Gutowski W J, Al-Kaisi M, Taylor SE, and Herzmann, D, 2004. Database tools promoting extensive, user-friendly access to the Iowa Environmental Mesonet. *Baker Proposal.*

Gadia S and Nair S,1993. Temporal databases: A prelude to parametric data. *In Temporal Databases: Theory, Design, and Implementation, pages 28-66. Benjamin/ Cummings.*

Gadia S K and Nair S S, 1998. Algebraic identities and query optimization in a parametric data model for relational temporal databases. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):793-807.

Gadia S K and Vaishnav J H, 1985. A query language for a homogeneous temporal database. *In Proceedings of the 4th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pp. 51-56, Portland, Oregon, USA.

Gahegan M, 2005. Beyond Tools: Visual Support for the Entire Process of GIScience. Exploring Geovisualization. *J. Dykes, A. M. MacEachren and M.-J. Kraak. Elsevier Ltd.*

George K, 2005. Spatio-Temporal Database, lecture 16. Dept of Computer Science, Boston University.

Griffiths T, Fernandes A A A, Djafri N, and Paton N W, 2001. A query calculus for spatio- temporal object databases. *In Proceeding of the 8th International Symposium on Temporal Representation and Reasoning*, 101-110.

Grooves R, 2009. United States Census Bureau. *http://www.census.gov/.*

Güting R H, Schneider M, 2005. Moving Objects Databases. Academic Press. ISBN 978-0-12-088799-6.

Huang B, Jiang B, and Lin H, 2001. An integration of GIS, virtual reality and the Internet for spatial data exploration. *International Journal of Geographical Information Science,* 15(5), 439–456.

Huang B and Lin H, 1999. GeoVR: a web-based tool for virtual reality presentation from 2D GIS data. *Computers & Geosciences* [Online] http://citeseer.ist.psu.edu/huang99geovr.html.

IBM, 2012. What is big data ? Bringing big data to the enterprise. [Online] *http://www-01.ibm.com/software/data/bigdata/*

Jensen C S and Snodgrass R T, 1999. Temporal database management. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):464-497.

Jiyong Z, 2005. Spatio-Temporal Database – Doctoral Course: Conceptual Modeling. EPFL, Switzerland.

Kraak M J, 2006. Beyond Geovisualization, *IEEE Computer Graphics and Applications*, 26(4): 6-9.

Lorentzos N A and Mitsopoulos Y G, 1997. SQL extension for interval data. *IEEE Transactions Knowledge on Data Engineering*, 9(3):480-499.

MacEachren A M, Wachowicz M, Edsall R, and Haug D, 1999. Constructing knowledge from multivariate spatiotemporal data: Integrating geographical visualization and knowledge discovery in database methods. *International Journal of Geographical Information Science 13(4): 311-334*.

Mokbel Mohamed F, 2004. Continuous Query Processing in Spatio-temporal Databases, *In Proceedings of ICDE/EDBT Ph.D. Workshop*, pp. 119-128.

Narayanan V, 2009. A Workbench for Advanced Database Implementation and Benchmarking. Master's thesis, Department of Computer Science, Iowa State University, Ames, Iowa.

Noh Seo-Young, 2006. Hybrid Storage Design for NC-94 Database Within the Parametric Data Model Framework. *In Proceedings of the International Conference on Computational Science and Its Applications*, Part II, pp. 145-154, Glasgow, UK.

North Central Regional Association of State Agricultural Experiment Station Directors. Expected Outcomes, 2004. NC094: Impact of Climate and Soils on Crop Selection and Management. [Online]. Available: http://www.lgu.umd.edu/lgu_v2/pages/attachs/ 474_NC94ExpectedOutcomes.html.

Open Geospatial Consortium, 2000. Geography Markup Language [OL]. *http://en.wikipedia.org/wiki/ GML*

Open GIS Consortium, 1999. OpenGIS Simple Feature Specification for SQL. Edition 1.1. Shekhar S and Chawla S, 2003. Spatial Databases: A Tour. *Prentice Hall*.

Pelekis N, Theodoulidis B, Kopanakis I, Theodoridis Y, 2004. Literature review of spatio-temporal database models, *The Knowledge Engineering Review, v*.19 n.3, p.235-274.

Poth A and Greve K, 2001. deegree [OL]. http://www.giub.uni-bonn.de/deegree/

Robert H, 2012. It's time for a new definition of big data. [Online] *http://mike2.openmethodology.org/*

Schneider M, 2009. Spatial and spatio-temporal data models and languages, *In Encyclopedia of Database Systems,* pages 2681–2685.

Shahabi C, Banaei-Kashani F, Khoshgozaran A, Nocera L, Xing S, 2010. Geodec: A framework to visualize and query geospatial data for decision-making. *Multimedia, IEEE* 17(3), 14- 23.

Sharma S, Gadia S, 2010a. On Analyzing the Degree of Coldness in Iowa, a North Central Region, United States: An XML Exploitation in Spatial Database (NC94), *First International Workshop on Database Management Systems, India.*

Sharma S, Gadia S, and Goyal S B, 2010. On the Calculation of Coldness in Iowa, a North Central Region, United States: A Summary on XML Based Scheme in Spatial Database (NC94), *International Conference on Advances in Information and communication Technologies, India.*

Sharma S, Gadia S, 2010b. An XML-Based Range Variation Approach to Render the Coldness in Iowa, a North Central Region, United States, *Second International Conference on Information Management and Engineering, China.*

Sharma S, Gadia S, Kumar N, Narayan V, and Zhao X, 2010. Climate Analysis in IOWA using XML and Spatio-temporal Dataset-NC94. *International Journal of Database Management Systems (IJDMS)*.

Sharma S, Tim U S, Smith P, Gadia S, 2011. Geo-Spatial Patterns Determination for SNAP Eligibility in Iowa, *International Conference on Advances in Computing and Communication, India.*

Snodgrass R T, 1995. The TSQL2 Temporal Query Language. Kluwer.

Sripada L N, Lu Chang-Tien, and Wu Weili, 2004. Evaluating GML Support for Spatial Databases. *28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts,* vol. 2, pp.74-77.

Takatsuka M and Gahegan M, 2002. GeoVISTA Studio: a codeless visual programming environment for geoscientific data analysis and visualization. *Computers and Geosciences 28(10). pp. 1131-1144*.

Tansel A U, Clifford J, Gadia S K, Segev A, and Snodgrass R T, 1993. A glossary of temporal database concepts. *In Temporal Databases: Theory, Design, and* Implementation, Benjamin/ Cummings, pp. 92-109.

Tansel A U and Tin E, 1997. The expressive power of temporal relational query languages. *IEEE Transactions on Knowledge and Data Engineering*, 9(1):120-134.

Tarnoff P J, 1998. Evolution or control. *Traffic Technology International, pp.32-35.*

Toman D, 1997. Point-based temporal extension of temporal SQL. *In Proceedings of the 5th International Conference on Deductive and Object-Oriented Databases*, 103-121.

Tory M and Moller T, 2004. Human factors in visualization research. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):72.84.

United States Federal Government, 1994. Federal Information Processing Standard [OL]. *http://en.wikipedia.org/wiki/FIPS*

Vazirgiannis M, Wolfson O, 2001. A Spatiotemporal Model and Language for Moving Objects on Road Networks, *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, p.20-35.

Villars R L, Olofson C W and Eastwood M, 2011. Big data: What it is and why you should care. *IDC White Paper. Framingham, MA: IDC*.

Vilsack T, 2012. United States Department of Agriculture, National Agriculture Statistics Service. *http://www.nass.usda.gov/*

Wakefield J, Shaddick G, 2006. Health-exposure modeling and the ecological fallacy. *Biostatistics*, *7*(3), 438-455.

Watters A, 2010. The Age of Exabytes: Tools and Approaches for Managing Big Data (Website/Slideshare). Hewlett-Packard Development Company.

Weiss R, Zgorski L-J, 2012. Obama Administration Unveils "BIG DATA" initiative Announces $200 Million In New R&D Investments. [Online] *http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_press_release.pdf*

White T, 2012. Hadoop: The Definitive Guide. *O'Reilly Media*. p. 3. ISBN 978-1-4493-3877-0.

Wijk J J van, 2005. The value of visualization. *In IEEE Visualization*.