# INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.



Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

# Efficient Visualization and Querying of Geographic Databases

by

Min Ouyang

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professor Peter Z. Revesz

Lincoln, Nebraska

November, 2000

UMI Number: 9992001



UMI Microform 9992001



Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

DISSERTATION TITLE

Efficient Visualization and Querying of Geographic Databases

BY

Min Ouyang

SUPERVISORY COMMITTEE:

| | APPROVED | DATE |
|---|---|---|
| Signature | Peter Revesz | 11/27/2000 |
| Typed Name | Dr. Peter Revesz | |
| Signature | | 11/27/00 |
| Typed Name | Dr. Stephen Reichenbach | |
| Signature | Sharad Seth | 11/27/2000 |
| Typed Name | Dr. Sharad Seth | |
| Signature | Sunil Narumalani | 11/27/2000 |
| Typed Name | Dr. Sunil Narumalani | |
| Signature | | |
| Typed Name | | |
| Signature | | |
| Typed Name | | |

GRADUATE COLLEGE
UNIVERSITY OF NEBRASKA

# Efficient Visualization and Querying of Geographic Databases

Min Ouyang, Ph.D.

University of Nebraska, 2000

Adviser: Peter Z. Revesz

Geographic Information Systems (GISs) are attracting more and more interest. For geographically distributed data, value-by-area cartograms provide a highly expressive visualization. Continuously changing GIS spatiotemporal data can be animated by cartogram animation. We propose several value-by-area cartogram animation methods, and a new algorithm for creating single value-by-area cartograms. Our algorithms provide highly expressive animations for GIS spatiotemporal databases.

We describe an $O(n)$ time approximate algorithm, which can transform a sequence of $n$ time series data points into a linear constraint database. We also describe how the approximation enables more efficient cartogram animations, as well as more efficient evaluation of database queries. The approximate evaluation has high recall and precision on queries.

We also propose a new query optimization algorithm which is based on the hypergraph representation of queries. The optimization algorithm uses recursive bi-partitioning of the hypergraph to derive an optimized query evaluation strategy. This algorithm creates evaluation strategies that are easily parallelizable, which will be more efficient in parallel computers.

# ACKNOWLEDGMENT

I would like to express my sincerest gratitude to my advisor, Dr. Peter Z. Revesz for introducing me to this exciting research area. This dissertation would not have been possible without his guidance, patience and constant encouragement. I would also like to thank Dr. Stephen E. Reichenbach, Dr. Sunil Narumalani and Dr. Sharad Seth for having made available their time and commitment to serve on my dissertation committee.

I would like to express my gratitude to my colleague Rui Chen. It is a wonderful teamwork experience working with him. I learned a lot from discussions with him in both theoretical research and system implementation.

I would also like to take this opportunity to thank my friends and colleagues who have helped to make my stay in Lincoln so much fun.

I dedicate this dissertation to my parents for always supporting and encouraging me and for their unflinching faith in my abilities. Without their unselfish love and affection I would not have written this dissertation.

# Contents

# Chapter 1

# Introduction

Geographic Information Systems (GISs) are specialized computer systems for the storage, retrieval, manipulation, analysis and display of large volumes of spatial or map type data [36].

Many GIS systems contain spatiotemporal data such that the geographically distributed values change continuously over time. These continuously changing data will create infinite number of tuples in traditional relational databases. Constraint databases [27] provide a natural way to represent them such that these infinite values can be represented in finite constraint tuples.

There are now a number of implemented constraint database systems, such as CCUBE [3], DEDALE [10] and MLPQ [28]. The architecture of these constraint database systems are very similar. They all contain several special features which never occur in relational and object-oriented database system design. These fea-

tures may include modules for constraint representation, data approximation and special data visualization.

Value-by-area cartograms provide a highly expressive visualization of geographically distributed data. For continuously changing data, an animation can be done by successive displaying of value-by-area cartograms at different time instances. Such an animation can reveal more information than is revealed by only a few selected value-by-area snapshots. In this dissertation we describe several value-by-area cartogram animation algorithms that can be used to visualize geographically distributed continuous spatiotemporal data that often occur in GIS systems. We implemented the algorithms as part of the graphical user interface of the MLPQ GIS database system.

In GIS systems (and also many other systems), continuously changing data are often measured and recorded only sporadically as time series data. Since a fine granularity of time may be needed, the traditional representation of time series data, as a set of data points, requires too much computer storage space and allows only inefficient data retrieval and querying. We propose more efficient alternative representations for time series data: piecewise linear approximation. In the piecewise linear approximation, the time series data $s$ is approximated by a piecewise linear function $f$, such that at each time instance, the error between actual data and the approximate data is less than some fixed error tolerance.

Piecewise linear approximation is very important in animation because it provides interpolation of data, enabling the animation to be done in any time granularity. The data compression in piecewise linear approximation also helps the evaluation process to work faster.

Query optimization is very important in query evaluation [18, 24, 29, 32, 35]. Wong and Youssefi's query optimization algorithm [35] for relational algebra queries is a well-known optimization algorithm that forms the basis of several systems. In this dissertation we propose a new query optimization algorithm which is an improvement of Wong and Youssefi's algorithm.

## 1.1 Outline of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 gives a brief introduction on relational and constraint databases, as well as reviewing the previous work on cartogram animations and query optimizations.

Chapter 3 describes several methods for value-by-area cartogram animation and a new algorithm for creating single value-by-area cartogram. We present the algorithm and methods such that they can be used to construct a cartogram animation system that works fast enough to avoid pre-computing and saving of the snapshots. This enables us to animate databases that have a large number of snapshots. This chapter is based on our work of [20, 21].

Chapter 4 discusses the piecewise linear approximation for GIS time series data. We propose an $O(n)$ complexity algorithm that can compress the original time series data while guaranteeing certain maximal error tolerance. It is based on the work of [4, 5, 26].

Chapter 5 tries to combine Chapter 3 and Chapter 4 to explore the usage of piecewise linear approximation in the value-by-area cartogram animation. The interpolation and data compression abilities in piecewise linear approximation help to make more smooth and more efficient value-by-area cartogram animation.

Chapter 6 is based on [6]. It discusses the approximate evaluation of queries based on the piecewise linear approximation. We show that approximate evaluation can get high precision and recall while creating much less constraint tuples.

Chapter 7, based on [19, 22], proposes a new hypergraph partitioning based query optimization strategy whose evaluations require less space for the intermediate size relations than the optimized expressions given by earlier algorithms. It also gives parallel evaluation strategies which can be more efficient in parallel computers.

Finally, Chapter 8 concludes the dissertation with some directions for future work.

# Chapter 2

# Background

## 2.1 Relational and Constraint Database Systems

Database management system is a very important component of a modern computer system. In its early ages in the 1960s, the network and hierarchical models were widely used. Now, the relational database model has become the primary model for commercial databases. For relational database model, there are some theories that assist the design of databases and the efficient processing of queries.

The relational database model was first proposed by Codd in 1970. In relational database model, a database is a collection of one or more *relations*. Each *relation* has a unique name and can be represented by a table with rows and columns [29].

In relational databases, each relation consists of a **relation schema** and a relation instance. The **relation schema** describes the design of the relation (table).

such as the relation name, the name and domain of each field of the relation (table). A *relation instance* is a table as the instance of the *relation schema*. The following Table 2.1 gives a relation instance for the relation schema

$$CSE413\_Scores = (name, SSN, score)$$

which record the scores of each student in the course of "CSE413".

**Example 2.1.1** Relational Database table for relation $CSE413\_Scores = (name, SSN, score)$:

| name | SSN | score |
|---|---|---|
| Amy | 999-99-0001 | 90 |
| Billy | 999-99-0017 | 85 |
| Charlie | 999-99-0117 | 88 |
| Dick | 999-99-0227 | 92 |
| John | 999-99-3227 | 98 |

Table 2.1: Database for Relation $CSE413\_Scores$

Besides relational database representations, an alternative way to represent the data is to use constraint databases. The constraint data model provides a finite representation of the unrestricted relational data model. The essential difference is that in the constraint data model, the value of any attribute is specified implicitly using variables and constraints [27].

**Example 2.1.2** Suppose in a single day we record the precipitation in a city with area $4 * 10^{12}$ (square inches). It began to rain at 9:00am, 0.5 inch per hour and

the rain stopped at 1:00pm. At 7:30pm. it began to rain 0.75 inch per hour and stopped at 11:30pm. Then the amount of rainfall in this city (in cubic inches) is shown in Figure 2.1

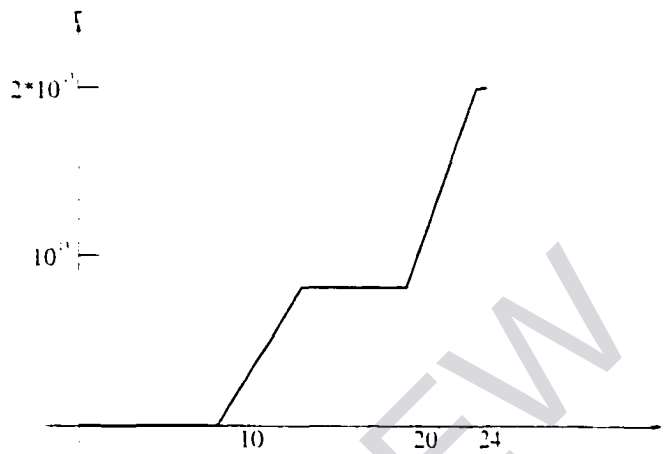


Figure 2.1: Amount of Rainfall

Suppose that the amount of rainfall at each time instance is represented in the following relation schema:

$$Amount\_of\_Rainfall = (time, rainfall)$$

In the unrestricted relational data model, a database instance could contain an infinite number of tuples. The rainfall relation contains an infinite number of tuples because the time instance can be any real number between 0 and 24, as shown in Figure 2.2.

The constraint relation for the above example is shown in Table 2.3. This

| time | rainfall |
|---|---|
| 00:00 | 0.0 |
| ... | |
| 10:00 | $2.0 * 10^{12}$ |
| ... | |
| 21:30 | $1.4 * 10^{13}$ |
| ... | |
| 24:00 | $2.0 * 10^{13}$ |

Table 2.2: Unrestricted Relational Database for Rain

constraint relation describes the same relation as in Table 2.2. However, the representation is different. In constraint relation, it uses a variable $t$ as the value of the time and the valuable $r$ as the rainfall. Both variables range over the real numbers. The first row of the of relation means that the rainfall is $r$ at the time $t$, if the condition $0 \leq t \leq 9.0, r = 0.0$ is true. We call such a condition a *constraint*. Each constraint can be a list of atomic constraints which are separated by commas that are read "and". The first constraint expresses that up to time 9.0 (9:00am), the amount of rainfall is 0. The next row expresses the rainfall is $2.0 * 10^{12} * (t - 9)$ when $t$ is between 9.0 (9:00am) and 13.0 (1:00pm). Other rows can be explained similarly.

Note that the entire constraint relation consisted of only five constraint tuples although it represents the infinite relation described earlier.

Having some databases in a computer system, a user can use query languages to request information from them. Relational algebra is one option, it consists of a set

| time | rainfall | |
|---|---|---|
| t | r | $0 \leq t \leq 9.0. r = 0.0$ |
| t | r | $9.0 < t \leq 13.0. r = 2.0 * 10^{12} * (t - 9)$ |
| t | r | $13.0 < t \leq 19.5. r = 8.0 * 10^{12}$ |
| t | r | $19.5 < t \leq 23.5. r = 8.0 * 10^{12} + 3.0 * 10^{12} * (t - 19.5)$ |
| t | r | $23.5 < t < 24.0. r = 2.0 * 10^{13}$ |

Table 2.3: Constraint Database for Rainfall

of operations that take one or two relations as input and produce a new relation as their result. The fundamental operations in the relational algebra are *select*, *project*, *union*, *set difference*, *Cartesian product* and *rename*. other operations include *set intersection*, *natural join*, *division* and *assignment* [29].

In relational algebra, we use lowercase Greek letter sigma ($\sigma$) to denote selection, the uppercase Greek letter pi ($\Pi$) to denote project, $\cup$ for union, $-$ for set difference, $\times$ for Cartesian product and $\rho$ for rename. We also use $\cap$ for intersection, $\bowtie$ for natural join and $\div$ for division. In this way, the query of finding the students whose scores are greater than 90 can be written as:

$$\Pi_{name}\ \sigma_{score>90}\ CSE413\_Scores \tag{2.1}$$

Relational algebra is a formal query language. However, commercial database systems usually use SQL language. SQL is not only a "query language", it contains many capabilities besides querying a database. Including features for defining

the structure of the database, modifying the database and for specifying security constraints [29].

The following is a SQL query that is equivalent to relational algebra in 2.1.

$$\begin{array}{ll} \textbf{Select} & name \\ \textbf{from} & CSE413\_Scores \\ \textbf{where} & score > 90 \end{array} \tag{2.2}$$

Another query language is Datalog, which is a rule-based language related to Prolog. In Datalog each rule is a statement saying that if some points belong to some relations, then other points must also belong to a defined relation. Each Datalog query contains a Datalog program and an input [28]. The Datalog query for the previous database retrieval problem can be written as:

$$High\_Scores(name) : -CSE413\_Scores(name, SSN, score),$$

$$score > 90.$$

Both the constraint databases and the relational databases can be queried by either relational algebra, SQL or Datalog.

## 2.2 Cartograms and Cartogram Animation

Many GIS databases contain spatiotemporal data such that the geographically distributed value change continuously over time. Population and precipitation

distributions are two such examples. Value-by-area cartograms provide a highly expressive visualization for this kind of data. A value-by-area cartogram is created by dividing the original map into small areas (cells) and then enlarging or shrinking each cell to make its area proportional to its given "value". For example, a value-by-area cartogram where the cells are the continental U.S. states and the values are their populations in 1990 is shown in Figure 2.2. the population data comes from the US Census Bureau http://www.census.gov. In this value-by-area cartogram, each state's area is proportional to its population. By looking at the map, we can easily see the population distribution in the continental United States.



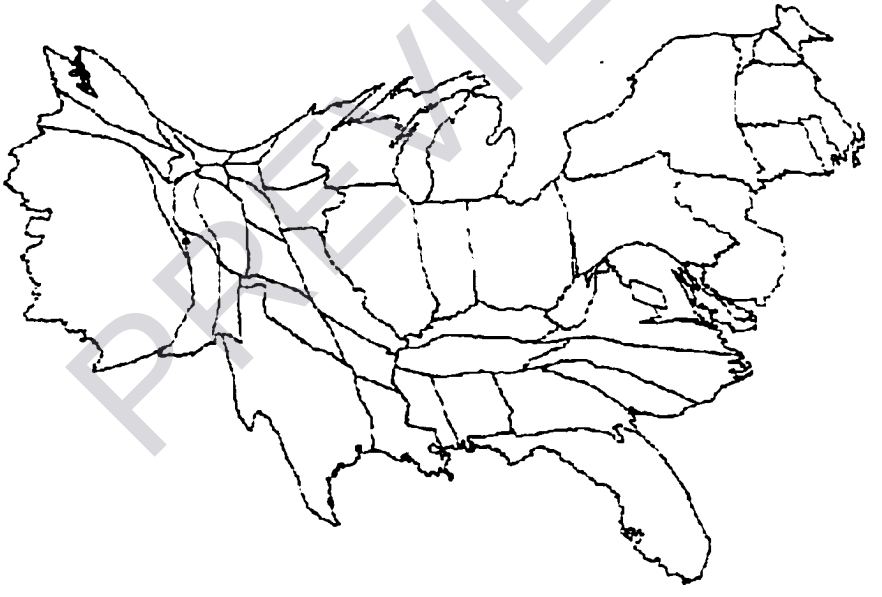

Figure 2.2: A Value-by-Area Cartogram for the U.S. Population in 1990

For the same set of data, there are several possible value-by-area cartograms.

In general, the easier it is to recognize which cells are which in a value-by-area cartogram the better (and more informative) it is. For example, for anyone familiar with the usual geographical map of the continental U.S. states, it is easy to recognize the individual states. In that way, one can learn that for example California has a high population compared to its area as it is enlarged, which Nebraska has a small population compared to its area as it is shrunk compared to the usual geographic map. It is in this way that cartograms can be informative and useful for people who do not wish to look at statistical tables.

There are two basic forms of cartograms: contiguous cartograms and noncontiguous cartograms. In contiguous cartograms, the internal cells are adjacent to each other, while in noncontiguous cartograms, this is not necessary the case. Figure 2.3 illustrates the original map with four cells, each has the geographic distributed value with it, the contiguous cartogram and the noncontiguous cartogram whose cell areas are proportional to the its "value".

Generally, contiguous cartograms looks better than noncontiguous cartograms because they look more like real maps, the geographic arrangement of cells also helps people to recognize the cells with respect to original geographic map. On the other hand, it is easier for noncontiguous map to preserve original shape of each cell, and they are easier to construct. Because if we do not care the continuity of the cartogram, we can simply enlarge/shrink each cell to its desired area.

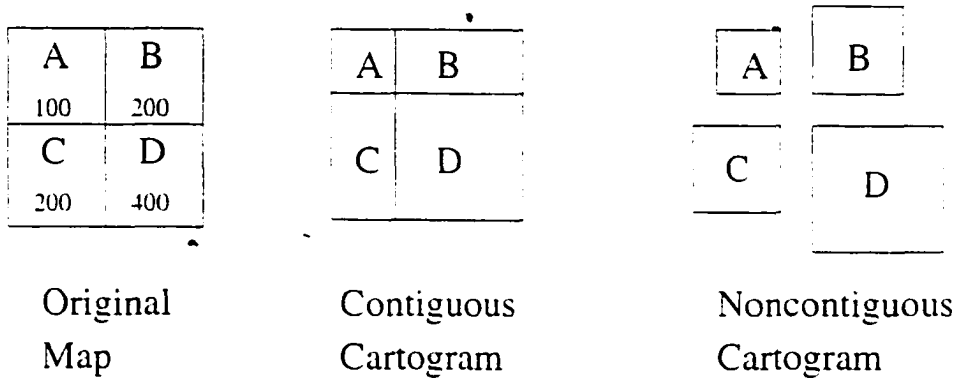


Figure 2.3: Contiguous and Noncontiguous Cartograms

Contiguous cartograms are much more difficult to construct. Usually the cells are represented by polygons. When trying to construct contiguous cartograms, each cell has to be enlarged/shrunken. The enlarge/shrink requirement for each cell may give conflict move direction for cell corner vertices. The cell shape has to be somehow distorted to keep the cartogram contiguity.

Cartograms can be drawn manually. However, computer programs are more desirable. From 1973, a few computer algorithms for creating contiguous value-by-area cartograms have been proposed [7, 13, 11, 30]. Among them, the pseudo-cartogram algorithm proposed by Tobler [30] is the earliest. Generally, pseudo-cartogram algorithm converges slowly (if it converges at all), and the area error is large.

Dougenik et al.'s algorithm [7] and Gusein-Zade et al.'s algorithm [11] are based on rubber-sheet transformation idea, which can give highly accurate cartograms.