Relational Database Framework for GeoSpatial Data Sets

by

RUEBEN CHINYAKATA

(R892050L)

A thesis submitted in partial fulfillment of
the requirements for the degree of

Masters of Science in Computer Science

Department of Computer Science
Faculty of Science
University of Zimbabwe

FEBRUARY 2006

Abstract

Geographic Information Systems (GISs) have emerged as the most powerful and effective decision-making tools in terms of gathering, analysis, interpreting, distribution and using geographically referenced (Spatial) data for use in a wide variety of application domains, such as urban planning, natural resource management, telecommunication network management, vehicle navigation, etc. GIS now embraces a broad range of disciplines including Surveying and Mapping, Remote Sensing (RS), Global Positioning Systems (GPS) among others.

However, the main drawback of GISs is the weak link with Relational Database Management Systems (RDBMS) because present day Geospatial databases are file based. Without formal mechanisms for data sharing, much time and resources are wasted in duplication of efforts and digital data conversion processes. The relational model is the predominant and well established data model, which is used to implement many commercial relational DBMS packages and application systems.

This dissertation presents principles, methods and mechanisms for the development of GeoSpatial Databases based on the relational database model. The research also presents a relational database framework that accommodates geospatial data sets and algorithms for spatial data analysis. The research proposes relational algebra extensions to include these spatial functions before.

This thesis will ultimately lay the foundation for the development of fully relational geospatial database systems.

.

Acknowledgements

I'd like to thank my supervisor, Mr. Kachepa, for the professional guidance and also for being so easy to work with. I'd also like to thank Dr G. Hapanyengwi, Mr Chikasha, Mr Nyambo, Mr Museba, and my colleagues Tichaona Dhliwayo, Aaron Masina and Mhonda Sibanda for their valuable help. I'd like to thank the whole 2nd MSc intake class who made the last two years so enjoyable.

Special thanks goes to my wife Caroline and the boys Kumbirai and Kundai.

Lastly this is a special dedication in memory of my late loving brother, Thomas (Changamire).

CONTENTS

Tables

Figures

Appendices

1.0 Introduction

1.1 Background to the Research

A Geographic Information System (GIS) is a computer based information system that is used to digitally represent and analyse geographic features present on the earth's surface and the events that take place (Dr Ashok Kumar Sinha and Surekha Dudhani, 2005).

Geographic Information Systems have recently emerged as the most exiting and powerful decision making tools in the information world due to their ability to integrate, manipulate and display a wide range of information to create a picture of an area's geography, environment and socio-economic characteristics (Health Geomatics, 2004). Currently, Geographic Information Systems usage range from basic mapping to supporting resource exploration and development, healthcare, transportation, telecommunication, environmental management, urban development, land use, amongst others.

In recent years the use of GIS, also referred to as Geotechnology, has emerged as the fasted growing technologies with a market value that was expected to exceed US$30bn by the end of 2005 up from US$6bn in 2004 (Batty, July 2004). This growth has captured the attention of two IT market leaders, Oracle and Microsoft. Oracle and Microsoft released Oracle 10g and Multipoint respectively and these products have raise considerable interest amongst leading GIS vendors.

1.1.1 GIS View of the World

GeoSpatial databases are database systems used by Geographic Information Systems to store data on objects located on the earth's surface and their associated attribute information. Geographic Information Systems create a representation of a map/region in the form of layers, also called thematic maps, connected by a common frame of reference (Grid Reference) as shown on Figure 1 below.

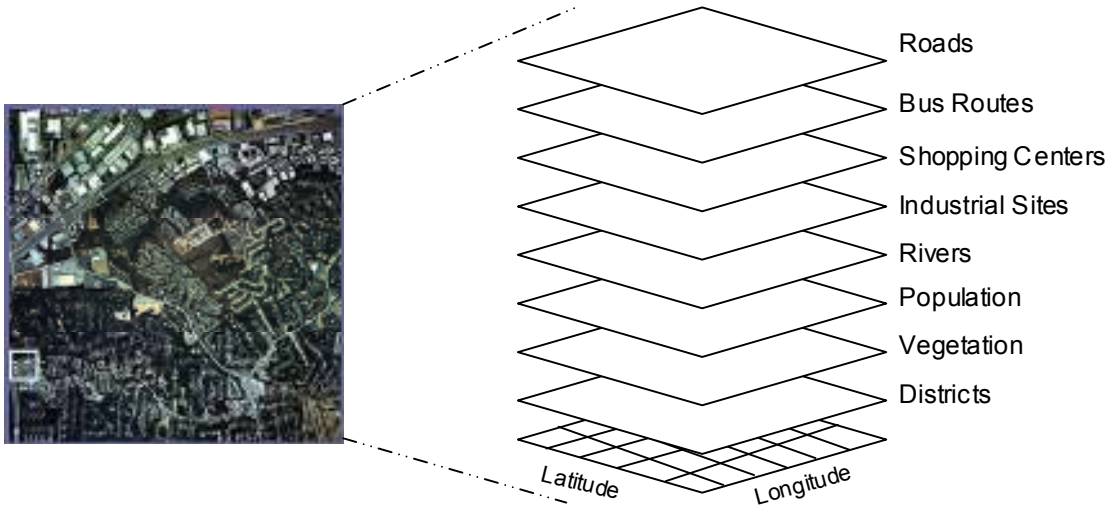

Figure 1: Separating data into layers *(Source: Health Geomatics, 2004)*

Each of these thematic maps (also referred to as a layer, coverage or level) is carefully overlaid on the others so that every location is precisely matched to its corresponding locations on all the other maps. The bottom layer is the most important since it represents the grid of a location reference system, such as latitude and longitude applying to all the layers. This allows different layers to be compared or analysed in combination. For example the accessibility of a winter crop in a particular district can be analysed

1.1.2 GeoSpatial Data Sets

Geospatial data is data is that explicitly references a geographic position on the earth's surface. A geospatial database contains two main sets of data namely spatial[1] data as well as non-spatial data (also referred to as Attribute data). Each feature on a layer has a unique identifier which distinguishes it from the rest of the features and it helps to link the feature to external systems.

Attribute data refers to properties of spatial entities as shown on Table 1 below.

Table 1: Example of Attribute Data

Road	Surface Type	Year Completed
M1	Tarmac	1951
A2	Gravel	2000
A3	Dust	1988

Spatial entities (also referred to as spatial objects or features) are natural, man-made or abstract objects of interest. These objects are described by geometrical (position and shape of the feature), topological (relations to other features) and attribute (all other non-spatial, usually numerical and textual) data. Spatial data representation and management have always been the primary concern in GIS research.

[1] Herein also referred to as GeoSpatial data

GeoSpatial Databases are designed to deal with objects in space that have identity and well defined extents and location. They are designed to work with a very large number of simple geometric objects (points, lines and polygons).

The information contained in a GeoSpatial Database is stored in the form of digital co-ordinates, which describes spatial features on a map. These are in the form of points (location), lines (roads or rivers) or polygons (districts or plots). The data is normally stored as different sets each representing a theme (or layer or coverage). These layers can be combined in a number of different ways for analysis or map production.

1.1.3 Sources of GeoSpatial Data

There are two main sources of GeoSpatial Data namely:

- **Arial Photographs**: these are images captured from a position above the earth's surface
- **Satellite Images**: these are collected by sensors aboard a satellite and then relayed to the earth as a series of electronic signals, which are processed by a computer to form an image.

1.1.4 Problems of GeoSpatial Database

Historically most GeoSpatial Databases have been file based, as the performance of commercial Relational Database Management Systems (RDBMS) did not make them a viable option (Dehua Zhao, Byunggu Yu, Dan Randolph, and Bong H. Hong, 2004). This means that GeoSpatial databases are currently not taking full advantage of relational database features such as backup and recovery, replication, data sharing, redundancy control etc.

Data Integration is a major problem in many GIS project with an average 40% of implementation budget going into integrating the various spatial and non-spatial components (P Batty, 2005). Modern integrated Lands Information Management System (LIMS) are comprised of many subsystems as shown on Figure 2 below. The GIS module forms a small but very important component that should be integrated to the broader LIMS. Such an integration can only be made simple if GIS databases are developed on the same relational database platform as the rest of the systems.

The current spatial query language is not rich enough to extract interesting patterns between the spatial and non-spatial data.

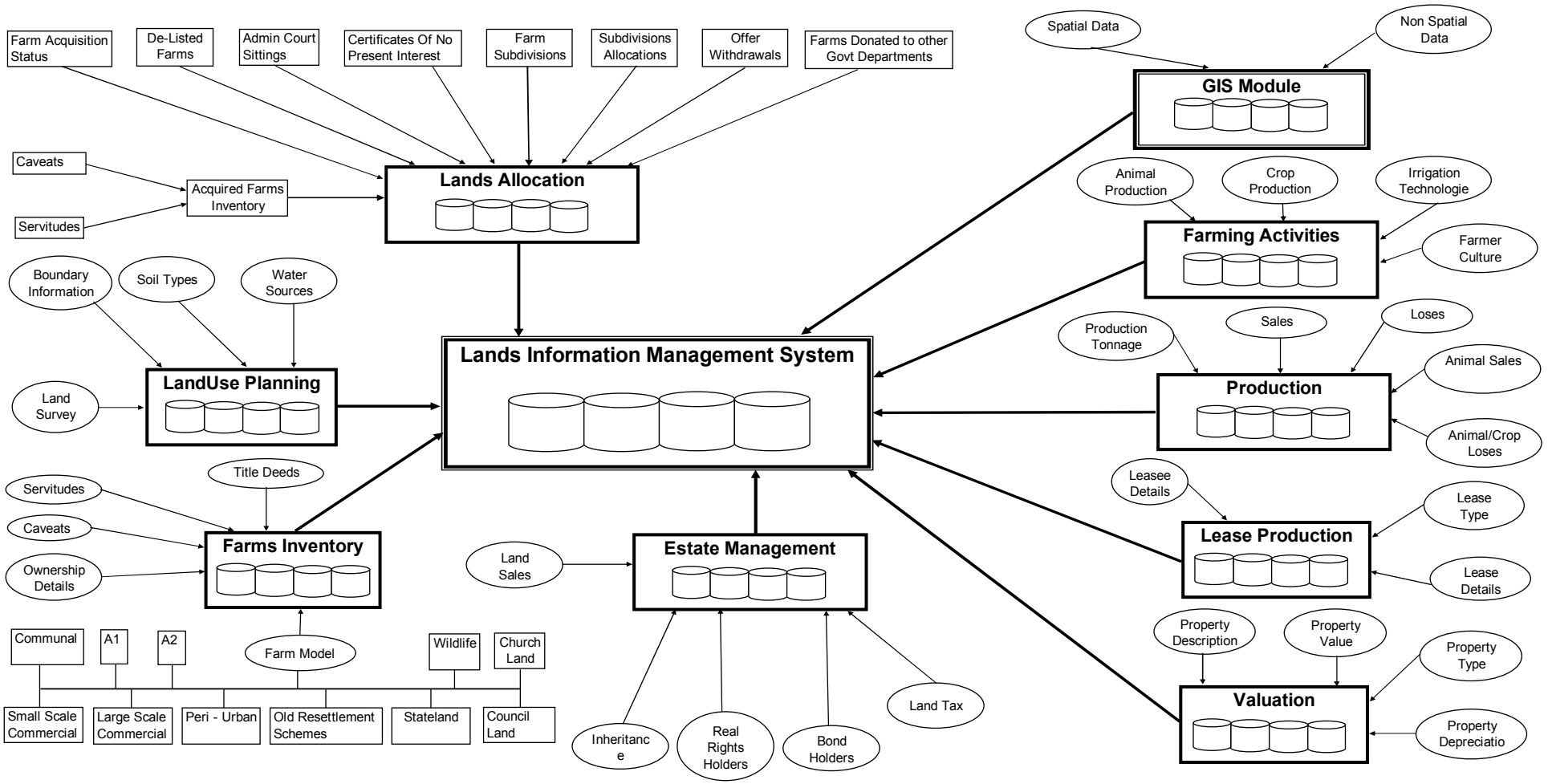

Figure 2: Relationship between GIS's and Relational Database Systems

1.2 Statement of the Problem

The main aim of this research is to develop a unified relational database framework that accommodates 2 Dimensional GeoSpatial data sets.

The research lays the foundation for the development of Geospatial databases on a relational database platform to promote interoperability as well as eliminate a number of problems as specified on section 1.1.4 above.

GeoSpatial data sets include location, shape, size, and orientation of objects on the earth's surface. By default, GeoSpatial data has some type of attribute (non-spatial) data associated with it that need to be stored such as shown on Figure 3 below. Non-spatial data (also called *attribute* or *characteristic* data) is that information which is independent of all geometric considerations.

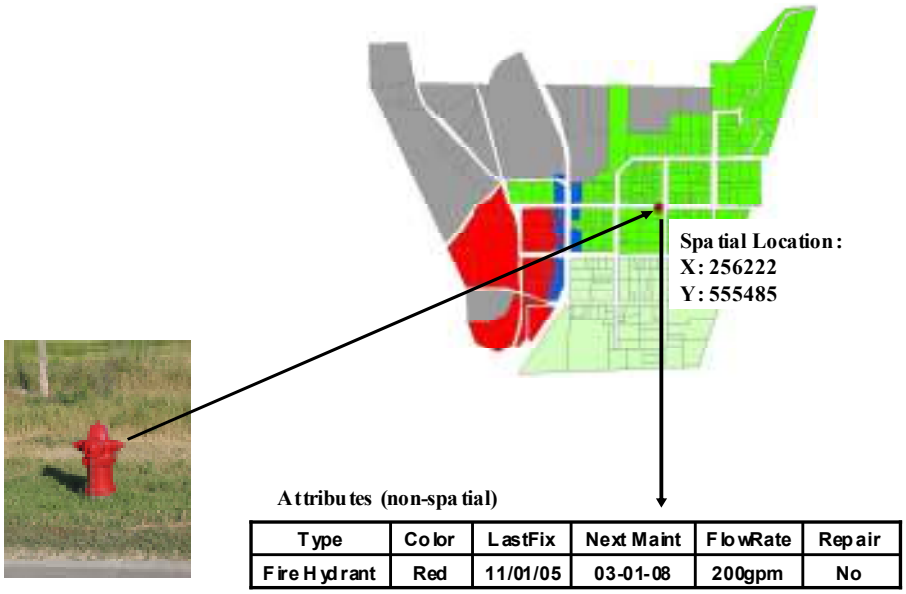

Figure 3: Spatial and Attribute data sets *(Source : Daniel P. Ames 2005)*

1.3 Research Objectives

The 3 main objectives of this research are to:

i. Define a relational database framework for GeoSpatial data sets. This includes the following:
   - Presentation of a formal definition of GeoSpatial objects
   - Representation of the geospatial objects using the Relational Model
   - Definition of Spatial Constraints

ii. Develop Algorithms for Spatial data analysis namely:
   - Basic Measurements
     - Point to point distance
     - Area of a polygon
     - Length of a line
     - Circumference of a polygon
   - Proximity determination
     - Point to line distance
     - Nearest Neighbour

iii. Develop a Query specification that allows for the extraction of useful information from the relational GeoSpatial framework. This entails:
   - Spatial Relational Algebra
   - SQL Extensions
   - Query Optimisation

1.4 Significance of the Study

This research facilitates the deployment of geospatial data in the well-established relational database platform thereby eliminating problems outlined on section 1.1.4 above. This brings several advantages including:

- Data Independence :- Applications become data independent so that multiple applications can use the same data and they can evolve separately over time
- Concurrency control :- This enables support for multiple concurrent applications
- Redundancy: Collecting data at a single location reduces redundancy and duplication
- Maintenance costs decrease because of better organization and decreased data duplication
- User knowledge can be transferred between applications more easily because the database remains constant
- Data sharing: Data sharing is facilitated and a corporate view of data can be provided to all managers and users. This makes it easy to integrate spatial data with other applications
- Backup and recovery: valuable data in a database is protected from system failure and incorrect update. Software utilities are provided to backup all or a part of a database and to recover the database in the event of a problem
- Database administration tools :- Specialized collection of tools are provided for setting up the structure of a database (the schema) creating and maintaining indexes, turning to improve performance, backing up and recovering, and allocating user access rights is performed by a database administrator.
- Security and standards for data and data access can be established and enforced

This research ultimately enables advanced knowledge discovery methods from geospatial databases to be developed. Combinations of spatial and attribute queries can build complex and powerful operations including:

- Proximity determination
- Potential flood hazards
- Distance planning (point to point distance and point to line distance)
- Calculation of area under crop, deforested areas, area of water bodies, etc

1.5 Scope of the Study

The scope of this research covers representation of Geometric data in the relational database model, spatial algorithms, and spatial querying.

The scope of each of the above objectives will include:

i. Definition of a relational database framework for GeoSpatial data sets will cover points, Multipoints, Polylines and Polygons
ii. Development of Algorithms for Spatial Functions to manipulate geometric objects will cover:
    - Point to point distance
    - Area of a polygon
    - Length of a line
    - Circumference of a polygon
    - Point to line distance
    - Nearest Neighbour

iii. Development of a Query specification that allows for the extraction of useful information from the relational GeoSpatial framework will cover:

- Specify extension of Select, Project and Join operations to include Spatial Functions and specify examples of SQL Extensions.

1.6 Limitations of the study

This research concentrates on the development of a relational database framework that accommodates geospatial data sets and the tool that can be used to manipulate geometric objects. The study, however, does not address various related areas such as the handling of altitude/terrain. Distance measurement is assumed to be point to point straight line distance and does not necessarily consider the fact that the earth's surface if curved.

The study also does not allow for the creation of query windows to allow users to query only a selected area of a map.

1.7 Thesis Organisation

The remainder of this dissertation is organised as follows: Chapter 2 presents the Literature Review. Chapter 3 outlines the methodology used in the research. Chapter 4 presents results of this research where as Chapter 5 discusses the findings. Chapter 6 presents conclusions and recommendations

2.0 Literature Review

This chapter reviews previous work done by different researcher in the area related to the development of Geospatial databases.

2.1 Challenges in the Development of Relational GeoSpatial Databases

In a paper entitled, Issues in Spatial Databases and Geographical Information Systems (GIS), Hanan Samet (2003), explored the following as challenges in the development of relational GeoSpatial Databases:

- Incorporation of Geometry into database queries without users being aware of it (seamless integration of spatial and non-spatial database)
- Expression of spatial integrity constraints
- Spatial Query Optimisation
- Graphical Query Language (output must be visual)
- Incorporation of imagery
- Spatial Relational Algebra
- GIS on the Web and distributed data and algorithms
- Knowledge discovery
- Interoperability (ability of GeoSpatial Databases to share data with other systems)

In another paper entitled, GIS Databases are Different by Peter Batty and Richard G. Newell (2005), the authors debated the suitability of standard commercial databases management systems for use in GIS. The paper presented fundamental challenges in the development of GeoSpatial databases based on relational database model including:

- **Performance:** Relational databases handle a very large number geometric object generally requiring more advanced search algorithms for improved performance.
- **Long Transactions**: The inability of database systems to handle long transactions as is required by GIS databases. Relational Databases are designed for short transactions e.g. bank transactions that take a few seconds to execute. GIS systems handle long transactions like the digitisation of a river that may take a day of two. Relational Database Management Systems (RDBMS) eliminate long transaction by checking out the transactions. But however, in GIS Systems, the user might want a private working space before his work is posted to the database. There may also be need for collaborative work which is not possible with current RDBMS.

The paper by Peter Batty and Richard G. Newell (2005), also explains that even if GeoSpatial databases are developed using the relational paradigm, integration with external systems will continue being a problem because spatial data types are still not explicitly supported by most external systems. The database system has to use complex record structure and retrieval algorithms for geographic data which means that it cannot be easily read or updated by external applications.

2.2 The need for interoperability

The white paper by ERSI (2003) entitled, Spatial Data Standards and GIS Interoperability, argues the need for Geospatial databases to be open to allow for the sharing of data. The paper explains that since GIS brings together disparate data sets there is need for the development of a framework for shared spatial data infrastructure and also the development of standards.

2.3 GeoSpatial Data Sets

The authors Dehua Zhao, Dehua Zhao, Byunggu Yu, Dan Randolph, and Bong H. Hong (2004) define the following GeoSpatial data sets that make up a basic map:

- **Point**: A *Point* object represent a single location in space such as a city centre, or any object on the ground. Point objects may also represent conceptual objects as well such as a potential marketing place or a location where there was a reported outbreak of a disease.
- **Multipoint**: A *Multipoint* represents an unordered number of points representing a conceptual object or group.
- **Polyline**: A *Polyline* is a sequence of two or more connected points (vertices) with linear interpolation between points. In each part, each consecutive pair of points defines a line segment. It represents roads, rivers, power lines, sewerage pipes etc.
- **Polygon**: A *Polygon* is defined as a set of three or more connected points (vertices) that form a simple and closed loop. Polylines represents dams, land parcels, districts, provinces, etc

2.4 Geometry

2.4.1 Definition of Polygons

The author Tom Davis (2000), in a paper simply entitled, Polygons, presented the following formal definitions of polygons and their properties:

- **Definition 1 (Polygon)**: Let $v_0, v_1,\ldots,v_n$ where $n\geq 3$ be a set of points in a plane such that $v_0 = v_n$ .The union of line segments $v_0v_1, v_1v_2,\ldots,v_{n-1}v_n$ is an n-sided polygon whose vertices are $v_1, v_2,\ldots$ and whose edges are the line segments $v_0v_1, v_1v_2,\ldots$. (Note that since $v_0 = v_n$, the polygon is automatically closed). The vertices of a polygon can be defined as $v_i = (x_i,y_i)$
- **Definition 2 (Jordan Polygon)**: A polygon P with vertices $\{v_i : 0 \leq i \leq n \}$ is said to be a Jordan Polygon if:
  i. The vertices v_i for $0 \leq i \leq n$ are distinct
  ii. Two distinct edges intersect only if they have a common vertex, and they intersect only at a common vertex.

  This definition emphasis that the polygon does not intersect itself and divides points on a plane into two regions called the inside and the outside. The inside has a bounded area and the outside is unbounded.
- **Definition 3 (Orientation)**: A Jordan Polygon is said to be oriented counter-clockwise if, as you move along the edges from v0 to v1, et cetera, the inside of the curve is to your left. Otherwise the polygon in oriented clockwise
- **Definition 4 (Convexity):** A Jordan polygon is said to be convex if the line segment connecting any two interior points is completely contained within the interior of the polygon. Any polygons that do not satisfy this condition are said to be non-convex, or concave
- **Definition 5 (Star-Shaped**): A Jordan polygon is said to be star-shaped if there exists a point p in its interior such that the line segments connecting p with any other point in the interior of the polygon lie completely in the interior of the polygon

2.5 Oracle Spatial

Oracle Corporation announced its first Spatial Relational product in August 2005, called Oracle Spatial (Jean Ihm, Xavier Lopez, August 2005). Oracle Spatial is an option on its latest Oracle Database 10g Enterprise Edition. The product has analysis features including spatial aggregate functions, area and length calculations, and linear referencing on countries on the world map.

2.6 Microsoft Location Server

Microsoft Location Server is a product that enables cell phone based location mechanism, which can typically track somebody to higher degree of accuracy of within a couple of centimeters.

2.7 Critique of the Previous Approaches

Geographic Information Systems (GISs) technology is evolving beyond traditional GIS community and is rapidly becoming an integral part of the information infrastructure in many organisations (ERSI, 2003). In order for organisations to fully realize the benefits of GIS technology, spatial data need to be shared and systems need to inter-operate.

The main advantage of developing GeoSpatial databases based on the relational model is that it allows the GIS vendor to concentrate of developing GIS functions (Peter Batty, 2005). This approach exploits most well tried and tested features of relational database management systems such as security, backup and recovery, etc.

This section analyses previous work as sited above. It should however, be noted that there has been several authors including corporations (Oracle, August 2005) who approached the problem of developing GeoSpatial databases based on the relational paradigm.

This thesis will propose a relational database framework for Geospatial data sets before exploring various challenges as outlined by Han Samet (2003) including:

- Incorporation of Geometry into database queries without user being aware of it (seamless integration of spatial and non-spatial database)
- Expression of spatial integrity constraints
- Spatial Relational Algebra
- Interoperability (ability of GeoSpatial Databases to share data with other systems)

2.7.1 GIS Interoperability

The paper by ERSI (2003) clearly spells out the need for the Geospatial databases to be open thereby allowing interoperability. The main aim of this research is to develop a framework that paves the way for the development of relational geospatial database capable of storing both spatial and attribute data. This will ensure that the database will be open to connectivity with external systems

2.7.2 Performance

The argument as presented above on performance is no longer an argument because recent advances in hardware and software performances have made the development of GeoSpatial databases based on the relational model a reality. Current RDBMS can now handle data warehouses which are several terabytes large.

2.7.3 Spatial Operators

The authors Dehua Zhao et al (2004) did not discuss spatial operations that can be borrowed from geometry since GeoSpatial databases handle very large numbers of geometric objects. Some of the most important spatial operators not discussed include:

- **Point to Point Distance**: This operator calculates distance between two points objects
- **Point to Line Distance**: This operator calculates the minimum distance from a point to a line (rivers, roads, power lines, etc
- **Nearest Neighbour**: Defines the nearest objects that satisfies a particular criteria
- **Intersect**: Computes the intersection of two lines (rivers, roads, telephone lines etc) or the point where a river crosses a boundary line
- **Area of a Polygon:** Calculates the area of a land parcel, or dam, etc
- **Perimeter:** Computes the distance round a polygon object
- **Direction:** this is a spatial operator used to locate objects on a map
- **Inclusion:** determines whether an object is within a given polygon

The success of Relational GeoSpatial databases depends on efficient spatial analysis algorithms taking into consideration the enormous number of geometrical objects that have to be handled. It is therefore, very important to explore these algorithms for efficiency.

2.7.4 Spatial Querying

The Ralf Hartmut Güting (2004) explored a spatial query language that is an extension of the Structured Query Language (SQL). The article is not comprehensive enough since the author did not specify the list of spatial operators in the proposed extensions. The author did not propose a representation of spatial objects using the relational model or various spatial functions that are necessary for manipulating geometric objects. He did not also address the need for various spatial functions to manipulate geometric objects

2.8 Existing Products

Leading database vendors Oracle and Microsoft are adding innovations designed to facilitate storage and retrieval of geospatial data. Their current offerings are initial releases addressing a new market area for these companies. The vendors will be expected to refine their offerings considerably within the next few years. This research is therefore, very important in the improvement of these products.

3.0 Research Methodology

The following outlines the methodology used in this research:

i. Entity Relationship Diagrams (ERDs) were used to graphically express each geospatial object's data model and then derive its representations as relations.

ii. Formal definitions of Spatial objects were used to define Spatial constraints to be enforced to ensure spatial data integrity

iii. Algorithms for spatial functions were then developed using standard algorithm design techniques especially the Divide and Conquer and the Greedy approaches

iv. The next step was to express Spatial operations (Select, Project and Join) incorporating various spatial functions as defined in the previous step.

v. An extension of the existing Structured Query Language was then proposed to allow for spatial joins as well as the inclusion of spatial functions defined above

4.0 Results of Findings

4.1 Relational Database Framework for GeoSpatial data sets

This section presents a relational database framework that accommodates GeoSpatial data sets. These are first represented using Entity Relationship Diagram (ERD) and ultimately as a relations that require normalization depending on attributes selected and their functional dependencies.

4.1.1 Points

A point object represent an object that has a single location like City, Polling Station, School, Hospital or a place where a particular event occurred (e.g. outbreak of a disease, etc). Point objects are defined by Key Attributes that distinguishes the point object from all others as well other non spatial attributes that help define the point. But most importantly point objects have a particular location determined by their < *x-coordinate, y-coordinate*>. Figure 4 shows the representation of point objects using an Entity Relationship Diagram.

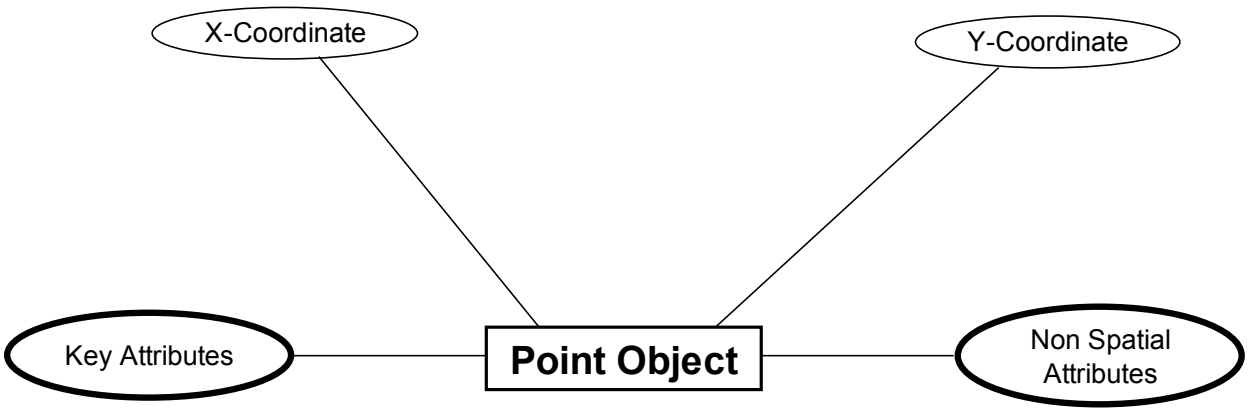

Figure 4: ERD representation of a Point Object

A relation that represent a point object has schema POINT_OBJECT(Key_Attribute(s), Non_Spatial_Attribute 1,…, Non_Spatial_Attribute N, x-coordinate, y-coordinate). This schema requires normalization depending on functional dependencies.

Examples:

Table 3 below shows the representation of a point object, City, as a relation with the City_ID as the Primary Key and non-spatial attributes City_Name, Population and Area as well as spatial attributes X-Corordinate and Y-Coordinate.

Table 2: Representation of City Point object as relations

City_ID	City_Name	Population (m)	Area (Ha)	X-Coord	Y-Coord
C01	Harare	2	800000	3000	2300
C02	Bulawayo	1	900000	10102	1111
C03	Mutare	0.8	280000	2014	2577
C04	Gweru	0.6	300000	8025	902

4.1.2 MultiPoints

A *Multipoint* objects represents an unordered number of points representing a conceptual object or group. A multipoint object is stored as $\{<x_1,y_1>,<x_2, y_2>,\ldots,<x_n, y_n>\}$ where $n \geq 1$

A multipoint object represents for example, outbreak of a disease, potential marketing or strategic places. Point objects are defined by Key Attributes that distinguishes the place from all others as well other non spatial attributes that help define the place. But most importantly point objects have a particular location determined by their <x-coordinate, y-coordinate>. In multipoint features these points are further organised into groups or sets.

Figure 5 below shows the representation of multipoint objects using an Entity Relationship Diagram.

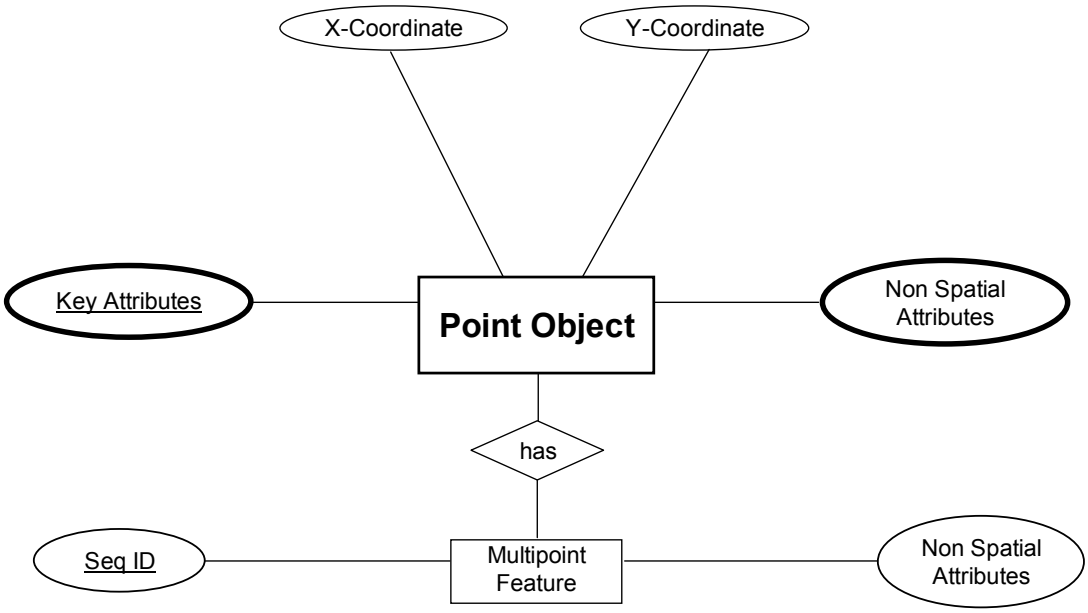

Figure 5: ERD representation of Multipoint Objects

The relation schema of the multipoint objects is as follows:

MULTIPOINT_FEATURE(Seq_ID, Non_Spatial_Attribute 1,…, Non_Spatial_Attribute N)

POINT_OBJECT(Seq_ID, Key_Attribute(s), Non_Spatial_Attribute1, … ,

Non_Spatial_Attribute N, x-coordinate, y-coordinate).

Tables 3 and 4 shows the representation of multipoint disease objects

Table 3: Disease relation that has multiple cases reported

Seq_ID	Name	Remarks
01	Malaria	Caused by mosquitoes
02	Cholera	Caused by poor hygiene

Table 4: Disease outbreak sites

Seq_ID	Centre_ID	Centre Name	X-Coord	Y-Coord
01	01	Cheziya	11	2
01	02	Nembudziya	22	50
02	01	Serima	3	14
02	02	Charthworths	8	1
02	03	Zimuto	11	33
02	04	Mupandawana	44	9

4.1.3 Polylines

A *Polyline* is a sequence of two or more connected points (vertices) with linear interpolation between points. Each consecutive pair of points defines a line segment. A polyline is stored as $v_0, v_1,\ldots,v_n$ where $n\geq 2$. The union of line segments $v_0v_1, v_1v_2,\ldots,v_{n-1}v_n$ forms the line.

A polyline objects represent roads, rivers, power lines, etc. A road for example is represented as a series of connected sections, each with two endpoints. Each endpoint is a

coordinate or location represented by their <x-coordinate, y-coordinate>. Figure 6 shows the representation of point a polyline object using a Entity Relationship Diagram.

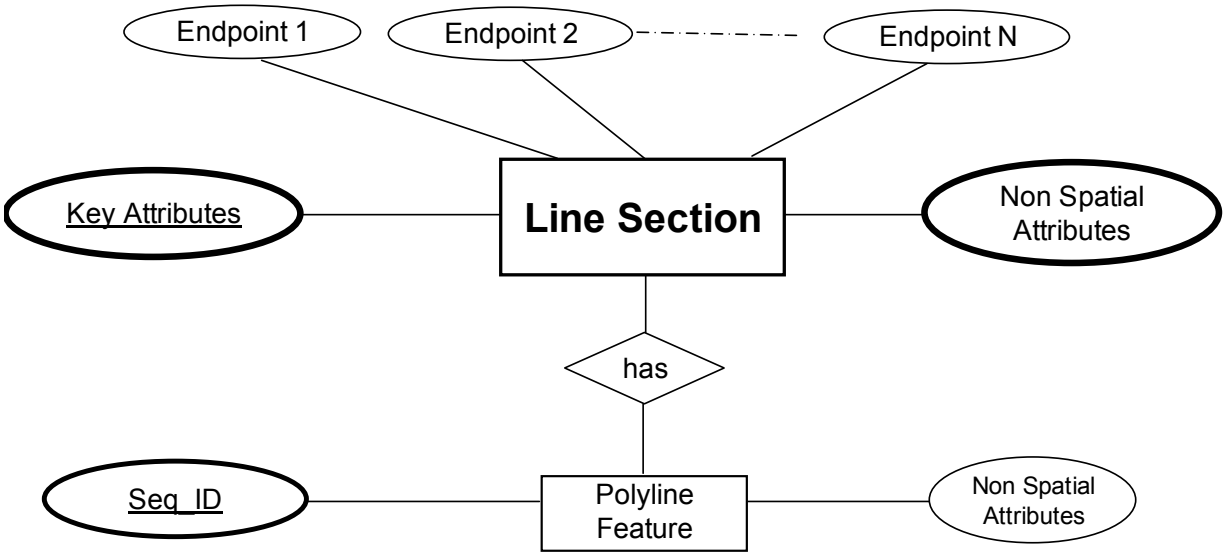

Figure 6: ERD representation of a Polyline object

The relation schema of the multipoint objects is as follows:

POLYLINE_FEATURE(Seq_ID, Non_Spatial_Attribute 1,…, Non_Spatial_Attribute N)

LINE_SECTION(Seq_ID, Key_Attribute(s), Non_Spatial_Attribute1, … , Non_Spatial_Attribute N, Endpoint1, …. Endpoint N). This can be normalized to get relations as given in the example below.

Tables 5 – 8 below shows the representation of polyline objects as relations. Each polyline is subdivided into sections, each with endpoints that captured as <x-coordinate, y-coordinate>.

Table 5: Representation of a Road Feature

Road_ID	Name	Year Completed
M1	Hre – Byo Road	1952
M2	Hre – Masv Road	1930

Table 6: Representation of a Road section

Road_ID	Section
M1	A
M1	B
M1	C
M1	D

Table 7: Representation of a Road section endpoints

Section	Endpoint
A	1
A	2
B	2
B	3
C	3
C	4
D	4
D	5

Table 8: Representation of road section endpoints as coordinates

Endpoint	X-Coord	Y-Coord
1	0	11
2	101	312
3	525	73
4	86	599
5	70	801

4.1.4 Polygons

A *Polygon* is a set of points $v_0, v_1,\ldots,v_n$ where $n\geq 3$ on a plane such that $v_0 = v_n$ .The union of line segments $v_0v_1, v_1v_2,\ldots,v_{n-1}v_n$ is an n-sided polygon whose vertices are $v_1, v_2,\ldots$ and whose edges are the line segments $v_0v_1, v_1v_2,\ldots.$ (Note that since $v_0 = v_n$, the polygon is automatically closed). The vertices of a polygon can be defined as $v_i = (x_i,y_i)$

Polygon objects represent land parcels, political boundaries, landuse areas, dams, disease zones, etc. Like Polylines, Polygons are also represented by a series of connected points but with the first and last points coinciding to form a closed loop. A district boundary for example is represented as a series of connected sections, each with two endpoints. Each endpoint is a coordinate or location represented by their <x-coordinate, y-coordinate>.

Figure 7 shows the representation of point a polygon object using a Entity Relationship Diagram.

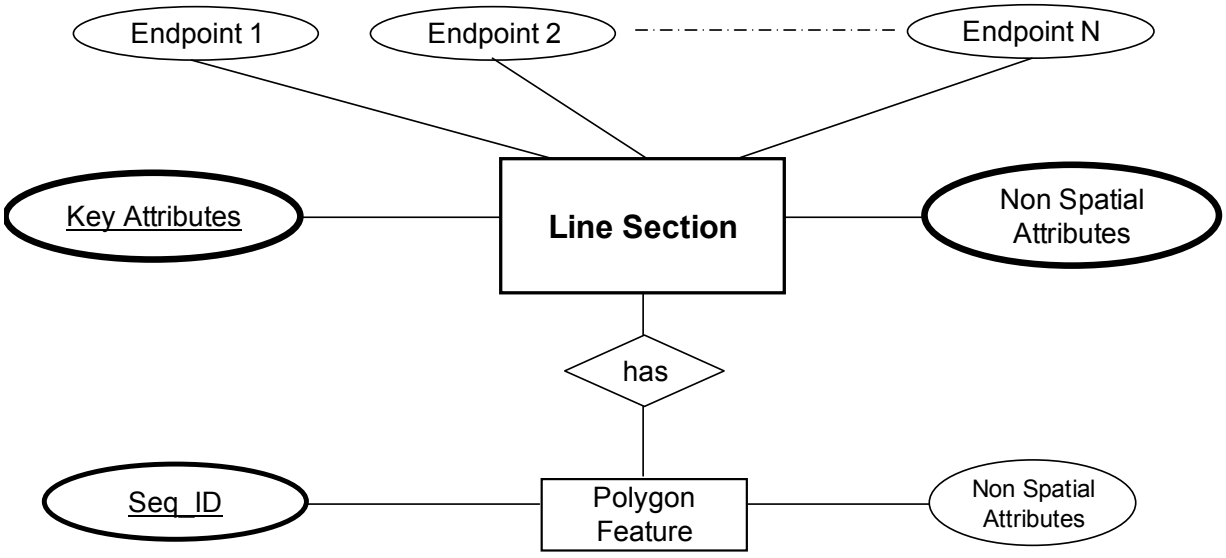

Figure 7: ERD representation of a Polygon object

Tables 9 – 12 below shows the representation of polygon objects as relations. Each polygon boundary is subdivided into sections, each with endpoints that captured as <x-coordinate, y-coordinate>. The first and last endpoint coincides to form a closed loop.

Table 9: District Relation

District_ID	Name	Province
D1	Tsholotsho	Mat South
D2	Gokwe	Midlands

Table 10: Representation of district boundary sections

District_ID	Section
D1	A
D1	B
D1	C
D1	D

Table 11: Boundary section endpoints

Section	Endpoint
A	1
A	2
B	2
B	3
C	3
C	4
D	4
D	5

Table 12: Representation of section endpoints

Endpoint	X-Coord	Y-Coord
1	0	1
2	1	3
3	5	3
4	6	5
5	7	8

4.2 Spatial Constraints

Spatial Constraints must be enforced to ensure spatial data integrity. Integrity refers to rules established to ensure relationships are valid, or that each attribute of an entity is entered properly and within the realm of its possibilities. The following constraints can be defined over any Polylines and Polygon:

- Polyline: $\{vi : 0 \leq i \leq n\}$ then $n \geq 2$
- Polygon: $\{vi : 0 \leq i \leq n\}$ then
    a. $n \geq 3$ (i.e. a polygon should have 3 or more edges)
    b. $v_0 = v_n$ (i.e. A Polygon should be closed)
    c. Two distinct edges intersect only if they have a common vertex, and they intersect only at a common vertex

4.3 Spatial Functions

Spatial Functions are used to analyse the functional relationships among geographic features and their attributes. This includes finding out how these features are connected and interact in real-life terms. This help people to better understand the world we live in. These functions are used to unravel complex factors affecting our natural environment. Sociologists use can use them to study patterns of human interaction. Epidemiologists use the functions to try to understand the spread of disease. *(Appendix A gives basic vector arithmetic principles that were used in the algorithms)*

Spatial functions accept one or more arguments and return a single value. An argument is a user supplied constant, variable or column reference. The format for a function is as follows:

Function_Name(argument1, argument2, …, argumentN)

Functions are used to:

- Perform calculations on data
- Modify individual data elements
- Manipulate output for groups of rows

4.3.1 Point to Point Distance

The straight line distance between two point objects (x_1, y_1) and (x_2, y_2) is:

$sqrt((x_2 - x_1)^2 + (y_2 - y_1)^2)$ according to Pythagoras Theorem

Point to Point distance defines the distance between Poling stations, between cities, etc.

4.3.1.1 Point to Point Distance Algorithm

//* this algorithm returns the distance between two points P and Q

Function Pt_Pt_Dist(P(x_1, y_1), Q(x_2, y_2))

begin

sx := $x_2 - x_1$; // difference along the x-Axis

sy := $y_2 - y_1$; // difference along the y-Axis

return sqrt($sx^2 + sy^2$);

end;

4.3.2 Point to Line Distance

Point to Line distance is a very useful function in GIS systems since it defines the distance of a point from a power line, river or road as shown on Figure 8 below.

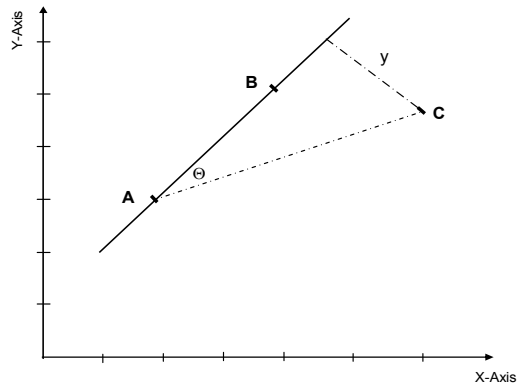

Figure 8: Point to Line Distance

Given 3 points A, B, and C, to find the distance from C to the line defined by A and B:

- Find two vectors from A to B (AB) and A to C (AC)
- AB x AC = |AB| |AC| Sin (Θ) but Sin (Θ) = y / |AC|
- Thus AB x AC = |AB| y
- Therefore, y = (AB x AC) / |AB|

To find the distance between point C and the line segment AB (the nearest point might be one of the endpoints):

- Perform Dot product AB**.**BC = |AB| |BC| Cos (Θ). If AB**.**BC > 0 then it means the angle between AB and BC is between -90 and 90, exclusive, therefore the nearest point to C on the segment AB is B
- Similarly if BA.AC >0 the nearest point is A
- If both Dot products are negative then the nearest point is along the line segment AB

4.3.2.1 Point to Line Algorithm

//* The Function below computes the Dot Product between two vectors AB and BC *//

Function Dot_Product(A(x_1, y_1), B(x_2, y_2), C(x_3, y_3))

begin

//* define vector AB and BC

AB := (x_2 - x_1, y_2 - y_1);

BC := (x_3 - x_2, y_3 - y_2);

//* Then return the Dot Product AB.BC which is:

//* n

//* $\boldsymbol{v.w} = \sum v_i w_i$

//* $i=1$

$\boldsymbol{v.w}$:= (x_2 - x_1)*(x_3 - x_2) + (y_2 - y_1)*(y_3 - y_2)

Return $\boldsymbol{v.w}$

end;

//* The Function below computes the Cross Product between two vectors AB and AC *//

Function Cross_Product(A(x_1, y_1), B(x_2, y_2), C(x_3, y_3))

begin

//* define vector AB and AC

AB := (x_2 - x_1, y_2 - y_1);

AC := (x_3 – x_1, y_3 – y_1);

//* Then return the Cross Product AB.AC

AB x AC := (x_2 - x_1)*(x_3 - x_1) - (y_2 - y_1)*(y_3 - y_1)

Return AB x AC

end;

//* The Function below computes the Distance from A to B *//

Function Distance(A(x_1, y_1), B(x_2, y_2))

begin

    //* define vector AB

    AB := (x_2 - x_1, y_2 - y_1);

    //* Then return the Distance from A to B

    Dist := SQRT((x_2 - x_1)^2 + (y_2 - y_1)^2)

    Return Dist

end;

//* The Function below then computes the distance from either line segment AB to a point C or from C to the line defined by segment AB *//

Function Polyline_Point_Distance (Polyline_Object_Name, C(x_3, y_3))

//* the function scans entire polyline; segment by segment and eventually finds the shortest distance from point C to the Polyline *//

Begin

  Least_Dist := 99999999999; //* set very large figure for the least distance of C to AB

  //* take each line segment AB with coordinates A(x_1, y_1) and B(x_2, y_2)

  For each line segment AB do

    Begin

        //* Find Distance from C to line defined by line segment AB. C can be either closer to points A or B, or it maybe closer to any point along the line AB *//

```
        Dist := cross_product (A(x1, y1), B(x2, y2), C(x3, y3)) / distance (A(x1, y1), B(x2,
        y2))
        Dot1 := Dot_Product (A(x1, y1), B(x2, y2), C(x3, y3));
        //* if Dot1>0 then C is closer to B
      If Dot1>0 then Dist := Distance (B(x2, y2), C(x3, y3));
        //* Check if C is closer to A
        Dot2 := Dot_Product (B(x2, y2), A(x1, y1), C(x3, y3));
      If Dot2>0 then Dist := Distance (A(x1, y1), C(x3, y3));
        //* otherwise find the distance to the line defined by line segment AB
      If  Least_Dist > ABS(dist) then Least_Dist := ABS(dist);
    End;
    Return ABS(Least_Dist);
end;
```

Analysis of Algorithm: This algorithm executes n times where n is the number of line segments in the polyline hence the algorithm is order O(n).

4.3.3 Length of a Polyline

The length of a polyline is found by adding the lengths of individual line segments, AB, BC, CD… using simple Pythagoras Theorem as shown on Figure 9 below

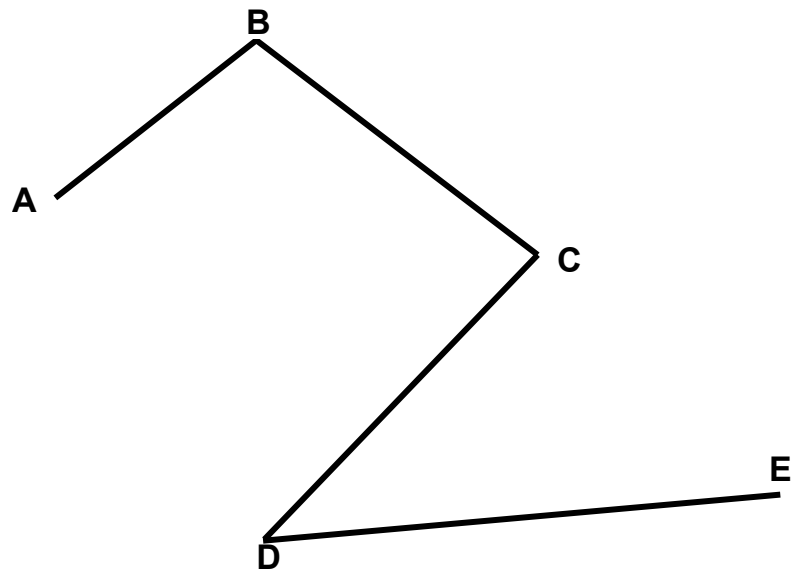

Figure 9 : Length of a Polyline

Let a Polyline Ψ be defined by its vertices $V_i = (x_i, y_i)$ for i =0, 1, …, n. For each edge V_iV_{i+1} of Ψ find the length of the edge $\xi_i = \text{Edge}(V_iV_{i+1})$. Then the total length of Ψ is equal to the sum of the lengths of all individual edges Δ_i for i =0, 1, …, n. That is:

$$L(\Psi) = \sum_{i=0}^{n-1} L(\xi_i)$$

Where $\xi_i = \text{Edge}(V_iV_{i+1})$

4.3.3.1 Length of Line Algorithm

```
//* the function receives N vertices sorted from left to right *//

Function length_of_line (P(i, j) in Array)
  Begin
    //* the line segments are read two vertices at a time
    temp_length := 0; Total_Length := 0;
    For i = 0 to N-1 do
      Begin
      //* get the other two coordinates from the array
          x1 := P(i,0);
          y1 := P(i,1);
          x2 := P(i+1,0);
          y2 := P(i+1,1);
      //* find the area of current triangle using the cross product
          Temp_length := SQRT((x2– x1)^2 + (y2 – y1)^2);
          Total_Length := Total_Length + Temp_length;
      End;
      Return Total_Length;
  End;
```

Analysis of Algorithm: This algorithm executes n times where n is the number of line segments in the polyline hence the algorithm is order O(n).

4.3.4 Perimeter of a Polygon

The perimeter of a polyline is found by adding the lengths of individual line segments, AB, BC, … using simple Pythagoras Theorem as shown on Figure 10 below. This is very similar to the length of a Polyline

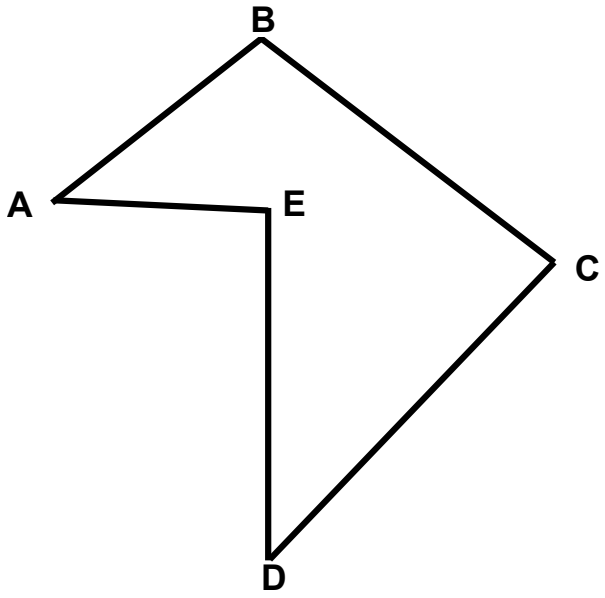

Figure 10: Perimeter of a Polygon

Let a Polygon Ψ be defined by is vertices $V_i = (x_i, y_i)$ for i =0, 1, …, n and $V_n = V_0$. For each edge V_iV_{i+1} of Ψ find the length of the edge $\xi_i = \text{Edge}(V_iV_{i+1})$. Then the perimeter of Ψ is equal to the sum of the lengths of all individual edges Δ_i for i =0, 1, …, n. That is:

$$L(\Psi) = \sum_{i=0}^{n-1} L(\xi_i)$$

Where $\xi_i = \text{Edge}(V_iV_{i+1})$

4.3.4.1 Perimeter of a Polygon Algorithm

```
//* the function receives N vertices sorted clockwise *//
//* This algorithm is exactly the same as the one given for the length of a polyline

Function length (P(i, j) in Array)
  Begin
    //* the line segments are read two vertices at a time
    temp_length := 0; Total_Length := 0;
    For i = 0 to N-1 do
      Begin
      //* get the other two coordinates from the array
          x1 := P(i,0);
          y1 := P(i,1);
          x2 := P(i+1,0);
          y2 := P(i+1,1);
      //* find the area of current triangle using the cross product
          Temp_length := SQRT((x2– x1)^2 + (y2 – y1)^2);
          Total_Length := Total_Length + Temp_length;
      End;
      Return Total_Length;
  End;
```

Analysis of Algorithm: This algorithm executes n times where n is the number of vertices that defines the polygon hence the algorithm is order O(n).

4.3.5 Area of Polygon

The Area function calculates the area of any Polygon given the coordinates around its perimeter as shown on Figure 11 below. The area of the polygon if found by Triangulating the polygon (e.i. dividing it into a number of triangles) and then calculating the area of the individual triangles as defined below.

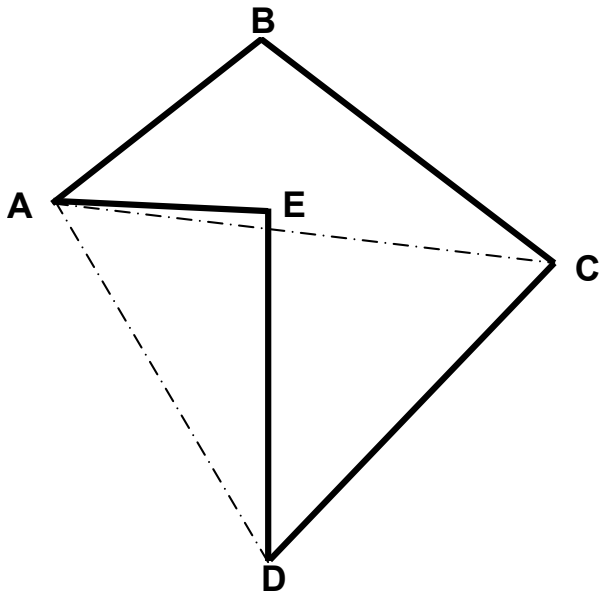

Figure 11 : Area of a Polygon

Let a Polygon **Ω** be defined by is vertices $V_i = (x_i, y_i)$ for i =0, 1, …, n with $V_n = V_0$. Let P be the first vertex on the polygon; and for each edge V_iV_{i+1} of **Ω** form the triangle $\Delta_i = \Delta PV_iV_{i+1}$. Then the area of **Ω** is equal to the sum of the signed areas of all triangles Δ_i for i =0, 1, …, n. That is:

$$A(\mathbf{\Omega}) = \sum_{i=0}^{n-1} A(\Delta_i)$$

Where $\Delta_i = \Delta PV_iV_{i+1}$

To compute the area of the above Polygon (Figure 11):

- Sort the vertices clockwise or counter clockwise
- The Polygon above is divided into triangles ABC, ACD and ADE. Some of the triangles might not be part of the polygon
- Take cross product of AB x AC to find the area of ABC; which will be negative because of the orientation of ABC
- Similarly, take cross product of ACD to get yet another negative number
- Lastly find the area of ADE which is positive
- Add the three areas and take the absolute number as the area of the Polygon

4.3.5.1 Area of Polygon Algorithm

```
//* the function receives N vertices sorted clockwise round the polygon *//

Function Area (P(i, j) in Array)
  Begin
    //* the polygon will be triangulated into triangles whose points are P(0), P(i) and P(i + 1)
    temp_area := 0;
    For i = 1 to N-1 do
      Begin
      //* First coordinate does not change
      //* get the other two coordinates from the array
          x1 := P(i,0) – P(0,0);
          y1 := P(i,1) – P(0,1);
          x2 := P(i+1,0) – P(0,0);
          y2 := P(i+1,1) – P(0,1);
      //* find the area of current triangle using the cross product
          Cross_prod := x1*y2 – x2*y1;
          Temp_area := temp_area + cross_prod;
      End;
      Return temp_area;
  End;
```

Analysis of Algorithm: This algorithm executes n - 2 times where n is the number of vertices that forms the polygon hence the algorithm is order O(n-2).

4.3.6 Intersection of Two lines

Finding the point at which two lines intersect is a geometric problem that is very useful is GeoSpatial databases. This is very important to find the point where a road crosses a river, point where electricity lines cross municipal boundaries, or a point where rivers meet.

Given two line segments with endpoints A, B and C,D. The line segments if extended to infinity will either meet on some point or they are parallel. Figure 12 below shows the different scenario of the two segments:

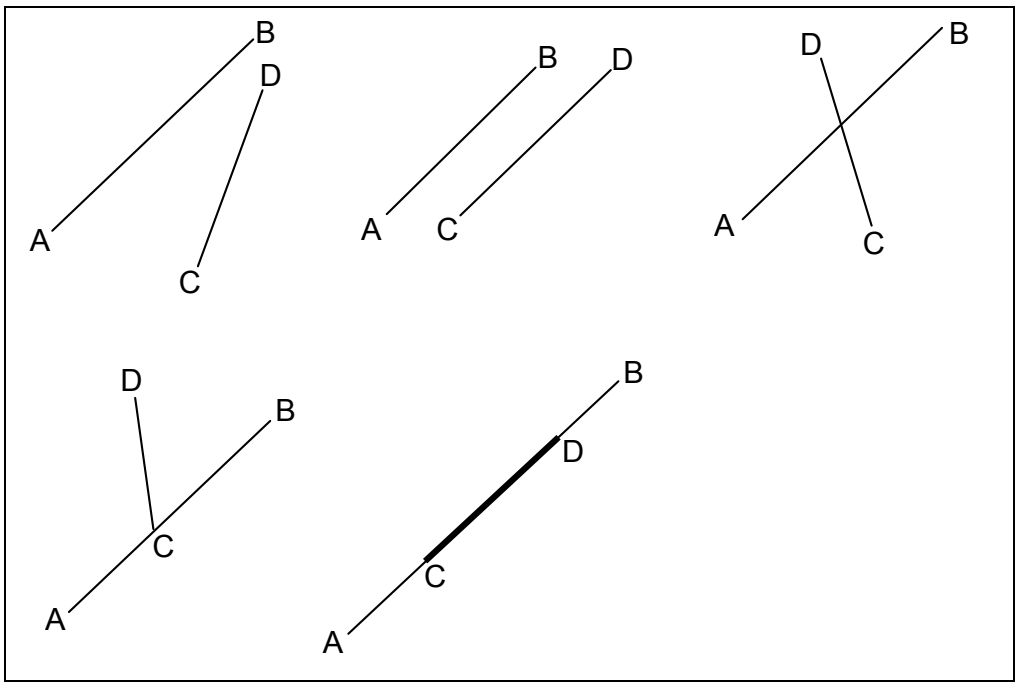

Figure 12: Scenarios of two line segments

We can write the parametric representation of the lines as:

$AB(t) = A + \mathbf{b}t$ ..... (1)

AB representing line segment from A to B and $\mathbf{b} = A - B$. Parameter t varies from 0 to 1. If t is allowed to vary from $-\infty$ to ∞ then the line will extend to form the parent line.

Similarly we can represent the line segment CD as:

$CD(u) = C + \mathbf{d}u$ ..... (2)

Where $d = D - C$

When the two line segments to meet then:

$A + \mathbf{b}t = C + \mathbf{d}u$ ..... (3)

We can write for convenience $\mathbf{e} = C - A$

$\mathbf{b}t = \mathbf{d}u - \mathbf{e}$ ..... (4)

There are two cases to consider: the term $\mathbf{d}^{\perp}\mathbf{b} = 0$ and $\mathbf{d}^{\perp}\mathbf{b} \neq 0$:

<u>CASE 1: The term $\mathbf{d}^{\perp}.\mathbf{b} \neq 0$</u>

We can Dot both sides with $\mathbf{d}^{\perp}$ to get:

$t = \mathbf{d}^{\perp}.\mathbf{e} \,/\, \mathbf{d}^{\perp}.\mathbf{b}$ ..... (5)

We can Dot both sides with $\mathbf{b}^{\perp}$ to get:

$u = \mathbf{b}^{\perp}.\mathbf{e} \,/\, \mathbf{b}^{\perp}.\mathbf{d}$ ..... (6)

The case where $\mathbf{d}^{\perp}.\mathbf{b} \neq 0$ means the parent lines do intersect but does not say whether the two line segments AB and CD intersect themselves. If t is outside the range (0, 1) then the line segment AB does not reach the line segment CD and similarly if u is outside the range (0, 1) then the line segment CD does not reach AB. However if both t and u are between the range (0, 1) then the two line segment intersect at some point I:

$I = A + (\mathbf{d}^{\perp}.\mathbf{e} / \mathbf{d}^{\perp}.\mathbf{b})\mathbf{b}$ ..... (7)

Found by substituting t in (1) by (5)

Therefore, given two line segments AB and CD with endpoints $A = (a_x, a_y)$, $B = (b_x, b_y)$ and $C=(c_x,c_y)$, $D = (d_x, d_y)$ then from the above:

- $\mathbf{b} = B - A$

  $\mathbf{b} = (b_x, b_y) - (a_x, a_y)$; thus

  $\mathbf{b} = (b_x - a_x, b_y - a_y)$

- $\mathbf{d} = D - C$

  $\mathbf{d} = (d_x, d_y) - (c_x, c_y)$; thus

  $\mathbf{d} = (d_x - c_x, d_y - c_y)$

- $\mathbf{d}^{\perp} = (-(d_y - c_y), d_x - c_x)$

- $\mathbf{b}^{\perp} = (-(b_y - a_y), b_x - a_x)$

- $\mathbf{e} = C - A$

  $\mathbf{e} = (c_x, c_y) - (a_x, a_y)$; thus

  $\mathbf{e} = (c_x - a_x, c_y - a_y)$

- $\mathbf{d}^{\perp}.\mathbf{d} = (-(d_y - c_y), d_x - c_x).(d_x - c_x, d_y - c_y)$

  $\mathbf{d}^{\perp}.\mathbf{d} = -(d_y - c_y)* (d_x - c_x) + (d_x - c_x)*(d_y - c_y)$

- $t = \mathbf{d}^{\perp}.\mathbf{e} / \mathbf{d}^{\perp}.\mathbf{b}$

  $t = ((-(d_y - c_y), d_x - c_x).(c_x - a_x, c_y - a_y)) / (-(d_y - c_y)* (d_x - c_x) + (d_x - c_x)*(d_y - c_y))$

  $t = ((-(d_y - c_y)* (c_x - a_x)) + (d_x - c_x)*(c_y - a_y)) / (-(d_y - c_y)* (d_x - c_x) + (d_x - c_x)*(d_y - c_y))$

- $u = \mathbf{b}^{\perp}.\mathbf{e} / \mathbf{b}^{\perp}.\mathbf{d}$

  $u = ((-(b_y - a_y), b_x - a_x).(c_x - a_x, c_y - a_y)) / (-(b_y - a_y), b_x - a_x).(d_x - c_x, d_y - c_y)$

  $u = ((-(b_y - a_y)* (c_x - a_x)) + (b_x - a_x)*(c_y - a_y)) / (-(b_y - a_y)* (d_x - c_x) + (b_x - a_x)*(d_y - c_y))$

4.3.6.1 Line Intersection Algorithm

//* the algorithm receives two Polylines as arguments and then takes each line segment AB of Polyline1 and checks if it intersects with any segment CD of Polyline2*//

```
Function Line_intersection (Polyline1, Polyline2)
  Begin
  For each line segment AB of Polyline1 Do
  //* check if AB intersect with any line segment CD of Polyline2
    For each line segment CD of Polyline2 do
      Begin
      //* Take the four coordinates as A(ax, ay), B(bx, by), C(cx,cy), D(dx, dy) )
      //* First check if d⊥.b = 0. If yes then the lines are parallel or they are superimposed
          Case1 := -(dy - cy)* (dx - cx) + (dx - cx)*(dy - cy);
```

If Case1 = 0 then

//* the lines segments are either parallel of superposed

Line segments don't intersect

Else

Begin

//* Check if both t = $\mathbf{d}^{\perp}\mathbf{.e}$ / $\mathbf{d}^{\perp}\mathbf{.b}$ and u = $\mathbf{b}^{\perp}\mathbf{.e}$ / $\mathbf{b}^{\perp}\mathbf{.d}$ are both within the range (0, 1) meaning that the two line segments indeed intersect, otherwise they don't. In the latter case their parent lines intersect if extended to infinity *//

$t = ((-(d_y-c_y)* (c_x-a_x)) + (d_x-c_x)*(c_y - a_y)) / (-(d_y - c_y)* (d_x - c_x) + (d_x - c_x)*(d_y - c_y))$

$u = ((-(b_y-a_y)*(c_x-a_x)) + (b_x-a_x)*(c_y - a_y)) / (-(b_y - a_y)* (d_x - c_x) + (b_x - a_x)*(d_y - c_y))$

If NOT (t is between 0 and 1 and u is between 0 and 1) then

Although not parallel the two line segments don't intersect

Else

Begin

//* Now that the lines intersect we can calculate the intersection point

I = A + ($\mathbf{d}^{\perp}\mathbf{.e}$ / $\mathbf{d}^{\perp}\mathbf{.b}$)**b** this is basically

I = A + t**b**

$I := (a_x, a_y) + t (b_x, b_y)$

Therefore the x-coordinate $I_x = a_x + tb_x$ and the x-coordinate $I_y = a_y + tb_y$ *//

$I_x = a_x + tb_x$

$I_y = a_y + tb_y$

Return (I_x, I_y);

End; //* else

End; //* else

End; //* For

End;

Analysis of Algorithm: This algorithm executes n * m times where n is the number of line segments in the first polyline and m is the number of line segments in the second polyline hence the algorithm is of the order O(nm) in the worst case.

4.3.7 Nearest Neighbours – using Voronoi Diagrams

The nearest neighbours function is one of the most important functions is GeoSpatial databases as it defines the closest points satisfying a particular criteria. For example it helps in identifying the closest neighbour with a combine harvester or the closest hospital with particular equipment etc. The nearest neighbour function is also important in citing businesses as far as possible from competitors.

A Voronoi Diagram, also called Dirichlet Tessellation, records information about what is close to what as shown on Figure 13 below. Let $P = \{P_1, P_2, \ldots, P_n\}$ be a set of points, called Sites, on a place. Two sites are said to be neighbours if they share a common edge. A Voronoi Cell $V(P_i)$ for point P_i is a set of q points nearer to P_i than any other sites.

A Voronoi cell for point P_i is defined as:

$V(P_i) = \{q \mid dist(P_i, q) < dist(P_j, q), \text{ for } j \mathrel{!}= i\}$

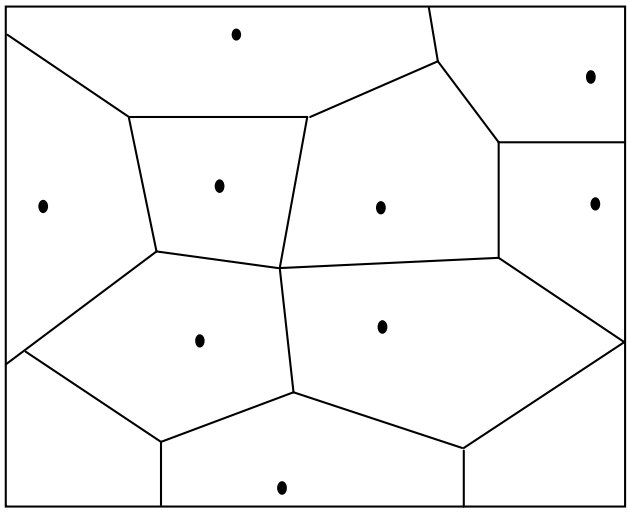

Figure 13: Voronoi Diagram

The following are properties of the Voronoi diagram:

- **Voronoi Edges**: Each edge is equidistant from its nearest neighbours P_i and P_j.
- **Voronoi Vertices**: Three Voronoi cells $V(P_i)$, $V(P_j)$ and $V(P_k)$ 0

4.3.7.1 Nearest Neighbours Algorithm

```
Function Nearest_Neighbours (Point p, Condition)
  Begin
    //*  Pick all sites satisfying the given Condition
    For all sites q ∈ {P1, P2, …, Pn} where Condition is true do
      Begin
      //* calculate the distance between p and q; dist(p, q)
      //* check if q is close to q than any other site r
      For all r ∈ {P1, P2, …, Pn} and r != q
            If dist(p, q) < dist(q, r) then
             Begin
               Draw perpendicular bisector between p and q
               Form vertices at each line intersection
            End;
      End;
      //* the nearest neighbours are a set of sites whose voronoi cells share a common
      boundary with cell V(p)
      *//
      Return list of sites whose cells share boundary with V(p);
  End;
```

Analysis of Algorithm: This algorithm executes n * (n – 2) times where n is the number of sites meeting the given condition hence the algorithm is of the order O(n(n – 2)) in the worst case.

4.3.8 Point within Polygon

The function is used to determine whether a point is within a triangle. This function can be used to determine the owner of a borehole or any other sites or the district in which an object is located.

To determine if a point is within a polygon or not is a matter of casting a "ray" from the point to infinity or any line known to be outside the polygon. The algorithm counts the number of times the ray crosses the polygon edges, as shown on Figure 12. If the number happens to be odd then the point will be in the polygon (e.g. points e, f, h and j in Figure 13). If the number is even then the point will be outside the polygon (e.g. points a, b, c, d, g and i on Figure 13). If the ray crosses the polygon at a vertex or an edge (e.g. points b, f, g and j on Figure 13) then the cases are counted as zero or twice

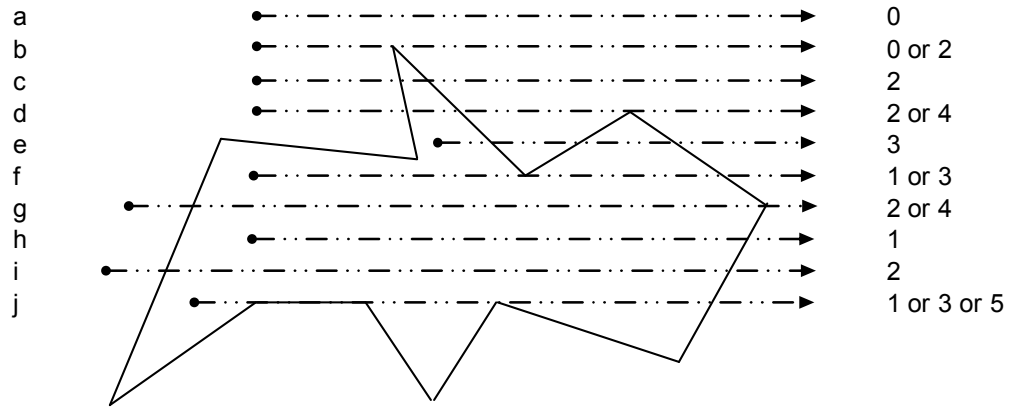

Figure 14: Determining if point is in polygon

4.3.8.1 Point within Polygon Algorithm

```
//* the algorithm accepts a point P(x, y) and checks if it within polygon P with n vertices
//* v0, v1,…,vn where n≥3 and v0 = vn . The vertices of a polygon are defined as vi = (xi,yi)
//* the algorithm returns the value True if the point is in the polygon q(xq, yq)otherwise it
//* returns the value False
//* The polygon vertices are placed in a 2 x 2 Array in a clockwise or anti-clockwise
direction
Function Point_within_polygon (P(i, j) in Array)
  Begin
    //* the line segments are read two vertices at a time
    counter := 0;
    For i = 0 to N-1 do
      Begin
      //* get the other two coordinates from the array
          x1 := P(i,0);
          y1 := P(i,1);
          x2 := P(i+1,0);
          y2 := P(i+1,1);
      //* check if ray crosses the edge
          If ray crosses edge vivi+1 then counter = counter +1;
          Elsif ray crosses polygon at vertex vi then counter = counter +1;
      //* note that the vertex will be counted again on the next edge
          Elseif ray coincides with an edge then counter = counter +1;
      End;
```

```
    //* check if counter is even or odd
    If counter MOD 2 = 0 then
        Return TRUE; //* point is outside the polygon
    Else
        Return FALSE //* point is inside the polygon
    End if;
End;
```

Analysis of Algorithm: This algorithm executes (n – 1) times where n is the number of vertices or the number of edges of the polygon hence the algorithm is of the order O(n – 2) in the worst case for each polygon searched

4.4 Spatial Relational Algebra

This section considers operations for manipulating sets of database objects with spatial attributes from an algebraic point of view. These operations can be classified as *spatial selection*, spatial projection, *spatial join*, and *other set operations*.

4.4.1 Spatial Selection

A selection is an operation that returns from a relation tuples satisfying a given predicate. The Spatial select operation is the same as the relational select except that in this instance it describes a selection based on a spatial predicate. Spatial select is unary operation since it operates on a single relation. Given a Relational R_1 we write the Spatial Select Operator as:

$\delta_{predicate}(R_1)$

Where the *Predicate* is a regular expression involving Relation Attributes and/or constants with comparators and logical operators

Examples:

$\delta_{area\ () > 2000}$(Districts); Selects all districts with Area exceeding 200 Km^2

$\delta_{x > 30 \wedge y < 50}$ (City); Selects all Cities found to the right of 30^o longitude and below 50^o latitude

$\delta_{distance\ (Centre,\ Kadoma) < 300}$(City); Selects all cities and towns within 300 of Kadoma

The cardinality of the Spatial Select Operation can be expressed as:

$$0 \leq | \delta_{predicate}(R_1)| \leq |R1|$$

In set notation given the relation instance r(R1) the we can express:

$$\delta_{predicate}(R_1) = \{t_i \mid t_i \in r(R1)\}$$

4.4.2 Spatial Project

The Spatial Project operation is a unary operation that returns its argument relation, with certain attributes eliminated. Given the relation R1 and its schema S(R1) we write the project as:

$$\Pi_{<A1, A2, \ldots, An>}(R_1)$$

Where $\{A1, A2, \ldots, An\} \subseteq S(R1)$

The Project operation eliminates any duplicates in the resultant relation.

Examples:

Given the Relation City as shown on Table 13 below:

Table 13: City Relation

City_ID	City_Name	Population (m)	Area (Ha)	X-Coord	Y-Coord
C01	Harare	2	800000	30	23
C02	Bulawayo	1	900000	10	11
C03	Mutare	0.8	280000	20	25
C04	Gweru	0.6	300000	80	90

We can define the project operation to be the Relation as shown on Table 14 below

$$\Pi_{<City\_Name,\ Area>}(City)$$

Table 14: Resultant relation after a project operation on City object

City_Name	Area (Ha)
Harare	800000
Bulawayo	900000
Mutare	280000
Gweru	300000

We can combine Project and select operations to get the relation as shown on Table 16:

$$\Pi_{<City\_Name,\ Area>}(\delta_{Area\ <\ 500000}(City))$$

Table 15: Resultant relation after a select and project operation

City_Name	Area (Ha)
Mutare	280000
Gweru	300000

The cardinality of the Spatial Project Operation can be expressed as:

$$0 \leq | \prod_{<A1, A2, \ldots, An>} (R_1)| \leq |R1|$$

Given R2 = $\prod_{<A1, A2, \ldots, An>} (R_1)$ the relation instances r(R1) and r(R2) can be expressed in set notation as:

$$\prod_{<A1, A2, \ldots, An>} (R_1) = \{t_i \mid t_j \in r(R1) \wedge t_i = t_j\ (r(R2))\}$$

4.4.3 Spatial Cartesian Product

The Spatial Cartesian Product is a binary operation that returns all possible tuple combinations from two relations. Given two relations R1 and R2 the Cartesian Product is denoted by R1 x R2. Given two relations as shown on Table 16 and 18:

Table 16: Roads Relation

Road_ID	**Name**	**Year Completed**
M1	Hre – Byo Road	1952
M2	Hre – Masv Road	1930

Table 17: Road Section relation

Road_ID	Section
M1	A
M1	B
M1	C
M1	D

We can define **Roads** x **Road_Section** as the relation given on Table 18 below that joins every tuple from the Roads and joins it to every tuple from Road_Section:

Table 18: Cartesian product of the Roads and Road_Section relations

Road_ID	Name	Year Completed	Road_ID	Section
M1	Hre – Byo Road	1952	M1	A
M1	Hre – Byo Road	1952	M1	B
M1	Hre – Byo Road	1952	M1	C
M1	Hre – Byo Road	1952	M1	D
M2	Hre – Masv Road	1930	M1	A
M2	Hre – Masv Road	1930	M1	B
M2	Hre – Masv Road	1930	M1	C
M2	Hre – Masv Road	1930	M1	D

4.4.4 Rename

The Spatial Rename is a unary operator used to change the name of a Relation. Given a relation R1 the Spatial Rename operator is denoted as:

$$\rho_x (R1)$$

Where Relation R1 is returned under a new name x

Example

$$\rho_{Water\_Sources} (Dams)$$

The operator changes the name of the Dams relation to Water_Sources

4.4.5 Union Operation

The Union Operation is a binary operation that combines all tuples from two or more relations. Given two relations R1 and R2 the Union of the two relations denoted $R1 \cup R2$. However, the Union operation requires that $R1(a_1, a_2, \ldots, a_n)$ and $R2(b_1, b_2, \ldots, b_m)$ be Union Compatible and this means that:

i. degree(R1) = degree(R2)
ii. n = m
iii. $dom(R1.a_i) = dom(R2.b_i)$
iv. The relation schemas should also be equal: S(R1) = S(R2)
v. The cardinality of the Union can be expressed as:
$\max(|R1|, |R2|) \leq |R1 \cup R2| \leq |R1| + |R2|$

In set notation given two relation instances r(R1) and r(R2) the we can express:

$$R1 \cup R2 = \{t_i \mid t_i \in r(R1) \vee t_i \in r(R2)\}$$

4.4.6 Intersection Operation

The Intersection Operation is a binary operation that returns tuples found in both two relations. Given two relations R1 and R2 the Intersection of the two relations denoted R1 $\cap$ R2. However, the Intersection operation requires that the two relations R1(a_1, a_2, …, a_n) and R2(b_1, b_2, …, b_m) be Union Compatible and this means that:

i. degree(R1) = degree(R2)
ii. n = m
iii. dom(R1.a_i) = dom(R2.b_i)
iv. The relation schemas should also be equal: S(R1) = S(R2)
v. The cardinality of the Intersection can be expressed as:
   $0 \leq |R1 \cap R2| \leq \min(|R1|, |R2|)$

In set notation given two relation instances r(R1) and r(R2) the we can express:

$$R1 \cap R2 = \{t_i \mid t_i \in r(R1) \wedge t_i \in r(R2)\}$$

4.4.7 Set Difference Operation

The Set Difference Operation is a binary operation that returns tuples found in one relation and not in the other for example Customers with a bank account but have no loan. Given two relations R1 and R2 the Set difference of the two relations is denoted by R1 - R2. However, the set difference operation requires that the two relations R1(a_1, a_2, …, a_n) and R2(b_1, b_2, …, b_m) be Union Compatible and this means that:

i. degree(R1) = degree(R2)
ii. n = m
iii. dom(R1.a_i) = dom(R2.b_i)
iv. The relation schemas should also be equal: S(R1) = S(R2)
v. The cardinality of the Set Difference can be expressed as:
   $0 \leq |R1 - R2| \leq |R1|$

In set notation given two relation instances r(R1) and r(R2) the we can express:

$$R1 - R2 = \{t_i \mid t_i \in r(R1) \wedge t_i \notin r(R2)\}$$

4.4.8 Spatial Join Operation

A spatial join is a binary operation that combines relations from two relations by comparing any two tuples through a predicate on their attribute values. The join operator forms a Cartesian product of two or more relations before selecting tuples with the same attribute values on some common attributes.

Given two relations R1(a_1, a_2, …, a_n) and R2(b_1, b_2, …, b_m) the Spatial Join of the two relations is denoted by R1 $\bowtie_\theta$ R2. Where θ is a predicate in the form of R1.a_i = R2.b_j (called an Equi-Join); although other comparators can be used for example <, >, ≤, ≥, ≠, etc.

Properties of a Spatial Join Operation:

i. degree(R1 $\bowtie_\theta$ R2) = n + m
ii. The relation schemas S(R1 $\bowtie_\theta$ R2) = { a_1, a_2, …, a_n} ∪ { b_1, b_2, …, b_m }
iii. The cardinality of the Spatial Join can be expressed as:

$$0 \leq | R1 \bowtie_\theta R2| \leq |R1| * |R2|$$

In set notation given two relation schemas S(R1) and S(R2) the we can express:

$$R1 \bowtie_\theta R2 = \{x \mid x (S(R1)) \in S(R1) \wedge x (S(R2)) \in S(R2) \}$$

Example:

Table 19: District Relation

District_ID	Name	Population (m)
01	Gokwe	1.1
02	Gutu	1.0

Table 20: Health Centres Relation

District_ID	Centre_ID	Centre Name	X-Coord	Y-Coord
01	01	Cheziya	11	2
01	02	Nembudziya	22	50
02	01	Serima	3	14
02	02	Charthworths	8	1
02	03	Zimuto	11	33
02	04	Mupandawana	44	9

The Spatial Join below gives a resultant relation as shown on Table 22 below.

$$\text{District} \bowtie_{(\text{district.District\_ID} = \text{Health\_Centres.District\_ID})} \text{Health\_Centres}$$

Table 21: Result of an Equi-Join between District and Health_Centres Relations

Health_Centres. District_ID	Health_Centres. Centre_ID	Health_Centres. Centre Name	Health_Centres. X-Coord	Health_Centres. Y-Coord	District. District_ID	District. Name	District. Population(m)
01	01	Cheziya	11	2	01	Gokwe	1.1
01	02	Nembudziya	22	50	01	Gokwe	1.1
02	01	Serima	3	14	02	Gutu	1.0
02	02	Charthworths	8	1	02	Gutu	1.0
02	03	Zimuto	11	33	02	Gutu	1.0
02	04	Mupandawana	44	9	02	Gutu	1.0

4.5 Spatial Queries

Traditional Structured Query Language (SQL) is inadequate to express spatial queries, therefore, there is need to extend its capability with spatial constructs.

4.5.1 Selection Statement

The basic structure of a SQL statement consists of three clauses: **Select**, **From**, and **Where**.

- **Select**: corresponds to the Projection operator in relational algebra as it lists the desired attributes in the query results
- **From**: Corresponds to the Cartesian Product operation in relational algebra. It lists the relations to participate in the Cartesian product
- **Where**: Corresponds to the selection predicate in relational algebra

Given a set of relations $\mathbf{r_1, r_2, \ldots, r_n}$ and a ser of attributes $\mathbf{A_1, A_2, \ldots, A_n}$

where $\mathbf{A_1, A_2, \ldots, A_n} \in S(\mathbf{r_1}), S(\mathbf{r_2}), \ldots, S(\mathbf{r_n})$; with $S(\mathbf{r_i})$ being the schema of relation $\mathbf{r_i}$ and **P** being a predicate

Then SELECT clause thus appears in the form:

SELECT $\mathbf{A_1, A_2, \ldots, A_n}$

FROM $\mathbf{r_1, r_2, \ldots, r_n}$

WHERE **P**

The above statement is equivalent to:

$$\mathbf{\Pi A_1, A_2, \ldots, A_n} (\delta_P (\mathbf{r_1 \times r_2 \times \ldots \times r_n}))$$

The predicate **P** clause may compare values in columns, literal values, arithmetic expressions of functions. The WHERE clause expects 3 elements:

- A column name
- A comparison operator
- A column name, constant or list of values

4.5.2 Spatial Functions

This section explores various examples of the use of spatial functions in Spatial Query Language:

4.5.2.1 Point to Point Distance

The following functions demonstrate the use of the spatial point to point distance function:

i. Find all cities with a population of above half a million and are within 200 Kms from Chivhu

```
SELECT   City_name, District, Population
FROM     Cities
WHERE    Distance(centre-x, centre-y, Chivhu) < 200 and population > 500000;
```

ii. Find the distance of all cities with a population of above half a million and from Harare

```
SELECT     City_name, District, distance(centre-x, centre-y, Harare)
FROM       Cities
WHERE      population > 500000;
```

4.5.2.2 Point to Line Distance

Find all Rural Service Centres that are that are within 2km from the M5 Road

```
SELECT    Centre_name, District
FROM      Rural_Centres
WHERE     distance(centre-x, centre-y,  M5_Road) < 2km;
```

4.5.2.3 Length of Polyline

Find the length of all tarred roads

```
SELECT    Road_name, length(Road_Name)
FROM      Roads
WHERE     Surface_Type='Tarred' ;
```

4.5.2.4 Perimeter of Polygon

Find the List of dams in Mash Central that have a perimeter of above 2km

```
SELECT    Dam_Name
FROM      Water_Bodies
WHERE     Perimeter(Dam_Name)>2km and District = 'Mash Central';
```

4.5.2.5 Area of Polygon

Find the List of farms with extent exceeding 2000Ha

```
SELECT    Farm_Name, District, Farm_Owner
FROM      Farms_Inventory
WHERE     Perimeter(Dam_Name)>2km and District = 'Mash Central';
```

4.5.2.6 Intersection of Polylines

Find the point where Odzi River meets Save River

```
SELECT    Intersection(Save, Odzi)
FROM      Rivers;
```

4.5.2.7 Nearest Neighbours

Find the List of dams in Mash Central that have a perimeter of above 2km

```
SELECT    Dam_Name
FROM      Water_Bodies
WHERE     Perimeter(Dam_Name)>2km and District = 'Mash Central';
```

4.5.2.8 Point Within Polygon

List all growth points in Midlands Province

```
SELECT    *
FROM      Service_Centres
WHERE     Centre_Type = 'Growth_Point' and within(Province,'Midlands');
```

4.5.3 Set Operators

4.5.3.1 Union

List all cities in Mash Central as well as all towns with a population between 20000 and 50000

```
SELECT    Centre_Name
FROM      Cities
WHERE     within(Province,'Mash Central')
UNION
SELECT    Centre_Name
FROM      Towns
WHERE     population between 20000 and 50000;
```

4.5.3.2 Intersection

List rural service centres that are within 20Km from the Zesa Z12 grid and are within 40Km from the Zambezi

```
SELECT    Centre_Name
FROM      Rural_Centres
WHERE     Line_Pt_Distance(Z12, Centre_Name) ≤ 12
INTERSECT
SELECT    Centre_Name
FROM      Rural_Centres
WHERE     Line_Pt_Distance(Zambezi, Centre_Name) ≤ 40;
```

4.5.3.3 Difference

List all farms whose extents is greater than 5000Ha excluding those with bilateral protection agreements

```
SELECT    Farm_Name, District, Farm_Size
FROM      Farms_Register
WHERE     within(Province,'Mash Central')
MINUS
SELECT    Farm_Name, District, Farm_Size
FROM      Farms_Register
WHERE     Ownership='Bilateral';
```

5.0 Discussions

This section analyses results of the research and highlights shortfalls of the algorithms presented.

5.1 Analysis

Framework: The framework presented in this research proves that geometric objects (Points, Multipoint, Polylines or Polygons) can be successfully represented using the relational model while at the same time maintaining all the properties of a good design:- that is relations that are normalised, indexed and all the data integrity constraints preserved.

Spatial Functions Algorithms: The spatial functions presented in this thesis depict the functional relationship amongst various spatial components in real life. They can be used to calculate the area under winter cropping, or they can determine the land parcel on which illegal activities such as gold panning are taking place, or finding nearest neighbours with satisfying a set criteria, etc.

Relational Algebra Extensions: This research also proved that we can expand the existing relational algebra in order for us to meet current and future demands for valuable information from GeoSpatial Databases

5.2 Limitations

Error Handling: The algorithms presented in this research do not attempt to rectify location errors that may arise as a result of distortions of aerial photographs or satellite images or as a result of the digitization process. Location errors can also result from improper conversion from one projection to another.

Map Scale: The algorithms do not accommodate the use of different map scales. Map users should generally be allowed to use scales of their choice. The computed distances should be multiplied by a simple factor to cater for the scale factor

Visual Queries: Implementations of the algorithms should allow for visual queries to be executed. To cater for the different scales the algorithms should multiply all distance by a scale factor.

Query Windows: The algorithm do not allow users to select an area of interest on a map to allow users to perform an analysis on the chosen area

6.0 Conclusions and Recommendations

6.1 Summary

The main intent of this research was to develop a relational database framework that accommodates geospatial data sets. Based on this framework the research further developed a number of very important algorithms for spatial data analysis as well as assessing the performance of those algorithms

On the methodology used in this research:- Entity Relationship Diagrams were used to represent the framework from which relations were derived. Standard development methodologies were used in the development of the various spatial analysis algorithms.

The scope of the framework covered the following points, multipoints, polylines and polygons. Spatial analysis algorithms covered included, point to point distance, point to line distance, area of a polygon, perimeter of a polygon, length of a polyline, point within polygon

The research proposed an extension of the existing relational algebra to accommodate spatial functions. The existing SQL was also extended..

6.2 Conclusions

The fact that Geographic Information System technology is changing the way people live is a reality. The condition of the Earth's surface, atmosphere, and subsurface can be examined by feeding satellite data into a GIS. Researchers have now the ability to examine the variations in Earth processes over days, months, and years. As an example, the changes in vegetation cover through a growing season can be animated to determine when drought was most extensive in a particular region. There are many other very advanced inventions that are still needed in the area.

Firstly this research presented a relational database framework for geospatial data sets with which system developers can use to develop geospatial databases. The advantages of this framework are that:

- Elimination of redundancy and duplication
- Applications become data independent so that multiple applications can use the same data and they can evolve separately over time
- Data sharing is facilitated and a corporate view of data can be provided to all managers and users. This makes it easy to integrate geospatial data with other related external systems
- Security and standards for data and data access can be established and enforced.
- Geospatial databases can then inherit many features from relational database management systems

Secondly the thesis presented a number of algorithms to enable analysis to be done on various geometrical objects on a map. Spatial analysis can be used to better understand the world we live in. Researchers can use these functions to unravel complex factors affecting

the natural environment. Sociologists can also use geospatial technology to study patterns of human interaction. Epidemiologists use the same functions to try to understand the spread of diseases. Geospatial functions can be used on many other areas as well.

Lastly this thesis proved that the algorithms used for spatial analysis can indeed work in optimal time hence they can be deployed in queries to add the needed functionality to SQL.

6.3 Recommendations

This thesis proposed a framework on which geospatial data sets can be represented and subsequently a number of algorithms for spatial analysis were also presented. However, in order for a complete geospatial database system to be developed the following future work is recommended:

i. **Image Errors**: Further research needs to be done on how to cater for errors arising as a result of distortion of images
ii. **Map Scales**: The algorithms should provide for different map scales to be used
iii. **Query Windows**: The algorithms should be extended to allow users to query a selected area of a map
iv. **Visual Queries**: There is need for the development of additional algorithms to allow for visual queries and the display of results on a map
v. **Terrain**: there is need to also extend the algorithms to 3-D to handle terrain.

References

1. **Jeab Ihm and Xavier Lopez**. August 2005 Oracle Spatial 10g. An Oracle White Paper. (http://www.oracle.com/technology/products/spatial)

2. **Oracle Corporation**. August 2005 Oracle Spatial (Relational) Reader/Writer. (http://www.oracle.com/technology/products/spatial)

3. **Oracle Corporation.** 2005 Oracle Database 10*g*. Oracle Spatial Network Data: An Oracle Technical White Paper. (http://www.oracle.com/technology/products/spatial)

4. **Batty Peter**. 1-Jul-2004 Future Trends & the Spatial Industry: Part On. Geospatial Solutions E-Newsletter. (http://www.geospatialonline.com/geospatialsolutions/article /articleDetail.jsp?id = 101548)

5. **Batty Peter.** 2005 GIS Databases are Different. Geospatial Solutions E-Newsletter. (http://www.geospatial-online.com/geospatialsolutions)

6. **Dehua Zhao, Byunggu Yu, Dan Randolph**, and **Bong H. Hong.** 2004 Relational Geographic Databases: University of Myoming

7. **Henry F. Korth and Abraham Silberschatz.** 1991 Database System Concepts. 2nd Edition: McGraw Hill. ISBN 0-201-50255-0

8. **Samet H.** 2003 Issues in Spatial Databases and Geographic Information Systems (GIS): Institute of Advanced Computer Studies

9. **Samet H.** 1990 Design and Analysis of Spatial Data Structures: Addison-Wesley. Reading. MA. ISBN 0-201-50255-0

10. **Daniel P. Ames.** 2005 GEOL 403/503. Principles of GIS Lecture notes

11. **ERSI**. August 2004 ArcGIS: Engineered for Interoperability. An ERSI White Paper (http://www.ersi.com)

12. **ERSI**. January 2003 Spatial Data Standards and Interoperability. An ERSI White Paper (http://www.ersi.com)

13. **ERSI**. October 2002 Metadata and GIS. An ERSI White Paper (http://www.ersi.com)

14. **Kenneth E. Foote and Donald J. Huebner.** 1996 Database Concepts. Department of Geography: University of Texas

15. **Kenneth E. Foote and Margaret Lynch.** 1995 Geographic Information Systems as an Integrating Technology. Context, Concepts and Definitions: Department of Geography, University of Texas

16. **Ralf Hartmut Güting**. 2004 Spatial Database Systems. Tutorial Notes: Fernuniversität Hagen, Praktische Informatik IV, D-58084 Hagen, Germany

17. **Gopal Krishna.P, Beema Rao. P, Rajendra Prasad. J, Dr K.S.K. Sai**; The Role of Spatial Information Technology (SIT) & Conventional Techniques in Participatory Natural Resource Management in drought prone areas of Warangal district in Andhra Pradesh; 2004 (http://www.gisdevelopment.net/application/natural_hazards/drought/nhdr0004pf.htm)

18. **Webopedia Computer Dictionary**: What is a Polyline?; 2005 (http:// www.webepedia.com/TERM/P/polyline.htm)

19. **David M. Mount**: Voronoi Diagrams and Fortune's Algorithm; Lecture Notes; Department of Computer Science, University of Maryland, College Park, MD, 20742; 2000

20. **Ioannis Emiris, Elias Tsigaridas**: Solution of Polygon; 9 Sept 2004

21. **Bal Krishna**; Land? Information? System?; 2004 (http://www.gisdevelopment.net/application/lis/overview/lisrp001.htm)

22. **Asima Misra and Deb Jyoti Pal**. 2004 Land Record Information Management System (LRIMS) – A Conceptual framework; (http://www.gisdevelopment.net/application/lis/overview/lisp0018pf.htm)

23. **Vinay Thakur, Ganesh Khadanga, D.S. Venkatesh, Dr D.R. Shukla and S.D. Meena**. 2004 Land Records Management System in India. Technical Framework. (http://www.gisdevelopment.net/application/lis/overview/ ma03188.htm)

24. **Tim Haithcoat.** 2004 Relational Database Management Systems. Database Design and GIS: University of Missouri Columbia, Presentation at the Geographic Resources Centre

25. **Tom Davies.** 2000 Polygons. (http://www.geometer.org/mathcircles)

26. **Kennet E. Hoff.** 2000 Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. Paper presented by Daniel Emmenegger. GDV-Seminar ETH Zürich

27. **Rolf Becker**. 2005 From Data to Information. The Role of Spatial Data Query: MAPS Geosystems, Sharjah, U.A.E

28. **Max J. Egenhofer and Andrew U. Frank**. 2004 LOBSTER:- Combining AI and Database Techniques for GIS. National Center for Geographic Information and Analysis: Department of Geography, Broadman Hall, University of Maine, Orono, ME04469, USA

29. **Cyrus Shahabi.** October 1994 Introduction to Spatial Database Systems. VLDB Journal V3, n4

30. **Gerald Farin**: 22 September 2003 Quadratic Splines over iterated Voronoi diagrams

31. **Dirk Vanderstighelen**: 2004 ETAP Reference Guide Book. Information Management through Geographic Information Systems and Remote Sensing. Chapter 36

32. **Samir Satpathy**: 13 April 2004 Geospatial interoperability – ORDBMS approaches

33. **Health Geomatics**: 2004 Geomatics and GIS: Definitions and Scope: MIM Centre, School of Informatics, City University, London.

Glossary of Terms

Accuracy: The extent to which an estimated value approaches its true value. GIS data have both *postional* accuracy (closeness of the GIS location of a feature to its true location) and *attribute* accuracy (whether the attributes of the feature are correct)..

Attribute: A description or characteristic of a geographic feature in a GIS, typically stored in a table and linked to the feature. For example, attributes of a road might include name, date of construction, width, and right-of-way.

Base map: A map containing geographic features such as roads and streams that are unlikely to change and can be used as a locational reference. Other data layers such as parcels are then fit to the base map.

Coordinate Geometry (COGO): Abbreviation of the term COordinate GeOmetry. Land surveyors use COGO functions to map parcel boundaries based on the legal description, often resulting in more precise locations and boundaries than digitizing existing maps.

Coordinate System: A reference system used to define horizontal and vertical distances on a map. A coordinate system is usually defined by a map projection, a spheroid of reference, a datum, one or more standard parallels, and a central meridian. Common coordinate systems include latitude-longitude, UTM, and State Plane. Data layers must be in the same coordinate system to be used together.

Database: A collection of interrelated data. A GIS database includes data about the position and the attributes of geographical features, also known as spatial data.

Database management system: provides for input, storage, and retrieval of the data.

Database schema: A system for organizing data in the database, usually based on a set of keys that link the data. For example, roads may be tied to maintenance records, civil township, or parcels.

Digital elevation: A digital representation of elevation or other continuous surface, typically used to represent

Digitize The process of converting a map to digital format.

Digitizer A device consisting of a table and a cursor with crosshairs and keys used to digitize a map, OR the person using a digitizing device to facilitate data conversion.

Edge Matching: The process of ensuring that details along the edge of two adjacent map sheets match correctly when scanned or digitized.

Geographic Information System (GIS): A computer system which permits the user to display, examine, manipulate, and analyze numerous layers of spatial data. A GIS is usually considered to include computer hardware, software, geographic data, and personnel

Georeference: Establish the real-world coordinates of digital data such as aerial photographs or GIS maps.

Global Positioning System (GPS): A system of orbiting satellites providing precise time and position information which enables a GPS receiver to calculate its geographical location. Originally developed by the U.S. military, GPS is now widely used for surveying, navigation, and even recreation.

Grid: One of many data structures commonly used to represent map features. A grid or rasterbased data structure is composed of cells of equal size arranged in columns and rows. The value of each cell, or group of cells, represents the feature value such as land cover.

Image: GIS data in the form of a graphic representation or description of a scene, rather than a database with attributes. Satellite images and aerial photographs (also a kind of image) are often used as the source for GIS data development.

Large scale: In mapping, it means to cover small areas in great detail

Latitude/Longitude: A spherical reference system used to describe locations on the Earth's surface. Latitude is the angular measurement north or south of the Equator. Latitudes are commonly referred to as parallels. Longitude is the angular measurement east or west of the Prime Meridian, an arbitrarily chosen line that passes through Greenwich, England. Longitudes west of the Prime Meridian are often stored as negative numbers. The distance covered by one degree of longitude varies as you move north or south. Locations in Indiana range from about 85 to 88 degrees West (longitude), and 38 to 42 degrees North (latitude).

Layer: A set of thematically associated data representing a single theme, such as soils, streams, roads, and land use. A layer is called a coverage or theme in some software packages.

Minimum Mapping Unit: The minimum size or dimension for features to be mapped for a given map scale. For example, long narrow features such as streams and rivers will be represented as lines rather than areas if their width is less than 1/10 inch. The minimum mapping unit for SSURGO soil maps is 5 acres, which means that soil types occupying less than 5 acres are not mapped.

Overlay A common GIS operation in which several maps are combined to perform an analysis. For example, an overlay function would be used to determine areas on well-drained soil, zoned for development, and currently in agricultural land use.

Polygon Area data in GIS. Areas are called polygons, because they are mapped using a large number of lines. Other kinds of features are lines and points. Polygons, lines, and points usually have attributes that describe the geographic feature they represent.

Precision The level of detail known about a value. Precision often specifies how many significant digits are available for coordinate values (e.g. single precision vs. double precision), but it does not imply anything about the accuracy of the value.

Projection A mathematical model for translating the Earth's three-dimensional surface onto a flat plane.

Raster : A cellular data structure composed of an array of rows and columns. The structure is commonly used to store image data. Raster is often referred to as grid.

Rectify To give real-world coordinates to an image (such as an aerial photo).

Relational Database Management System (RDBMS): Software that can access data organized in tables that may be related together by a common "key" item, field, or column. An RDBMS has the capability to recombine the data items from different files, providing powerful tools for data usage. This also helps with data maintenance, because changes made by one user or department are seen by all.

Remote Sensing: The science of observation without direct contact. Examples are photographs from airplanes, satellite images, and radar data.

Resolution The size of the smallest feature that can be mapped and measured. The larger the map scale, the higher the possible resolution (For example an image with a scale of 1:12,000 may have a resolution of 1 meter, while an image of scale 1:100,000 may have a resolution of only 30 meters).

Scale: The relationship between distance on the map to distance on the ground, often expressed as a representative fraction of distance, such as 1:100,000 (one unit of distance on the map represents 100,000 of the same units of distance on the Earth), or two different units such as 1 inch=1 mile or 1 inch=2,000 feet. Scale does not have an absolute meaning with

digital data, since once digitized the data can be used at any scale (although data should never be used at a scale larger than the original map scale).

Scanner: A device which makes a digital image of a paper map. A scanner does not automatically prepare a map to be digital data.

Topology: The geometric relationship between objects located in space. Often includes terms like adjacency, containment, and connectivity, which help in performing GIS analyses.

Vector A way of storing and manipulating data that is based on a linear representation rather than a grid cell or raster. Vector data allow more accuracy in storing features that have well-defined boundaries.

Appendices

ANNEX A: Geometry Concepts

This section reviews several geometric concepts necessary for analysis and manipulation of objects on a map.

Vectors

A Vector is defined as any object that has both magnitude and direction, e.g. force, velocity, etc. Figure 4 below shows two points P = (1, 3) and Q = (4,1). The displacement from P to Q is a vector *v*=(3,-2), calculated by subtracting the coordinates of the points individually i.e. (4-1, 1-3). In other words to get from P to Q we shift down by 2 and to the right by 3.

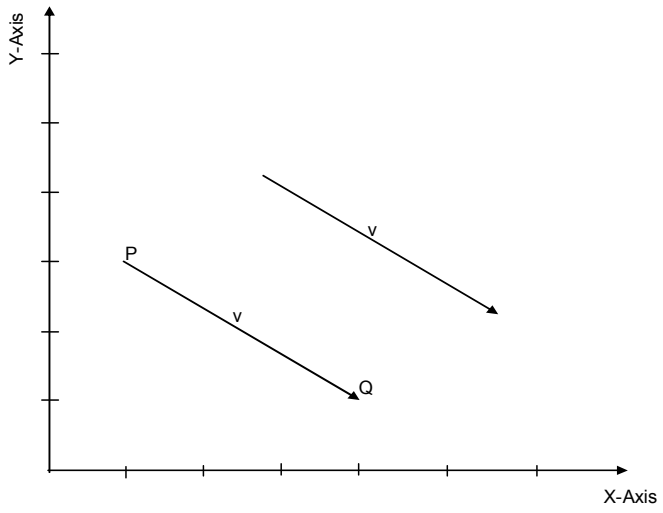

Figure 15 : Representation of a vector

The difference between two points is a vector: $v = Q - P$

The sum of a vector and a point is a point: Q = P + *v* in other words Q is formed by displacing P by *v*

Thus an *n*-dimensional vector is represented by an *n*-tuple : $w = (w_1, w_2, \ldots, w_n)$

Vector Arithmetic

Vector Addition

Given two vectors a = (x_1, y_1) and b = (x_2, y_2), the sum of the two vectors a + b = $(x_1+ x_2, y_1+y_2)$. The sum is a vector as shown on Figure 16 below.

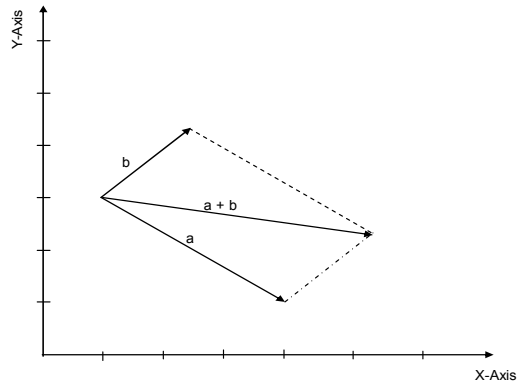

Figure 16 : Vector Addition

Scalar Multiplication

A vector *a* can be multiplied by scalars *s* to get *sa* e.g. if *a* = (2,3) and *s* = 6 then *6a* = (12, 18)

Vector Subtraction

A vector *a* can be multiplied by negative scalars -*s* to get –*sa* for example if *a* = (2,3) and s = -1 then (-1)*a* = (-2, -3) which is basically a vector in the opposite direction.

Vector subtraction *a* – *b* is then *a* + (-1)*b*

Affine Combinations of Vectors

A linear combination of m vectors $v_1, v_2, \ldots, v_m$ is a vector of the form $w = a_1v_1 + a_2v_2 + \ldots + a_mv_m$. Where $a_1, a_2, \ldots, a_m$ are scalars. We say the linear combination is Affine if $a_1 + a_2 + \ldots + a_m = 1$ for example $3a + 2b - 4c$

Convex Combinations of Vectors

A convex combination arises as a further restriction of an affine combination. Not only must the coefficients of the linear combination sum to unity but each coefficient must be non-negative. That is $a_1 + a_2 + \ldots + a_m = 1$ and $a_i \geq 0$, for $i = 1,\ldots,m$. As a consequence, all a_i must lie between 0 and 1. As an example $0.3a + 0.7b$ is convex

Magnitude of a Vectors

Given a vector $w = (w_1, w_2, \ldots, w_n)$, the magnitude (or length or size) of the vector denoted by $|w|$ is defined as the distance from the tail to head of the vector.

$|w| = \text{sqrt}(w^2_1 + w^2_2 + \ldots + w^2_n)$ on the basis of Pythagoras' Theorem

Unit Vector

It is sometimes useful to scale a vector so that the result has unit length. This process is called normalising a vector, and the result is called a Unit Vector also referred as direction.

The unit vector denoted $\hat{a} = a / |a|$ OR better still; $a = |a| \hat{a}$

Perpendicular Vector

Given the vector $\mathbf{a} = (a_x, a_y)$, then we define the vector $\mathbf{a}^{\perp} = (-a_y, a_x)$ as the counter clockwise perpendicular vector to **a**.

The following are properties of $\mathbf{a}^{\perp}$:

- $\mathbf{a}.\mathbf{a}^{\perp} = 0$
- $|\mathbf{a}| = |\mathbf{a}^{\perp}|$

The Dot Product

The Dot Product of two vectors is simply the sum of the products of the corresponding elements. Given two vectors $v = (v_1, v_2, \ldots, v_n)$ and $w = (w_1, w_2, \ldots, w_n)$, the dot product, denoted :

$$\boldsymbol{v}.\boldsymbol{w} = \sum_{i=1}^{n} v_i w_i$$: this produces a single scalar value

The Dot Product is useful in the calculation of the Angle between two vectors or rather two intersecting lines as shown on Figure 17 below:

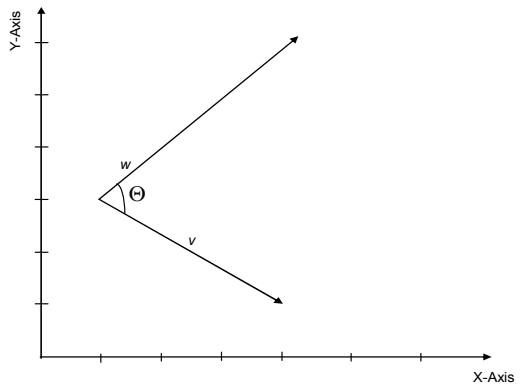

Figure 17 : Angle between two vectors

The Dot Product $\boldsymbol{v}.\boldsymbol{w} = |\boldsymbol{v}||\boldsymbol{w}| \text{ Cos } (\Theta)$

Hence $\text{Cos } (\Theta) = \boldsymbol{v}.\boldsymbol{w}/|\boldsymbol{v}||\boldsymbol{w}|$

Its also important to note that if :

- $\Theta = 90^o$ (e.i. the lines are perpendicular) then $\boldsymbol{v}.\boldsymbol{w} = 0$ since Cos(90) = 0: We also say these vector are **Orthogonal** or **Normal**
- $\Theta = 0^o$ (e.i. the lines are parallel) then $\boldsymbol{v}.\boldsymbol{w} = 1$ since Cos(0) = 1
- $\Theta > 90^o$ then $\boldsymbol{v}.\boldsymbol{w} < 0$ (negative) since $\text{Cos}(\Theta) < 0$ for $\Theta > 90^o$
- $\Theta < 90^o$ then $\boldsymbol{v}.\boldsymbol{w} > 0$ (positive) since $\text{Cos}(\Theta) > 0$ for $\Theta < 90^o$

The Cross Product

The Cross Product, also called Vector Product, of two vectors ***a*** and ***b*** is another vector in the ***z*** direction as shown on Figure 18 below.

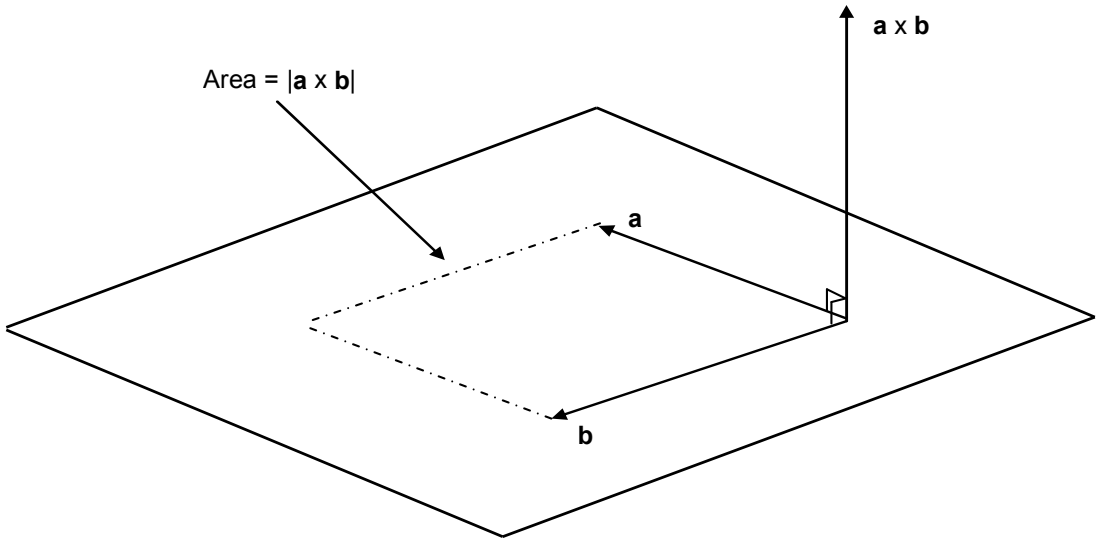

Figure 18 : Cross Product

Given 3-D vectors $\mathbf{a} = (a_x, a_y, a_z)$ and $\mathbf{b} = (b_x, b_y, b_z)$, their Cross Product denoted **a** x **b** is defined in terms of standard unit vectors **i**, **j** and **k**. Where:

$$\mathbf{a} \times \mathbf{b} = (a_y b_z - a_z b_y)\mathbf{i} + (a_z b_x - a_x b_z)\mathbf{j} + (a_x b_y - a_y b_x)\mathbf{k}$$

The Cross Product has the following properties:

- **a** x **b** is perpendicular (orthogonal) to both **a** and **b**
- The length of **a** x **b** is equals the area of the parallelogram formed by **a** and **b**.
  | **a** x **b** | = |**a**||**b**|Sin(Θ), where Θ is the angle between **a** and **b** measured from a to b or b to a whichever produces an angle less then 180^o. **a** x **b** = 0 if and only if, **a** and **b** have the same or opposite direction or either has length zero.
- The direction of **a** x **b** is given by the right-hand rule. When working in a right-handed system the fingers of the right hand fingers point in the **a** and **b** direction then a x b will point in the thumb direction