Control and Cybernetics

vol. **35** (2006) No. 1

Decision support based methods to facilitate 3D volumetric locking in a new peer to peer based spatial database system

by

Gregory Vert[1], Ashley Morris[2], J.S. Heaton[3]

[1]Department of Computer Science, University of Nevada, Reno, NV, USA

[2]School of CTI, DePaul University, Chicago, USA

[3]Department of Geography, University of Nevada, Reno, NV, USA

e-mail: gvert@cs.unr.edu

Abstract: Spatial databases present challenges to data management not found in typical RDBMS. Among these is the desire of users for highly distributed access to data. A new peer to peer database model is being developed that supports high distribution and concurrency of information access. Some aspects of the model are presented in this paper. However the new peer to peer model faces the critical problem of how to lock data as it becomes highly distributed. Conceptual methods borrowed from decision support and data mining are proposed to create a new type of locking model based on 3D volumetrics. The methods and mathematics proposed in this paper can become the basis for the implementation of a new type of peer to peer based spatial database system where spatial information flows to where it is needed on demand.

Keywords: peer to peer, database, fuzzy logic.

1. Introduction

As spatial data and GIS systems have become more widely utilized, users have become more demanding of the capabilities and performance they require of a useful system. Traditional RDBMS technology has seen advances in speed, distribution methods and reliability, but the application of such advances has been slow to transfer to the world of spatial database systems. Part of the reason for this has been the fact that spatial data is intrinsically different from other data types because it requires a geometric and contextual relationship to be maintained among spatial objects. Consequently, there are questions that need investigation when developing methods of distributing spatial data for performance purposes. Distribution in conjunction with the contextual nature of spatial data also requires the development of methods to help manage data integrity problems in such an environment.

Several software packages are widely used in conjunction with spatial data management. Most attempt to provide a single centralized repository for all spatial data resources in an organization. Most software created thus far has little or no support for highly distributed spatial data access. Few, if any, address methods dealing with distributed data integrity problems that arise from users at different locations working on the same region of data simultaneously as a result of contextuality. Problems in current systems arise from limitations in the use of underlying technologies and from limitations of the approach they have taken.

As an example, SmallWorld is a centralized system that was introduced in 1990 to manage spatial datasets (Easterfield, Newell and Theriault, 1998). It is presented as an open solution that runs on multiple platforms, including Windows NT and UNIX. SmallWorld spatial data is relatively platform neutral and attempts to model the world that it manages in terms of objects. These objects can have spatial and aspatial attributes and are implemented through the use of a virtual database. This database sits on top of relational technology and can interface to industry-standard relational databases such as Oracle, Ingres and Sybase. Data from these databases are extracted and assembled into SmallWorld objects.

The Spatial Database Engine (SDE) (Spatial Database Engine, 1998) from ESRI is another product on the market that has been created to manage spatial data. It supports large spatial databases that are built on top of commercial centralized repositories of relational database packages common in industry such as Oracle, Informix, DB2 and Microsoft's SQL server. This package provides a central point from which users can retrieve spatial information. Because SDE retrieves from centralized databases located on several different computers it could be considered to be a virtual distributed system.

Concurrency access to the same regions of data by multiple users can be an issue with the SDE. The SDE allows for multiple users to access the data that it controls. However, access to data in SDE is only as good as the underlying databases it retrieves data from. Since most RDBMS's are limited in spatial data management capabilities, SDE will only be as effective to a users perception as the weakest RDBMS behind it.

Oracle has developed a product that is designed to manage spatial data referred to as the Spatial Cartridge. This application provides an integrated set of functions and procedures that enable spatial data to be stored and accessed efficiently. Similarly as to other approaches the Spatial Cartridge uses an Oracle RDBMS to store data. One of the features of the cartridge is the ability to store spatial and attribute data in the same monolithic centralized database (Spatial Cartidge for Windows NT, 1997).

PANDA (Egenhofer, Frank, 1989) is a spatial DBMS system developed at the University of Maine in the Department of Spatial Information Science and

Engineering. It is an extensible database management system. Based on object-oriented software techniques, it was designed to deal with non-standard applications and spatial data. As such, it is a series of object-oriented classes that can be extended, or used to encapsulate spatial data into objects. Application of object orientation methods is promising; however, these efforts seldom address distribution problems, concurrency or the data integrity problems of spatial data in a highly distributed DBMS.

In the above systems, as is the case with most others, users must query a central location for spatial information. Centralization works well for management of data, but when people interact to complete projects it is more typical for users to operate in a peer to peer fashion. In other words, documents are likely to float from user to user on a project for editing and information sharing. This natural pattern of information flow is counter to the concepts of a centralized repository, such as is found in most models of RDBMS. The pattern provides a metaphor for a new type of database model of information sharing based on the concept of peer to peer exchanges of information. In such a system, there is no hierarchy of management, or centralized location for data. Such a model would view chunks of spatial data as regions that reside on the nodes in a computer network. In this model the distributed regions of data collectively become the entire database. Data in this model would flow where it is needed, in the fashion that an amoeba searches for food.

The conceptual model developed as part of this research proposes a new type of spatial database system that supports peer to peer distribution of spatial data over a widely distributed network. In the development of this model, problems of data integrity must be considered where there are overlapping regions with the same objects in them being edited by different users. Traditional locking of data in these cases is not the best solution because spatial data has geometric relationships with other spatial data that somehow must be preserved in any scheme where areas, spatial extents, can overlap. The issue becomes one of how to lock and to what extent to lock when spatial extent overlap occurs.

This paper presents some details of a new peer to peer spatial DBMS (P2PS-DBMS) model and methods based in decision support systems and data mining to develop a new type of locking paradigm based on 3D volumetrics, and organization of thematic spatial data into a layered cube based on semantic relationship of the layers themes. This initial conceptual investigation has lead to many questions that will need to be addressed in future work.

This paper is organized in the following fashion. Section 2 provides a detailed overview of problems with existing spatial DMBS data distribution and the goal of this research. Section 3 provides details of the approach taken and Section 4 presents conclusions and summary.

2. Problems of locking

The goal of this research was to develop a new model for management and distribution of spatial data based on the peer to peer mechanisms typical of team projects. The benefits of such a model would be rapid, localized access to spatial regions of data, parallel simultaneous knowledge capture by many users working on many regions of spatial data at the same time, and high information exchange among users by sharing regional spatial data. Such a system also would need to have new mechanisms for insuring integrity of data when there are conflicts of information about the same given spatial object.

Current approaches to Spatial DBMS suffer from several problems when one wants to provide highly concurrent, peer to peer access to spatial data. In general most spatial packages on the market are not considered distributed because they rely on the older concept of a centralized repository of spatial information. Distribution of spatial data presents problems not found in typical databases. Among these is that a user of spatial data typically needs to see all data surrounding a point of interest. We refer to this as spatial context. The reason that spatial context is important is that editing or analysis decisions often need to consider surrounding spatial information, because everything is related to everything else, but near spatial objects are more related than distant spatial objects. Spatial context can be implemented by presenting a view of a region. In this case a view may be defined as a rectangular region of spatial data that is being operated upon and which is a subset of the overall region defined by the union of all existing views.

Complicating the concept of spatial context is the need to maintain data integrity in overlapping contexts. For instance, if two contexts overlap the possibility exists that two users, working unknowingly of the other, may modify the same spatial object location or properties, such as the coordinates of a line segment.

ESRI, a producer of GIS software notes that there are several types of problems found when trying to merge or reconcile two spatial datasets in a versioning system (ESRI, Versioning, 2005; ESRI, What Is Versioning?, 2005). Merge and reconciliation problems are created by:

- addition of an new spatial object
- deletion of a spatial object
- modification of an existing spatial object.

These types of modification are difficult enough to deal with when trying to reconcile versioned sets of spatial data into one set. They are also the types of modifications that must somehow be addressed by a locking scheme for distributed spatial data where spatial extents overlap, containing the same objects and thus are subject to corruption of information

To deal with this problem a centralized database can distribute a tuple of data which can be locked. The semantics of locking and concurrency management are fairly well defined for such an operation borrowing from principles of fragmentation and concurrency management (Shekhar, Chawla, 2003). There are several methods of fragmentation of tuple data that are employed in a traditional distributed RDBMS. Horizontal fragmentation distributes tuples across a system as subsets of the original data tuples (rows) found in a centralized table. Vertical fragmentation produces the same distributed effect by sending subsets of relations (columns) from a tuple all of which have methods of managing concurrency.

However, both of the above methods for distribution are geared towards management of alpha numeric data that does not have the property of spatial context or nexus. There is no spatial semantic to the fact that a tuple that is being distributed is related to another tuple when they are fragmented and distributed. In other words one row of alpha numeric text in a table has generally no relationship to the tuples of data immediately preceding or following them. However, a spatial context suggests that tuples are related because they can be describing objects in a view. Consequently, a different scheme that addresses spatial context must be developed for fragmenting and distributing spatial data. The benefit of such a scheme is that users can potentially have smaller and custom tailored views of spatial data that are directly related to regions of interest to them. The regional chunks of data can be described semantically as a view where all the tuples of data in the view at the storage level are related to adjacent rows. In this model of related tuples based on a view, regional chunks of spatial data could then be passed between users in a peer-like fashion because a view represents a cohesive atomic unit. However, there are several factors that complicate spatial distribution of small regions of data across a network.

The first problem is how to build a model of all the spatial regions that users may be interested in viewing or editing. The second problem is how to lock and or reconcile changes to the same spatial object whose properties may have been edited differently in two different but overlapping views of spatial data located on different computers. ESRI has a product that attempts to deal with this via versioning and long transactions (ESRI, Versioning, 2004). However, the approach is based on the concept of a centralized database model where methods for version reconciliation are well established.

This paper presents the initial conceptual underpinnings of the concept of volumetric locking in a peer two peer spatial DBMS model (P2PSDBMS) that can be used for locking and support reconciliation of conflicting distributed spatial information. Such a model has the potential to support view fragmentation based on spatial contexts, peer to peer distribution and reconciling differences in spatial properties of objects found in overlapping views. The investigation builds on previous research in new types of fuzzy based spatial databases (Vert, Morris, Stock, 2002, 2003; Vert, Stock, Jankowski, 2002, 2005), fuzzy based query mechanisms (Vert, Morris, Martin, 2004) and suggests a new type of database system that utilizes decision support techniques to create effective 3D volumetric locking in such a system

The model proposed demonstrates how the application of decision support methods can be used to maintain the data integrity of the spatial properties of an object when it is being viewed and edited by multiple users. This is initial work, which has identified many areas that need further investigation. These are discussed in the conclusions and noted in the following sections.

3. Approach

In order to understand locking in the P2PSDBMS model it is necessary to present some current thinking, background and research about such a model. A goal of this research was to investigate the development of an architectural model that could manage problems found in distribution and editing of highly distributed spatial data in a P2PSDBMS. The first phase of this research was to create a local logical representation of spatial data that would allow for the geometric relationship among spatial objects to be analyzed. At the core of this model is the concept of the spatial data simulator (SDSIM). The SDSIM organizes spatial data into three dimensional layers that can be used to support the locking processes proposed later in this paper.

The SDSIM is a container of geographical information. Fig. 1 illustrates the SDSIM concept. Every layer of the SDSIM can instantiate a different layer of

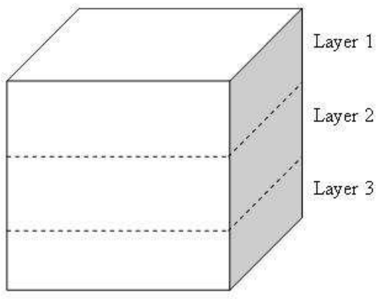

Figure 1. The Spatial Data Simulator concept (SDSIM)

spatial data. For example, if a spatial data set consists of three layers, a road layer, a forest layer and housing layer, layer one of the SDSIM could hold the road layer data, layer two could hold the forest layer data and layer three would hold the housing layer data. The SDSIM represents the underlying stored file data by instantiation of the data's spatial and thematic relationships. When developing the layering concept, it is observed that a unique three dimensional relationship of objects in various layers is created. For instance, if layer 3 has a street system, the intersection of streets is going to have a unique angular and distance relationship to the representation of houses in layer 2. Of note is that not all layers will have a meaningful relationship such as street systems in one layer being related to weather patterns above the streets in another layer. This is discussed later in the paper.

Data organized into the SDSIM is stored locally on a computer and instantiates local spatial data. However, it does not model the distribution of all spatial data in a set across a collection of computers. To accomplish this; the concept of a virtual space was developed. This space is a virtual description of the spatial data space. The data space is defined to be the extent of the bounding hull of all regional spatial data views in a universe of discourse. It is from this data space that users select regional data views of spatial data that are then distributed to local nodes in a network for editing. This concept can be represented as:

$$\begin{aligned}\mathrm{VS} = \{\ \mathrm{BH} \mid &\min(\textstyle\sum \text{datasets x1}), \min(\ \textstyle\sum \text{ datasets y1}),\\ &\max(\ \textstyle\sum \text{ datasets x2 }), \max(\ \textstyle\sum \text{ datasets y2 }) \ \}\end{aligned}$$

where :

VS – Virtual Space
BH – Bounding hull
x1 – minimum lower left hand x value of all datasets
y1 – minimum lower left hand y value of all datasets
x2 – maximum upper right hand x value of all datasets
y2 – maximum upper right hand y value of all datasets.

Visually the space may be represented as shown in Fig. 2, where c, d, a, and b are regional data sets, defining a rectangular region (bounding hull) that describes the extent of the universe of discourse. Of note is that the model assumes that some regions will not have regional data defined for them.

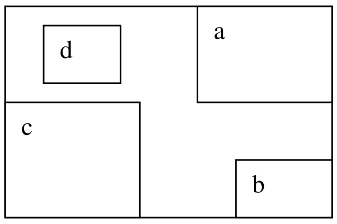

Figure 2. Virtual data space with views of regional data

Initial selection of data from this space is performed by users using a sliding rectangular window of equal width and height. This type of selection operation is what is defined as range or regional query (RQ) (Beckmann et al., 1990). It works on the premise, given a polygon P, of finding all spatial objects (SO) that intersect P. The rectangular polygon represents the contextual view of a region of spatial data that the user is interested in retrieving and can be expressed mathematically as:

$$\text{RQ(p)} = \{\ \text{SO} \mid \text{Geometry (SO)} \cap \text{Geometry (P)} \neq \emptyset\}.$$

At the physical storage level of data, this selection may be going against an initial repository of data, but this is not constrained to being a database. The selection operation, however, is defined using relational algebra as would a typical data base query. The repository is only an initial source of information and its data management functionality has not been defined at this point in the research. At time t_0 user selects data from the repository using the rectangular view area. The selection is defined by a typical relational algebra operator that has been extended for spatial extent selection. This operation is defined as:

$$\sigma_{\texttt{boundpoly}}(\text{dr}) \in \text{VS}$$

where:
dr – data repository at t_o
boundpoly – x1,y1,x2, y2 defines a subset of the virtual data space.

For the purpose of initial model development, selections have been constrained to rectangles that are regular in size. The reason behind this is that it simplifies the research examination of the issues presented when a user wants to select a region already being owned and edited by another user. Future investigation will examine more exotic geometric selection methods.

Regionally selected spatial data are placed on individual machines during the initial phases of the systems operations. However, as the system continues to operate, the extent of fragmentation of regions existing on various machines can increase. As users start requesting data in a peer to peer like fashion from other user's computers, the proliferation of copies of the same regional spatial data can occur. That can lead to problems in data integrity to which fuzzy set theory can be applied to potentially solve.

Peer to peer data swapping and view based selection have the same problems and issues as found in traditional RDBMS, concurrency control and locking. Concurrency control prevents users working on the same tuple in a database from destroying each others changes. A variety of methods have been examined to attempt to deal with concurrency. The simplest of this is that of using a binary lock. Binary locks have two states for a given data item, that of being locked or unlocked. If an item is locked, it can only be written to by the owner of the lock. Other users wishing access have to wait until the lock is released. In order to implement this scheme all that is needed is a binary variable. The algorithm for binary locking is relatively straightforward.

Spatial databases have typically utilized R-tree and B-tree locking on underlying data structures when they are located on the same local machine (Kornacker, Bank, 1995; Chakrabart, et al., 1999; Lehman, Yao, 1981). Traditional methods of locking do not work well for the P2PSDMS system because of the contextuality of spatial data mentioned earlier. This implies that a region of spatial data must be locked, not just a tuple. However, in a system where multiple regions of the same spatial data may exist on different computers, such as found in the P2PSDBMS, binary locking mechanisms become complicated and impractical. An example of how the contextual locking requirement might apply is the situation of moving a street that has houses surrounding it. If a user decides that a street is in an incorrect location and decides to move the street, this user will also need to move houses associated with the street. This can be addressed with a topology rule that states which street lines must not intersect house polygons. However, knowing where and how to apply the locks at the level of database object can require a complicated process of reasoning.

In the P2PSDMS model we have made an initial investigation of a method based on decision support and data mining techniques to maintain data integrity via a new locking mechanism when multiple overlapping copies of the same regional data is shared among many users. To understand this method one has to understand how users may operate when utilizing the peer based model.

In a use case scenario a user may request another user's data for editing. This data may exist on another computer where the users, unaware of each other, edit the same spatial object. If the two users involved are attempting to read data at the same time, this is generally not considered dangerous to the integrity of the data and access can be granted to the data. However, spatial extent overlap can complicate the problem of maintaining data integrity. When selecting a spatial view of data to edit that is being edited by another user elsewhere in the system, there are three possible geometries that may exist in the spatial extent overlap of the local data set and the retrieved set from another persons computer.

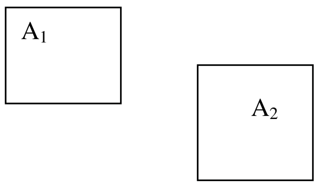

Figure 3. Disjointness of two sets of spatial data

The first spatial extent that must be considered is that of disjointedness as seen in Fig. 3. This condition can be described as

$$\text{IA} = \{\ \text{SO} \mid \text{Geometry}(\text{A}_1) \cap \text{Geometry}(\text{A}_2) = \emptyset\}$$

where:

IA – intersection area

SO – spatial objects
A_1 – data view off user A
A_2 – data view of user B.

The second spatial extent is that of proper overlap, where all spatial objects of one set are included in the other set as seen in Fig. 4. This condition might be described by:

$$IA = \{ SO \mid Geometry(A_1) = Geometry(A_2)\}$$

Equality in this case means one set or the other contains complete geometric copy (subset) of another spatial dataset.

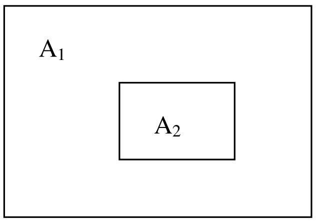

Figure 4. Proper overlap of two sets of spatial data

The final overlapping scenario can be referred to as a partial overlap where the collections of objects in both views are subsets of each other. This can be described as

$$IA = \{ SO \mid Geometry(A_1) \text{ subset } Geometry(A_2) \text{ AND } Geometry(A_2) \text{ subset } Geometry(A_1)\}$$

and is demonstrated in Fig. 5.

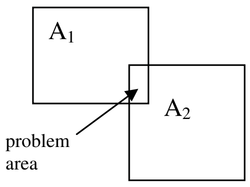

Figure 5. Partial overlap of two sets of spatial data

When overlap occurs, there can be a case where the same object O in both peer views has a modified spatial property as shown in Fig. 5 “problem area”.

In this case, a line segment in view A1 may have the coordinate pair (1,10)(10, 1) and the same line segment may have a coordinate pair of (2,10)(11,2). This creates an editing problem because these coordinate pairs describe the same line segment in two different views A1 and A2. Detection of this fact and reconciliation methods are a problem in the P2PSDBMS model and can be addressed through the use of decision support methods. In this case an advanced type of decision support based locking tailored to spatial data can do a lot to mitigate the need for reconciliation. It is observed that the degree of locking and reconciliation management required depends on the type of overlap that occurs.

As mentioned previously, concurrency management requires locking mechanisms on alpha and number data in a relational database. There are a number of methods of doing this (Kornacker, Bank, 1995; Chakrabart et al., 1999; Lehman, Yao, 1981). However, spatial data has geometric context, in other words a house in a spatial database may be represented by a series of points and line segments. If a user modifies a point in the structure of data, for instance drags the point, all the other points connected to that point via a line relationship should also be locked. What can be noted is that locking in a spatial context is not record based, but based on a region in space that needs to be locked. The farther from the location of editing activity in spatial data a point or spatial object is, the less likely that there will be a need to conduct locking.

In the above scenario, 2D locking could be useful but probably is too simplistic for spatial data. In fact, when one considers that spatial data is organized into thematic layers, e.g. road systems, housing, ground cover, it can be deduced that a change in any given layer's spatial geometry can affect geometries in other layers. As an example, consider dragging road systems intersections in layer 1 which now places an intersection directly on top of a house in layer two. To illustrate this point consider Fig. 6 where the P2PSDBMS model has organized data into the SDSIM data cube (Fig. 1). It is relatively easy to visualize a three-dimensional space where we can see 3D partial overlap.

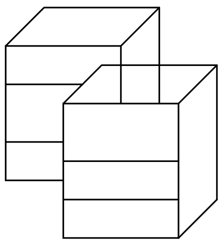
Figure 6. Example of three dimensional overlap of thematic data

Fig. 6 suggests that 3D partial overlap requires a type of locking that is also three-dimensional in nature. We define this to be volumetric locking borrowing on terms from calculus and computational geometry (Anton et al., 2002).

In fact, overlap in 3D and thus the need to lock can also happen as a function of proper overlap. Because of this scenario, this research pointed to the fact that locking in the P2PSDBMS needs to be regional and that the region must be defined as occurring in three dimensional space.

It is also clear that for the disjoint overlap no concurrency management is required since the spatial extents of data are different. Partial overlap and proper overlap require some type of method to resolve conflicts in object spatial properties and thus require some type of locking.

To resolve locking in a three dimensional space, decision support methods seem a natural application because they can classify and make decisions that suggest how to address complicated conflicting questions. In the case of three dimensional overlap, a decision needs to be made about how to lock the space and the degree of locking that might be required. This results from the fact that while there may be a physical overlap in the data being edited by two users in the P2PSDBMS architecture, the nature of editing may not require locking. For example, if there is overlap, two users may be merely viewing the data and thus no locking is required.

This research identified three patterns of editing activity that can occur in an overlap area. These patterns suggest the need to do volumetric locking only if data is being modified, added or deleted.

The first of these is referred to as point editing. Point editing occurs when a user in two sets edits a single point in the overlap area. In this particular example one wants to lock that point from modification in another overlapping extent. Additionally, there is a need to have a degree of potentiality to lock on surrounding spatial object that may have spatial geometries changed as a result of a point modification. The example cited previously is that of modification of value of points x, y, when it is a part of a collection of points describing a house. An example can be found in Fig. 7 where the point being edited has a lock applied to it and surrounding points have a decreasing potential to be locked as a function of distance from the point being edited. Of note is that the concept of potentiality of locking was identified as part of this research, however, was beyond the scope of research. At this place in the investigation it is thought that perhaps application of fuzzy logic and a user specified parameter might be a potential method to implement potentiality locking.

The second type of editing pattern that can occur within an overlapping area is sporadic and unpredictable. This pointed to the need for the development of a regional locking scheme referred to as rectangular volume locking (RVL), which is presented in more detail later. In this type of locking user's editing and modification of data is sporadic and unpredictable, with the overlapping spatial extent. Edits may occur in a three dimensional sense in that they occur on spatial data in various layers of the SDSIM model. Because of the erratic

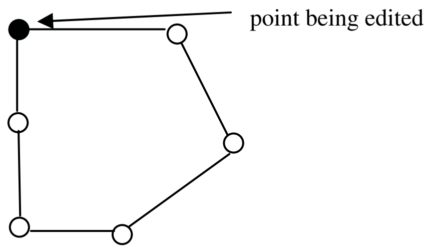

Figure 7. Point editing that may require a degree of locking as a function of distance

nature of such edits an aggressive model of locking had to be developed and that model is proposed to be locking a three dimensional volumetric space for the user conducting the edits. In this case the research has identified the problem that time plays in the decision to lock. A user may show erratic editing active for a time span of tx, and then suddenly shift to a point editing behavior previously mentioned. In this case, to make the data as much available as possible, the model of locking should be adaptive, responding to changes in editing behaviors. Unfortunately, due to the novelty of the concepts and methods developed in this initial research, we could not address this temporal change in editing behaviors but will consider it in future research. An example of this three dimensional RVL locking can be seen in Fig. 7 where the points represent places a user edits over a specified time span. The random nature of edits is clearly seen.

Fig. 8 is a top view of two 3D SDSIM spaces with overlapping spatial extents. In the diagram, the black dots are the objects or points that may be being edited or requested for editing by A1, or A2. One can see that the editing pattern is random and scattered, thus creating a need to lock all objects in the overlap area.

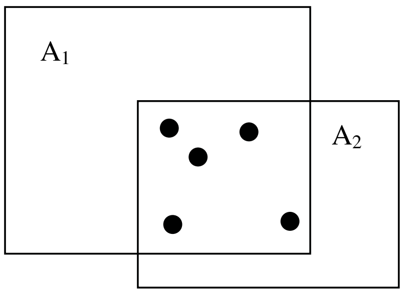

Figure 8. Top view of 3D SDSIM's with overlap for user A1 and A2 leading to locking of a 3D rectangular space

Finally, the third type of editing pattern behavior identified was that of edits within a tightly constrained region of the spatial overlap extent. In this type of editing, user activities can usually be constrained to a circular area. The type of locking required in this scenario is referred to as circular volume locking (CVL) An example of this can be found in Fig. 9, where edits are contained in a circular volumetric area.

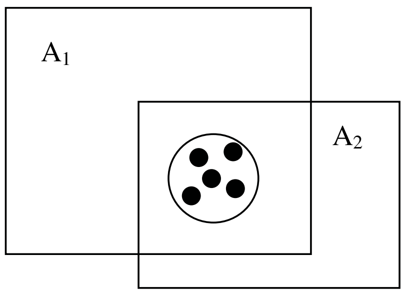

Figure 9. Top view of 3D SDSIM's with overlap for user A1 and A2 leading to locking of a 3D circular space

As with the other types of range locking identified it is observed that this method can have its efficiency improved by defining volume boundary by the minimum and maximum points in any given direction. Such a space might be visualized as a semi deflated cylinder with spiky points representing the minimum and maximum points.

In the identified types of editing patterns and suggested locking required for the P2PSDBMS there needs to be a method(s) of determining what type of locking should be applied. Decision support and data mining methods of clustering and decision trees seem ideal for determining how to lock based on a classification of variety of input criteria.

Initially, it was thought that a decision tree would be the best way to decide which type of lock to apply. Decision trees have been around for a while and much has been written about their application and development (Roiger, Geatz, 2003). Most tree development techniques require some sort of training data that is then placed into a special algorithm which generates a tree (Elmasri, Navathe, 2003) with specific classification outcomes. However, in the case of this investigation into locking, the outcome classifications are known in advance so it seemed that there was no need for training data. The approach became one of creation of the tree based on factors that could affect the classification of which type of lock to apply. In the future, it would be interesting to investigate further the application of training data to dynamically generate a locking classification tree as the system operates.

When the concept of using decision trees to lock was considered, it was realized that another type of classification mechanism, that of clustering could be useful before determining the type of lock to apply. Specifically, clustering looks at grouping of data items into small groups such that these groups have something in common (Garcia et al., 2002). There are several categories of clustering that can be applied to classification problems. Hierarchical methods start with all patterns in a single cluster or as separate clusters, and successively split or merge until a criterion is met. Partitional methods start from an initial grouping and iteratively relocate data points until a pattern criterion is met. This is typically the class that k-means falls into and the pattern is usually determined as a function of feature vector distance (Elmasri, Navathe, 2003) and can be denoted as:

$$\text{distance(rj,rk)} = \sqrt{|\text{rj}_1 - \text{rk}_1|^2 + \ldots + |\text{rj}_n - \text{rk}_n|^2}.$$

The algorithm utilized commonly is K-means which has the following form:

```
Begin
     choose k records as centers for k clusters
repeat
     assign each record to a cluster st. the distance
     between the record and center of the cluster is small

     recalculate the center of each k (mean)
     based on records assigned to the cluster
until reassignment stops
End
```

In the above algorithm one must select the number of clusters believed to exist in the system. For the sake of this research, we selected:

$$\text{k} = (\text{j}/2 * \text{k})/2$$

where:
k – number users logged in
j – number of users editing or modifying data.

This justification for value of k assumes that at any given time, half the users logged into the system could have potential conflicts in modification of data with other logged in users that are actively editing data. However, this assumption has not been tested and probably would change because of circumstances not yet investigated. Additionally, K-means is not the only method of clustering. Because this work is conceptual we did not have a chance to consider the effect of efficiency or accuracy that a fuzzy k-means or c-means algorithm may have on the operation of locking. This is definitely a subject of future research.

Because of the spatial nature of the clustering technique, this method seemed ideal for determining if locking among overlapping sets was even required. To accomplish this several feature vector components are thought to be necessary for calculation of clustering utilizing distance measures and k-means methods.

Clustering in this investigation is initially utilized to determine if locking via a decision tree is required. Clusters mean that data sets are close together in the feature space and should be passed onto the decision tree to determine whether to lock or not lock and how to apply a volumetric lock.

The clustering function is a modified version of k-means and denoted by SEclust(u) for spatial extent cluster. SEclust() takes as a parameter a feature vector u, against which the clustering algorithm is applied. A clustering represents the degree of semantic distance from one spatial data set to another and thus the possibility of spatial extent overlap. A minimal distance leads to clustering and locking. The vector u has the following elements:

- Si.timespan
- Si.Pti
- Si.centroid
- Si.edittype
- Si.criticality

where the above have the following attribute naming template:

Si – is the identifier of a given spatial data set i, e.g. S1 would identify set one of data

Si.timespan.st – the start time of editing a set

Si.timespan.et – end time of editing on the set

Si.Pti – the x, y, z values of point i describing the bounding rectangular hull of the set data points denoted by Si (see above). Note: there are four of these points where N ranges from S1.pt1.x to S4.pt4.z.

Si.centroid – the physical center of a set bounded by its spatial extent

Si.criticality – user defined value to skew clustering to create tighter clusters or looser clusters.

Si.edittype – user defined measure of editing being done on a set where updates and deletes have minimal distances and reads and creates have maximal distances.

The points defining the bounding hull, Si.Pt, and the centroid, Si.centroid, are utilized in a distance measure calculation to determine a degree of overlap in the spatial extent of two sets. The equation for this calculation is given as:

$$\mathrm{a} = \sum_{i=1}^{4} \sqrt{(\mathrm{S1.Pti.x - S2.Pti.x})^2 + (\mathrm{S1.Pti.y - S2.Pti.y})^2 + (\mathrm{S1.Pti.z - S2.Pti.z})^2}.$$

Sets may have an overlap in their spatial geometry but still not need locking due to the point in time when they may be edited. This is especially true if a versioning system is in place and two overlapping sets are on different lineage trees. However, if sets overlap in time, there is a need to measure the temporal distance for use in the k-means clustering. The attributes Si.timespan are utilized for this distance measure (b) and the equation is given by:

$$b = \text{weight}*\frac{1}{\text{S1Overlap} + \text{S2Overlap}}$$

where S1Overlap and S2Overlap are calculated as:

$$\text{S1Overlap} = \frac{|\text{S1max - S2min}|}{\text{S1max - S1min}}$$

and

$$\text{S2Overlap} = \frac{|\text{S1max - S2min}|}{\text{S2max - S2min}}$$

where:
S1max, S1min – max min of time span
S2max, S2min – max min value of time span
weight – utilized to increase the impact of time.

The determination of overlapping time span is complicated in that a span consists of a start and end time, not just a single point in time. Several scenarios of overlap can be seen in Fig. 10.

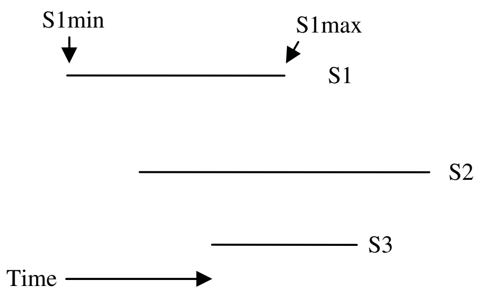

Figure 10. Diagram of overlapping time periods

The algorithm developed to determine time span overlap notices that overlap in time spans can be found first by sorting the start (Si.timespan.st) and end (Si.timespan.et) times of the spans for spatial data sets e.g. S1, S2. If there is overlap, it will show up in the sorted list as an S1 value next to an S2 value. When subtracted, this then becomes a measure of distance between the sets in time space. The algorithm for this is described as follows:

```
if Sx.timespan = Sy.timespan //proper overlap
   return distance 0
else
   Sort(Sx.timespan, Sy.timespan by the start times .st
   and end times .et)

   if a Sx.timespan.st or .et is adjacent to
   any Sy.timespan.st or .et
      subtract the two values
      calculate b term
      return b time value as a measure of time overlap
         and distance
```

Figure 11. Algorithm to determine time span overlap

While sets may have overlapping physical extents, the extent may be stretched in one direction or the other. This is can be due to the fact that sets can have considerably different dimensions as shown in Fig. 12.

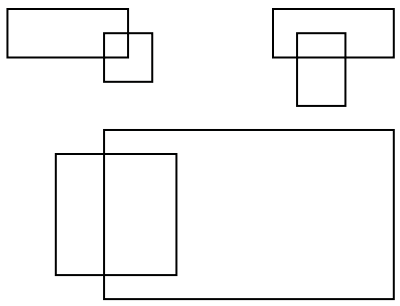

Figure 12. Examples of different types of overlap based on spatial geometries of sets

Because of this, it is also important to get a distance measure based on relative centrality between two sets. This can be accomplished by measuring the distance between the centroids. The distance equation for this is given by:

$$c = \sqrt{\begin{array}{l}(\text{S1.centroid.x - S2.centroid.x})^2 + \\ (\text{S1.centroid.y - S2.centroid.y})^2 + \\ (\text{S1.centroid.z - S2.centroid.z})^2 .\end{array}}$$

Another attribute utilized in clustering is that of user assigned criticality. This attribute allows a user input to tighten or loosen clustering based on factors not already included in the feature vector. A user might use criticality to force locking on all sets regardless of semantic distance or lock only sets very close in semantic distance. For instance, a user may be looking at two overlapping data sets doing "what if analysis" and not want any locking to occur. In this case the data is not going to be saved and a low criticality value might be selected. Criticality can be set by a user to a value of:

high – where distance $= 0$ or user assigned
low – where distance $= \infty$ or a very high value
medium – where low $<$ distance $<$ high value.

Criticality can be denoted as:

$$\mathrm{d(u)} = \left\{ \begin{array}{l} \infty \,|\, \mathrm{u = low} \\ 0 \,|\, \mathrm{u = high} \\ [0, \infty] \,|\, \mathrm{low} < \mathrm{u} < \mathrm{high} \end{array} \right\}$$

Finally, the type of editing being done on a set should also be considered in the clustering. For this purpose we defined a Create, Read, Update, Modify model having the following values:

	Distance
Create	∞
Delete	0
Modify	0
Read	∞ or very high value

$$\mathrm{e(u)} = \left\{ \begin{array}{l} \infty \,|\, \mathrm{u = create} \vee \mathrm{read} \\ 0 \,|\, \mathrm{u = delete} \vee \mathrm{modify} \end{array} \right\}.$$

The Si.edittype is applied to the set in general and not to individual points in the set.

The overall distance vector utilized in the cluster algorithm mentioned previously for Si can be denoted as

$$\mathrm{Vsx = a + b + c + d + e}$$

where:
a – is a measure of spatial overlap
b – is a measure of temporal overlap
c – is a measure of distance between the central points of the bounding hulls of the sets that may not have the same geometries
d – is a user defined measure of criticality that something be locked
e – is a measure of the semantic distance (relationship) between editing changes that can create conflicting information in the database.

SeClust() has the following general logic:

```
SeClust(u)
Begin
   choose (j / 2 * k) / 2 records as centers for k clusters
repeat
   calculate Vsx utilizing feature vector u
   assign each record to a cluster st. the between the record
   and centerof the cluster is small
   recalculate the center of each k based on records
   assigned to the cluster
until reassignment stops
End
```

Figure 13. Cluster algorithm logic utilizing semantics distance measures (Vsx)

From SEclust() a classification is made about which data sets are similar to others and which need to be examined for potential locking. Sets of spatial data (Si, Sj, ...) found in the same cluster are candidates for potential volumetric locking.

However, SEclust() does not comment on how to lock. For this purpose, we decided to utilize a statically defined decision tree in conjunction with the cluster requiring locking. We informally refer to this construct as a "cluster tree". A graphic of the cluster tree is shown in Fig. 14. The semantic of the

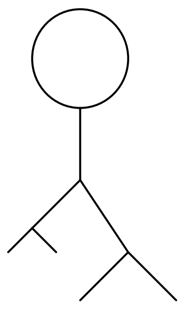

Figure 14. Schematic of a single cluster tree

cluster tree is to use clustering at the top to identify overlapping sets that need some type of spatial locking and then feed the members of the cluster into the tree below as a means of determining how the locking should be applied. A graph of clusters might of course have the cluster tree symbology with every cluster, however it is plausible that some clusters will not have tree's attached to them. The cluster tree concept is an offshoot of seeking a solution to locking in the P2PDBMS and needs to be investigated further.

In development of the decision tree it was decided not to train the tree with test data as is typically done in a decision tree. However, it is an unresolved issue for future research about what role training might have in a system that dynamically develops its own locking schemes in response to user editing activity over given time periods and defines dynamically a function SElock(v) that implements the dynamic locking logic. The locking function that implements the tree part of the cluster tree (SElock(v)) takes a feature vector, v, that contains the following attributes:

Si.points[ ].

SElock() takes as input the u vector from SEclust(), however Si.points[ ] is a new element that contains identifiers for the points or spatial objects that are requested for modification. The contents of Si.points[ ] can be point data, but also could be spatial object identifiers. Therefore feature vector v is defined to be:

$$v = (u, \text{Si.points}[\,]) \; .$$

The decision tree determines which type of lock to apply. For the purpose of this research we define a new type of locking construct, that of volumetric locking. This type of lock is applied to a three dimensional space and to all object or points within the space. Consequently, because of the splitting of spatial data into thematic layers as mentioned with the SDSIM, a volumetric lock is applied to several layers as they are stacked into a three dimensional cube. The semantics of organizing the stacking was beyond the scope of this initial research but needs to be investigated in the future. The reason for this can be seen when one considers stacking a thematic layer of roads and street layout with housing locations in an layer that is immediately above or below the streets layer. In this case there is a semantic relationship between the layers which should be locked volumetrically. However, it makes no sense to stack weather data next to street data because there is no relationship of street systems to patterns of weather, thus no rationale for volumetric locking in this case. The semantics of organizing these layers will be the subject of subsequent research.

The decision tree develops three locking decision outcomes, namely to apply a:

- cylindrical volumetric lock (CVL)
- spherical volumetric lock (SVL)
- rectangular volumetric lock (RVL).

The nomenclature for the lock type is derived from the geometric space described by the type of lock.

A CVL lock is cylindrical in shape. An RVL is rectangular in shape and an SVL is spherical. All are applied across several thematic layers in the SDSIM with a locus centered on the points being requested for locking. Additionally, the selection of the type of lock to apply is based on the geometric patterns of points being requested for locking found in the Si.points[ ] array. More specifically, if a user is requesting a pattern of points that is erratically scattered throughout the overlapping extent (Fig. 7), then an RVL is applied which locks all points and spatial objects in the overlap. This is the most aggressive form of locking. Ideally, one wants to lock as little an area as possible. The next less aggressive type of locking is applied with the CVL. If points requested for locking are found to be located generally in a cluster (Fig. 8), then a cylindrical volume region can be calculated, which then determines which points and object to lock by identifying those inside the region.

The least aggressive locking is the SVL lock. It is generated when Si.points[ ] contains a single point. This point is then locked and a degree of locking is calculated for surround points as a function of their distance from the single point. The degree of locking semantic is not fully defined at this point, but is thought to be useful for applying predictive locking utilizing decision support techniques. It is the subject of future research.

With the above types of locking defined, the decision tree for determining the type of lock to apply is as in Fig. 15.

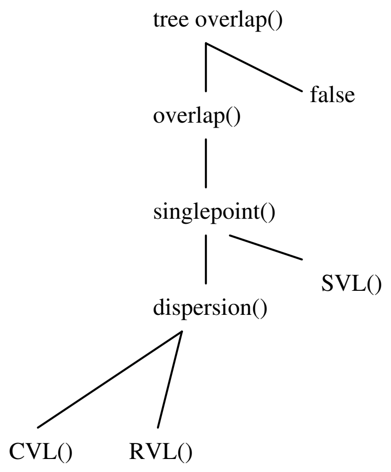

Figure 15. Decision tree component of a cluster tree for determination of CVL, RVL, SVL locking

Above the treeoverlap() function is the cluster part of the tree (not shown) which is implemented via SEclust(u). SEclust() passes its u vector, which is concatenated with a v feature vector that contains some additional information utilized in the locking decision tree processing. The elements of the vector mentioned previously include:

- points modified during time span
- user specified aggressiveness where the value promote locking to a higher level of locking (refer to diagram)

which, when concatenated, includes all the elements of the u vector utilized in SEclust().

The tree determines how to lock primarily by examining the extent of editing changes and the change of spatial object geometry (SOG) over time represented by:

Δ SOG / time.

The first function in the tree treeoverlap() implements logic to verify that the spatial areas for two sets of data that potentially overlap, do in fact overlap. SEclust() only suggests overlap by examining a number of temporal, spatial and editing elements that may be associated with overlap. The algorithm for treeoverlap() is given in Fig. 16.

```
given rectangular regions A,B

if B < A in x axis then swap (A, B) points

assign points to:
   A.ptmax.x
   A.ptmin.x
   A.ptmax.y
   A.ptmin.y
   B.ptmax.x
   B.ptmin.x
   B.ptmax.y
   B.ptmin.y

if A.ptmax.x > B.ptmin.x and
      A.ptmin.y < B.ptmin.y < A.ptmax.y or
      A.ptmin.y < B.ptmax.y < A.ptmax.y then
      return overlap = True
else
      return false
```

Figure 16. Algorithm for treeoverlap() given rectangular regions A, B

In this algorithm max and min points are assigned as shown in Fig. 17.

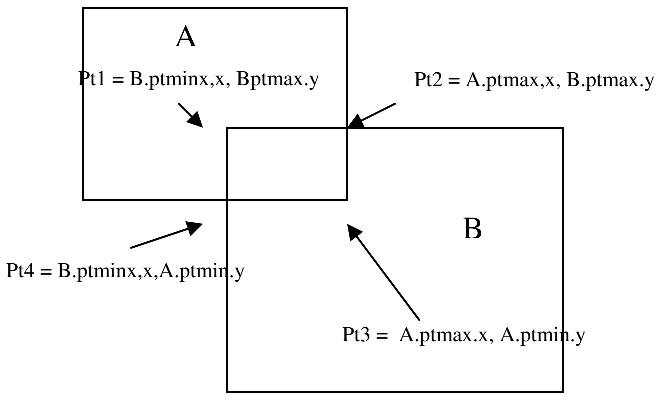
A
Pt1 = B.ptminx,x, Bptmax.y
Pt2 = A.ptmax,x, B.ptmax.y
B
Pt4 = B.ptminx,x,A.ptmin.y
Pt3 = A.ptmax.x, A.ptmin.y

Figure 17. Max and min point assignment for determining if overlap exists

The treeoverlap() function returns true if there is overlap or false if there is no overlap. A true value continues processing with the overlap() function.

The overlap() function determines the coordinates of the overlapping rectangle. This is done by using the max and minimum values shown in Fig. 17:

Pt1 = B.ptmin.x, B.ptmax.y
Pt2 = A.ptmax.x, B.ptmax.y
Pt3 = A.ptmax.x, A.ptmin.y
Pt4 = B.ptmin.x, A.ptmin.y

where the rectangular intersection area (Fig. 17) is denoted as:

I = (Pt1, Pt2, Pt3, Pt4)

The next decision made by the clustertree is to apply a SVL regional lock. This is done by examining the arity of points found in the Si.points[ ] element of feature vector v. If there is a single point in this array then a SVL lock will be generated on data. The degree of lock is calculated as a function of distance from the point. The proposition is that points farther from the singular point need not be modified. Therefore a degree value of locking is applied to all points within a user specified radius. While the logic of this argument seems to make sense, the method and semantics of how one locks a point with a degree of e.g. 0.5 is not understood at this point and the subject of further work and definition.

The equations for applying this type of lock are given by the following:

```
Si.point[1] = lock degree = 1, a full lock
for all Pt.N < radius
  Pt.N.lockdegree = lockcalc() end
```

where:

$$\text{lockcalc(x,y,z,pt.N)} = \frac{1}{\sqrt{(\text{x - pt.N.x})^2 + (\text{y - pt.N.y})^2 + (\text{z - pt.N.z})^2}}$$

and:

$$\begin{aligned}\text{x} &= \text{Si.Pt[1].x}\\ \text{y} &= \text{Si.Pt[1].y}\\ \text{z} &= \text{Si.Pt[1].z}\end{aligned}$$

where:
x, y, z = the center of the spherical locking region
Pt.N – any point in the region bounded by radius in 3D space.

Upon examination, one can see that a locking degree is calculated in 3D space given from the above equations. It is important to note that at this stage of research due to the nature of 3D locking it is important to define which layers of spatial thematic data are stacked adjacent to each other. As mentioned previously, a spatial layer in the SDSIM containing a road data should be adjacent to a layer containing house data by the logic that the placement of roads is related to the location of houses. However, road layers and weather data probably do not have this sort of relationship.

The next function in the cluster tree is invoked if a SVL lock is not applied. The dispersion() function needs to evaluate the degree of dispersion in the requested spatial objects (SO's) or points requested for editing. This function starts the process by sorting all points in the Si.points[ ] feature array element. It then finds the minimum and maximum values in the x and y directions. The next step is to find the minimum and maximum values of the points of a box defining a bounding rectangle and compare this to a radius that would determine the need for circular CVL locking. CVL locking could be implemented by creating bounding rectangle, however, the goal is to lock as little a spatial region as possible. Since a circle has less area than a rectangle, a CVL lock will lock fewer spatial objects than a rectangular lock. The algorithm is shown as follows, given the labels in Fig. 18.

The algorithm to determine if a CVL lock should be applied is given in Fig. 19.

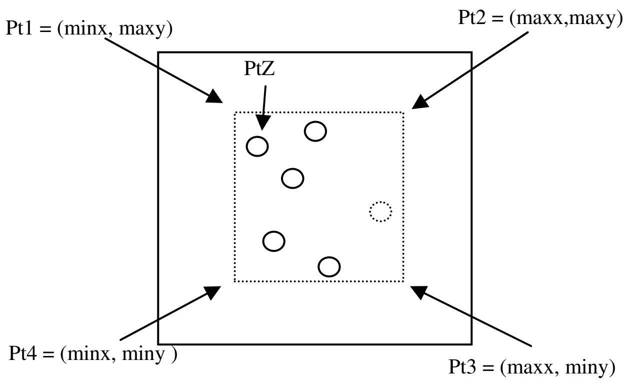

Figure 18. Example of point dispersion in which CVL locking would be selected

```
find points Pt1 -- Pt4 from max, min values in Si.points[]
centerx = .5(maxx - minx)
centery = .5(maxy - miny)
radius = distance(minx, .5(maxy - miny), centerx, centery)
find the point (Ptz) with the smallest distance to any bounding
point Pt1,,,Pt4 that has been requested for editing
Ptzdistance = distance(centerx, centery, Ptz.x, Ptz.y)
if Ptzdistance < radius then
  CVLlock()
else
  RVLlock()
```

Figure 19. Algorithm to determine if CVL locking should occur

CVL locking in the above example has several constraints that must be observed for successful operation. First, the rectangle created by Pt1,,,Pt4 must have equal length sides. Secondly, based on the first assumption, the center of the circular locking areas will be the exact center of an equal sided rectangle. While these constraints seem artificial, they are necessary for this initial attempt. An unequal sided rectangle can lead to problems in the generation of the circular region that may or may not include the correct points to lock. Future work will investigate this effect for improved resolution and efficiency.

The RVL lock is the most aggressive and would lock all points within the bounding hull (dashed line in Fig. 18). If the CVL is applied, locking will prune the region where edits can not occur by approximately half of the area of an RVL lock.

The algorithm for applying a CVL lock is fairly straightforward:

```
CVLlock()
find center point of editing area

distance1 = maxx - minx
distance2 = maxy - miny

if distance1 > distance2 then
   lockradius = distance1 / 2
else
   lockradius = distance2 / 2

for all layers
   for all points with distance from center < lockradius
      lockpoint(pt)
   end
end
```

Figure 20. Algorithm for CVLlock()

Of note is that the locking scheme for CVL and RVL does not have an intrinsic (no z value calculation) three dimensional component such as a SVL lock. To lock in 3D space, the lock is applied in the z dimension to all layers in the SDSIM falling within a projection of the 2D locking area on the layer being initially locked.

The locking algorithm for RVL is much the same as CVL and given by Fig. 21.

```
for all layers
   for all points minx < ptx < maxx and miny < pty < maxy
      lockpoint(pt)
   end
end
```

Figure 21. Algorithm for RVLlock()

A geometric enhancement to the cluster tree presented earlier (Fig. 15) shows the organization of CVL, RVL and SVL locking and decision support processes in Fig. 22.

Overall, the above locking scheme can be mathematically represented by the equation:

f(u,v) = (SElock(SEclust(u),v))

where:

u, v – are the feature vectors for SElock() and SEclust()

f(u, v) - selects and applies the type of lock required.

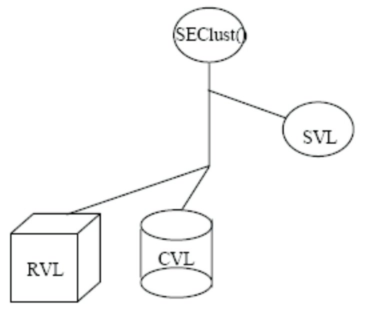

Figure 22. Enhanced schematic of cluster tree (Fig. 15) for SVL, CVL, RVL locking.

Finally, the overall locking process utilizing the above equations and concepts can be modeled with the following steps:

1. Initial editing of a set of data begins
2. A second editing session (new user) starts with overlapping spatial extent
3. The user of the second editing session is notified of potential locking and queried to:
   (a) select the area or region s/he will be
   – editing
   – define expected time span of editing
4. Feature vector u, v is passed to the cluster tree algorithm by invoking f(u,v)
5. f(u,v) applies a minimal volumetric lock, or does not apply locking
6. Go to step 2.

4. Conclusion

The P2PDSDB model presented can provide a new type of highly distributed spatial database system where users have high availability of regionally defined access to the spatial data and surrounding regions. This capability is not provided in the traditional RDBMS. Due to the contextual nature of spatial data, it is important to be able to view objects surrounding a spatial object of interest and to manage differences in spatial properties when views overlap with discrepancies from the editing process.

One of the most difficult questions to answer for this model is how to accomplish locking in this highly distributed peer to peer environment.

The model presented in this paper suggests means for doing sophisticated spatial locking and reconciliation based on decision support and data mining techniques.

The contribution of presented research is the:

- development of the theoretical underpinning of efficient locking in a new type of spatial database system
- application of decision support methods in the process of determining how to lock
- fusion of decision trees, clustering techniques and spatial geometries to propose a method of locking
- development of a model for facilitation of 3D locking through the use of organizing layers of thematic spatial data into the SDSIM model
- development of the concepts of volumetric locking and classifications of such types of locks.

The initial research presented in this paper suggests the need to investigate several areas in further detail, including methods to reconcile and lock distributed spatial objects that appear to be the same but with different spatial properties and what types of underlying storage structures best support this implementation.

It is important to note that this research is preliminary and theoretical so that there are a number of research questions raised by the work that need to be investigated.

One of these is how to determine the ordering of layers and how this can affect the effectiveness of locking. Also, the underlying file structure necessary to support the mathematics has not been investigated.

Future research will consider several question raised by this work. One of these is how to conduct irregular polygonal locking. This could maximize data access by minimizing the three-dimensional locking volume. Additionally, the CVL and RVL locks probably should not be applied blindly to all layers in the SDSIM. The solutions to these questions and more can lead to a potentially much more powerful model of data sharing than is currently seen in typical database technology.

References

Anton, H., Bivens, I. and Davis, S. (2002) *Calculus.* John Wiley & Sons.

Beckmann, N. et al. (1990) R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In: *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, **19**, 2, May 1990.

Chakrabart, K. and Mehrotra, S. (1999) Efficient concurrency control in multidimensional access methods. *SIGMOD 1999, Proceedings ACM SIGMOD international Conference on Management of Data*, June 1-3, 1999, Philadelphia, PA, 25-36.

Easterfield, M., Newell, R. and Theriault, D. (1998) Mangement in GIS. In: *Applications and Techniques*, Smallworld Co. Technical Report.

Egenhofer, M. and Frank, A. (1989) PANDA: An Extensible DBMS Supporting Object Oriented Software Techniques. *Database Systems in Office Engineering and Science*, March 1989.

Elmasri, R. and Navathe, S. (2003) *Fundamentals of Database Systems.* Addison Wesley.

Garcia, H., Jeffery, U. and Widom, J. (2002) *Database Systems.* Prentice Hall.

Kornacker, M. and Bank, D. (1995) High concurrency locking in R-trees. *VLDB '95 Proceedings of 21th International Conference on Very Large Databases*, Morgan Kaufmann.

Lehman, P. and Yao, S. (1981) Efficient locking for concurrent operations on b-trees. *ACM TODS,* **6** *(4), 650-670.*

Roiger, R. and Geatz, M. (2003) *Data Mining.* Addison Wesley.

Shekhar, S. and Chawla, S. (2003) *Spatial Databases.* Prentice Hall.

Spatial Cartridge for Window NT. Oracle White Paper, June 1997.

Spatial Database Engine. ESRI White Paper, April 1998.

What Is Versioning? ESRI, http://www.esri.com/software/arcgis/geodatabase/about/versioning.html: 2005-04-14.

Version Mangement in GIS – "Applications and Techniques", Smallworld Co. Technical Report,

Versioning. *ESRI Technical Paper*, January 2004.

Versioning. ESRI, http://support.esri.com/, 2005.

Vert, G., Morris, A. and Martin, C. (2004) A Fuzzy Based Method for Classifying Semantically Equivalent Spatial Data Sets in Spatial Database Queries. *Proceedings of NAFIPS 2004*, Banff, Alberta, Canada, June 2004.

Vert, G., Morris, A. and Stock, M. (2003) Extension of fuzzy data model to an object-oriented framework for managing GIS datasets. *Transactions on Knowledge and Data Engineering.*

Vert, G., Stock, M., Jankowski, P. and Gessler, P. (2002) An architecture for the management of GIS datasets. *Transactions in GIS.*

Vert, G., Stock, M. and Jankowski, P. (2005) A fuzzy metdata model for management of GIS datasets. *International Journal of Geographic Information Science.* In revision.

Vert, G., Stock, M. and Morris, A. (2002) Extending ERD modeling notation to fuzzy management of GIS datasets. *Data and Knowledge Engineering.*