Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 503 – 507

ISBN 978-83-60810-22-4
ISSN 1896-7094

A Spatial Database Approach to Virtual Worlds Systems

Ovidiu Pârvu
Politehnica University of Timisoara,
Automa tic Control and Computer Science
Bd. Vasile Pârvan 2, Timisoara, Romania
Email: parvu@cs.utt.ro

Ionel Jian
Politehnica University of Timisoara, Automa
tic Control and Computer Science Bd. Vasile
Pârvan 2, Timisoara, Romania
Email: jian@cs.upt.ro

***Abstract* — The rapid pace of computer technology evolution makes web applications suitable for new approaches in taking steps towards a 'more' natural interaction between users of basic online social networking services. One of the properties of "today's web" is the ease by which people develop online applications. This paper focuses on the possibility to use spatial database management systems as the building blocks for virtual world like applications facilitating the development of immersive systems for small developers. We propose a model for such an application, the role of the spatial database and elements that should be taken into consideration in the development process. We underline the advantages and disadvantages of using spatial databases for these systems.**

I. Introduction

TODAY'S computer world revolves around the evolution of the web. The advent of "Web 2.0" generation of internet applications contributed to the integration of computer technologies into everyday's social life. Online social networks and communities, massive multiplayer online games are some of these mainstream applications generating a new type of "virtual" culture. As the web evolves towards a services oriented structure, the classic internet browser concept remains at the core of ordinary user online access. New communities founded on common interest (art, business, education) arise from the user interaction facilitated by such services.

Information is still mainly presented on a page by page basis. Hardware evolution (the increase in processing power, data communication speed and the raise graphics processing units) adds new dimensions to information presentation and user interaction technologies. The switch to a *natural* type of interaction is recommended by the success of 3d multiplayer online games (ex: World of Warcraft) and virtual world systems (ex: Second Life). These applications construct 3d environments that try to mimic a "natural one" (at least from the interaction perspective and not necessary the content). Virtual environments offer users (avatars) the possibility to have unique abilities, to own "virtual objects", to create new content, to exchange information, etc. The effort implied by the development and maintenance of such systems is significant.

Creation of common internet applications and content is accessible to a large number of instructed individuals (as opposed to the creation of complex virtual world systems). As most of the *important content* generation could be handled by application users, the developer has to concentrate on efficient world and general data management; user interaction, communication and management; and finally on the presentation technology (the client - usually with its corresponding 3d engine). Most of these application layers use "in-house" developed technologies which are not accessible to small developers. So, the central part of a virtual world[1] system is spatial, user and world specific entities data management. Two of these requirements could be easily satisfied by the use of high performance database management systems. The third one requires systems capable of managing complex vector and raster data (typical for geographic information systems). Spatial database management systems come in hand for such tasks. Modern RDBMS are merging support for spatial databases through specialized components (ex: Oracle Spatial) into the classical relational model they are implementing. By using such a database system user, spatial and entity data could be treated as a whole in a simple SQL like manner. Complex tasks in data optimization, structuring, storing, distribution, tweaking are assumed by the RDBMS with spatial features.

II. Spatial Databases

When we refer to spatial data manipulation applications (geographical or not), the high storage requirements involved by such systems is the first thing which comes to our mind. Speaking about large amounts of data, today's technology revolves mainly around relational database management systems and the SQL language. Spatial information support is merged into mainstream (general) database management products: Oracle (Oracle Spatial [2]), PostgreSQL, MySQL. This type of spatial database system does not necessary assume that data is used strictly in geographic information systems. CAD systems, graphics modeling software could also benefit from the same type of data management.

SQL based DBMS's work with a limited set of basic data types. In order to integrate spatial features, these systems add support to specific geometric data types. In this paper we are focusing on two dimensional representations although the same principles are extended for higher dimensions support. The coordinate system is a key element of spatial data representation. As geographical fidelity is not an important issue, we're assuming the use of Cartesian coordinates instead of geodetic coordinates.

Object representation in such systems is reduced to modeling element sets of primitive types[3] (either a single ele-

ment or a collection of elements). These types are used to form arrays of vertices or vertices based on simple geometric concepts:

- *points*
- *line strings* (not necessary straight lines, could be connected arcs, or straight lines and arcs compounds)
- *polygons* (not necessary formed by straight lines, used to define boundaries, described by interconnected lines and arcs)

Two well known structures result from the spatial arrangement of objects: partitions (disjoint regions) and networks (graph like structures formed by nodes interconnected by lines / arcs).

A significant issue in spatial applications arises from the computer representation of numbers. Finding the intersection point of two line segments and then testing if the point is on one of the lines might yield unexpected results. One way to solve such problems is represented by the acceptance of a tolerance value. Spatial DBMS offer support for the definition and use of tolerance values. A tolerance value represents the maximum distance for which two points could be considered the same by the DBMS operations / functions.

A two-tier querying model[2] is a usual approach in spatial databases. The first pass (reduced cost) returns a subset containing the exact result set. The second step refines the results of the first one and generates the final result set. This step can be accomplished by the client application lightening the database workload.

The general performance of a database management system is closely related to its data indexing capabilities. The case of spatial databases points towards a spatial indexing scheme. Such indices organize spatial data into meaningful structures which increase the spatial query speed. Spatial indices and the two-tier querying model assume the use of object approximations. A common method used in spatial systems is the minimum bounding box approximation. Many candidate solutions in a query could be rejected in the first step applying such a simple approach. Groups of bounding boxes could be described by larger bounding boxes and so on generating a tree-like structure. This structure is represented by an R-tree index[3]. Other solutions closely related to space partitiong techniques such as quadrees can be used. R-tree index based query performance depends directly on the R-tree structure. A high number of spatial data manipulation operations lead to R-tree performance degradation, thus the DBA is responsible with the index rebuilding policy.

Spatial queries require conditions specific to the spatial database objects. Spatial relationship operations between database elements (or database elements and elements of the same nature) are essential when building spatial join queries. The main topological relationships [3][4] between pairs of 2D spatial database entities are: *disjoint*, *equal*, *touch*, *contains / inside*, *covers / coveredBy*, *overlapboundaryintersect*, *overlapboundarydisjoint*. Other extremely useful operators implemented by spatial DBMS determine whether spatial objects are positioned within a certain distance or help to determine the nearest neighbors.

Aggregation functions in today's relational database systems are taken as a must so that the lack of spatial equivalents would seem unnatural. The result of such functions might be: returning the center of gravity of objects, concatenating geometries, returning the minimum bounding box of object groups, generating a topological union of the objects. Database systems "harvest" spatial data by offering spatial analysis and mining features.

The use of vector data (parametric defined geometric objects) in spatial databases facilitates data manipulation / operations and reduces needed storage. Still, raster information coupled to a grid-based approach is required by many applications. Modern spatial DBMS offer support for both vector and raster[5] combining features in a seaming less way. Raster information can be usually organized in multiple raster “planes”, each having a possible different purpose. Relations between raster data and database reference coordinate systems are established through georeferencing operations.

III. A Spatial Based System Model

The build of an online application that encourages its users to interact through a virtual world system seems a complicated task, but by using today’s technologies, even small developers could take the challenge. We are trying to make a presentation of the most important components involved in modeling future network-centric applications. We start by reviewing the main requirements of a social networking oriented virtual world application:

- *the purpose of the application*. This purpose could be generally described by information exchange and information retrieval between / by system users. Forums, online stores, social networking services, art galleries, massive multiplayer games are just some of the usual online applications that could benefit from a virtual world style of presentation.
- *the rules / laws of the world* (not an object of this paper; are subject only of the purpose of the application).
- *the logical representation of the virtual world*. Application space is modeled similar to the natural, geographical environment (terrain features, borders and coordinate systems). Basic space structures are formed by the definition and manipulation of *world objects* and entities. These structures added to the modeled space form the background on which interaction takes place. Management of the space and structures is one of the most difficult and important issues the system has to address.
- *user interaction*. User interaction is manifested at two levels: direct interaction with the actual system (adding and removing content, manipulating system elements) and interaction with other users (done through system elements changes – based on the first level or real-time or direct interaction through usual means: natural speaking, instant messaging, basically all fast data exchange).
- *virtual world representation*. Directly linked to user interaction, representation of the virtual world does the link between the “behind the screen” user and system

data. When simplified, this means a 3d scene rendering process.

These requirements point to the identification of key application blocks used in the system build. The classic three-tier architecture is the prevalent model in web-centric applications. Building a virtual world system on this model is convenient, but each of the three layers has to be tweaked in order to handle the spatial organization of data:

- *the spatial database layer*. User data, object data, space description data and other information needed by the system have to be managed in a unitary, simple centralized way. Large quantities of data are subject to high performance RDBMS, but a simple SQL approach (without some SQL extensions) is not suited for managing spatial information. The spatial component is essential and a hybrid classical-spatial approach is needed. A possible solution is offered by an Oracle Database which adds spatial data features and support (Oracle Spatial) to its querying engine in a seamless way.
- *middle layer*. Even if RDBMS are high performance solutions, heavy reliance on extremely frequent (required by real-time information exchange) database data manipulation operations (inserting, deleting, updating data) would lead to database abuse and performance degradation since not all the system data has to be stored (but it has to be controlled in a centralized manner). In order to guarantee a high performance, secure and fast operation, a middle layer component that has control over some of the database operations is mandatory.
- *the client*. From the user side of things, a thin client capable of presenting data (3d or not), fetching data directly from the database, receiving "pushed" data from the middle layer component is needed. Even if in-browser technologies are meeting demands for the development of such a client there are still some drawbacks.

A. Data Structuring in a Spatial Database Environment

The simulated world environment is designed around the features of the spatial database system. As long as this system is managed similar to the rest of the data, we could treat the entire database as a "whole". User data and object data are closely related to the spatial representation. The user position in the virtual space, access rights and his virtual estates must all be described in terms of the spatial components. Object position and some of its proprieties (ex: its geometry) are also spatial by nature. The whole complementary information should be treated in a standard relational database way. Data access rights can be handled on two levels: direct user to object transmissible access rights (modifiable by all / owner / group, accessible by all / owner / group) or "geographic" related access rights (for example: A package containing a personal message could be "dropped" by a user on the territory owned by other user in the same group. Depending on the policy implemented on the respective territory, only the second user could access the contents of the package. Other user could publish art in his own virtual exhibition and allow only friends to enter its perimeter. These are a natural ways in which information exchange is taking place in virtual or real environments).

This database structure revolves around space representation (the central component of the system) inside the application. Construction of a virtual complex space requires the use of different layers and types of data. These layers don't have to adhere to real, geographic principles as we're dealing with a virtual environment as long as an "expected" outcome is generated. We underline the main spatial data types affecting the spatial database structure and the virtual environment properties (ex: boundaries, elevation, terrain texture) :

- *vector data* - used for parametric definition of world entities (roads, lakes, territories, estates, tree distribution etc.) as well as object geometry. Definition of the entire object mesh in spatial database terms is not well suited mainly for performance and storage reasons, but virtual world related object information such as the minimum bounding box or 2d object outline would still be very useful(ex: prohibition of overlapping objects). Operations on vector data take place at the spatial DBMS level, so there is no need to invest in developing features that handle this data (intersections, unions, finding neighbors).
- *raster data* - Vector data is not suited to describe all of the virtual world features since it is harder to obtain or generate. Spatial data with a high granularity could lead to performance and storage space loss when stored as vector data. A mix of vector and raster data is suited in this case. Managing data of both types is a key feature when deciding on which database management product is used (ex: Oracle Spatial [2] offers raster support through its GeoRaster [5] component as well as spatial vector data).
- *grid based* - Sometimes a highly regular organization of spatial data[9] is needed by the general application operation, design and performance (ex: management of spatial data on the client and database in a similar manner). Although the grid can't be viewed as a stand-alone type, it combines vector and raster concepts. Hexagonal grids can be well used for keeping track of database fetched data.

Database performance is the key to the general system performance which in turn affects the maximum number of simultaneous users and the quality of system services. Spatial query (spatial information retrieval, spatial joins, etc.) just like any other query is highly influenced by the use of indexes. The spatial indexing scheme is integrated into the DMBS (there is no need to implement auxiliary indexing schemes or "manually" partition space), but there are still cases when external intervention is needed: spatial index degradation analysis and index rebuilding have to be considered. Some DBMS help the extraction of relevant information (used for adjusting and analyzing the virtual world behavior) from the spatial database through data mining techniques by providing specially designed functions.

B. Controlling the System Behavior

Dealing with large quantities of data, the client (on the presentation / interaction level) fetches information directly from the spatial database without the need of an auxiliary tier. But what happens when this data is subject to fast

change? What happens when other users change object or even their own locations in real-time? How do we achieve real-time interaction between users? When trying to answer these questions, the imminent use of middle -tier component is obvious. So, what are the tasks this middle –tier application component has to fulfill?

First of all, these tasks deal with the general control of the system, not only with real-time communication between users. Since the system requires the user to act similar to real situations, the user actions have an immediate effect on the system. One task that has to be accomplished is a permanent check of the client connection health taking right actions in case of failure. The virtual world design starts from the premise that the user moves constantly, interacts with objects (which in turn could be modified constantly) and interacts with other users in real-time. If we were to permanently record the user position in a persistent manner (directly to the database), the generated overhead on the database would be significant as other users data should be updated in the same time. Still, the system has to be aware of the user's location as soon as he moves. It is the burden of middleware [7] to keep track of this information for every user connected to the system. The same approach is taken with objects that are subject to rapid change. User or object modifications could affect other users also and they must be notified immediately. Information must be *pushed* from the middle tier application to all users that are affected by it. One reason to avoid the development of a browser-only client solution is the momentary low support for in-browser *information pushing* technologies.

Selection of users that are affected by a system change is influenced by their position in the virtual space (a users movement affects only the users who "see" him). Once something changes, only users in the neighborhood are informed to update their own virtual world data. Grouping users based on their position in space has to be done by the middle layer application (Figure 1) with the aid of space partitioning structures (for example: quadtrees[8]). It is the application developer who has to build the components that manages these structures. Peer-to-peer communication between the users[6] in the same neighborhood could be used in order to conserve the virtual world's system bandwidth (ex: a user updates data directly from another user without resorting to the middle-tier app or a user informs other users in the neighborhood of the changes he made) as long as the middleware still has control over what is happening in the "world". A peer-to-peer solution is still questionable as long as malicious users could transmit false information and is just as risky as letting users update the database directly. System security (user limitations, access rights, other verifications) must be insured on this intermediary level as well as on the database level.

The grouping is done in close relation to space representation and the coordinate system used by the spatial database. Users modify world data but they cannot update the database directly. It is the middleware which feeds new data to the database securing system persistency (at a regular time interval, scheduled, when the connection is lost, when certain events occur or when a users end their sessions) and it could also control the spatial index rebuilding policy.

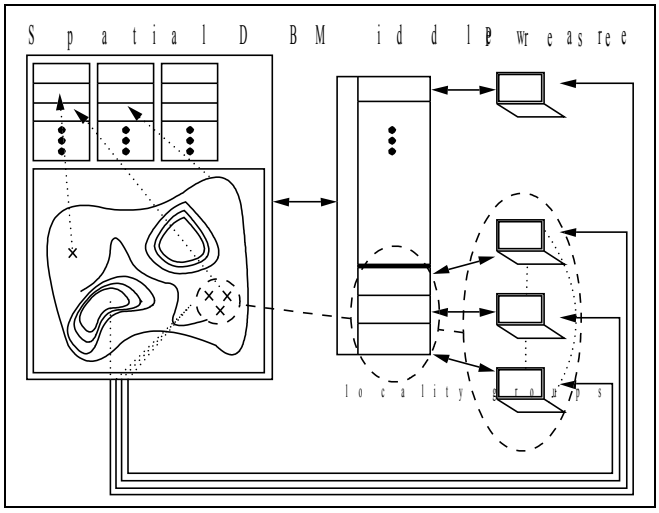

Figure 1: Spatial Based System Model

C. Data Presentation in the Spatial Environment

Viewing the system from the ordinary user's perspective, everything is reduced to the small thin-client. This small application is responsible with the direct user to virtual world interaction. Spatial data is presented in a natural, 3d virtual reality way in order to create an immersive experience. The complexity of the environment affects the technical solution adopted on the rendering side of things and even the client size. The performance of the 3d engine contributes to the immersion as more geometry is rendered and 3d data is handled in an efficient way. Even with reduced input data from the system, 3d content can be generated "on the fly" starting from limited terrain property information.

Initially, there is no information about the virtual world and the client must fetch neighborhood data from the spatial database. Information is fetched for a given perimeter; but the user moves and as he gets closer to the borders, new data must be prefetched without interference to the user experience. Once data from the spatial database gets on the client there's no need to rapidly dispose of it even if it is of no immediate use. What happens if the user returns to a previous place? We cache the data so that the information is immediately available without consulting the database. Then again, what happens when the cached "zone" was updated by other users or by the system? As seen in figure 2, the grid structure comes in hand here as the control of modified spatial data could be handled easily. Each cell could bear a timestamp (controlled by both the database and the middleware application) used either by pushing (control app to client or client to client) or by polling information (at the client's request).

A realistic 3d world representation is dependent on the amount of world data displayed. The maximum possible view distance in a scene can be quite significant in relation to the world data that is to be fetched from the database or

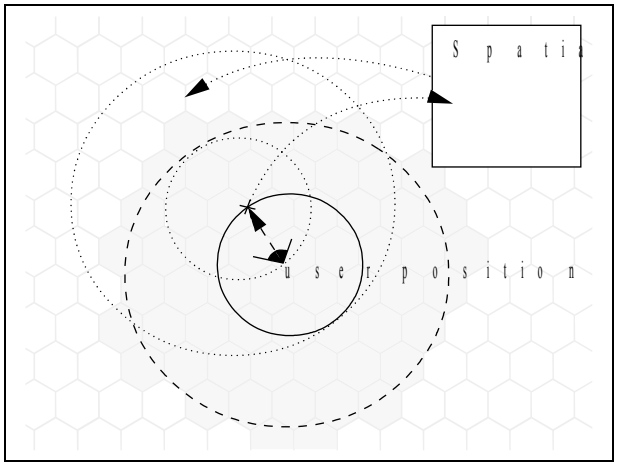

Figure 2: Client fetching data from the spatial database

is already on the client. Direct limitation of the viewing frustum is a simple solution but a progressive level of detail in connection with spatial database layers could be well used (fetching only highly relevant data for distant cells).

IV. Conclusion

Spatial databases are suited as building blocks for large virtual world online systems relieving the developer from building complex data management applications. Spatial data as well as general application data can be treated in a unitary, relational, centralized way resulting in simple and fast virtual world data manipulation. Depending on the chosen DBMS, virtual world data operations such as migration, backup, optimization and data distribution are handled directly by the database system so there's no need to implement in-house, proprietary specialized tools. Other DBMS features for statistics generation and data mining can also be applied for the virtual environment. On the other side, a spatial database management system approach limits the extension of the application beyond the features of the DBMS. Application specific tricks and tweaks are harder to implement leading to possible performance waste.

References

[1] I. Giannopoulos, O. Nikolaidou, D. Anagnostopoulos, "Realistic Virtual Environments Navigable Over the WWW"

[2] C. Murray, D. Abugov, N. Alexander, B. Blackwell, J. Blowney, D. Geringer, A.Godfrind, M. Horhammer, R. Kothuri, R. Pitts, V. Rao, S. Ravada, J. Wang, J. Yang," Oracle Spatial User's Guide and Reference, 10*g* Release 2", Oracle, pp. 3-17 June 2005

[3] R. Hartmut Güting, " An Introduction to Spatial Database Systems *", VLDB Journal (Vol. 3, No. 4),* October 1994

[4] M. J. Egenhofer, "Binary Topological Relationhips", *Proceedings of the 3rd International Conference on Foundations of Data Organization and Algorithms,* pp.: 457 – 472, 1989

[5] C. Murray, J. Blowney, J. Xie, T. Xu, S. Yuditskaya ,"Oracle Spatial GeoRaster, 10 *g* Release 1", Oracle Corporation, pp. 2-31 December 2003

[6] E.Tanin, A.Harwood, H. Samet, D. Nayar, S. Nutanong, "Building and Querying a P2P Virtual World", GeoInformatica , Volume 10, Number 1 , pp. 91-116, March 2006

[7] R. E. Schantz, D. C. Schmidt, " Middleware for Distributed Systems, Evolving the Common Structure for Network-centric Applications", Communications of the ACM, Volume 45, Issue 6, pp 43-48, June 2002

[8] E. Tanin, A. Harwoody, H. Samet, " A Distributed Quadtree Index for Peer-to-Peer Settings", *Proceedings of the 21st International Conference on Data Engineering* , pp.: 254 – 255, April 2005

[9] K.Sahr, D. White, A. Jon Kimerling ,"Geodesic Discrete Global Grid Systems", Cartography and Geographic Information Science, 2003