

# US PATENT & TRADEMARK OFFICE

## PATENT APPLICATION FULL TEXT AND IMAGE DATABASE



( 1 of 1 )

---

<b>United States Patent Application</b>	<b>20070250477</b>
<b>Kind Code</b>	<b>A1</b>
<b>Bailly; Olivier</b>	<b>October 25, 2007</b>

---

### Ranking and Clustering of Geo-Located Objects

#### Abstract

A method of updating information stored in an index associated with spatially-related objects is discussed. The method includes accessing a hierarchical multi-level index having leaf nodes containing information about an object and non-leaf nodes storing information about a number of nodes related to the non-leaf nodes, adding a representation of the object at a leaf node in the index, and traversing parents of the leaf node toward a root node, and incrementing counts of each node in the traversal path.

---

Inventors:        **Bailly; Olivier; (Oakland, CA)**  
Correspondence   **FISH & RICHARDSON P.C.**  
Name and            **PO BOX 1022**  
Address:            **MINNEAPOLIS**  
                          **MN**  
                          **55440-1022**  
                          **US**

Assignee Name   **GOOGLE INC.**  
and Address:     **1600 Amphitheatre Parkway**  
                          **Mountain View**  
                          **CA**  
                          **94043**

Serial No.:        **740210**  
Series Code:      **11**  
Filed:              **April 25, 2007**

<b>U.S. Current Class:</b>	<b>707/2</b>
<b>U.S. Class at Publication:</b>	<b>707/002</b>
<b>Intern'l Class:</b>	<b>G06F 17/30 20060101 G06F017/30</b>

---

*Claims*

---

1. A method of updating information associated with spatially-related models, comprising: accessing a hierarchical multi-level index having leaf nodes containing information about a model and non-leaf nodes storing information about nodes related to the non-leaf nodes; adding information about the model at a leaf node in the index; and traversing parents of the leaf node toward a root node, and incrementing counts of nodes in the traversal path.
2. The method of claim 1, wherein incrementing counts comprises creating a parent node and setting its count to a primary value.
3. The method of claim 1, wherein the hierarchical multi-level index is in the form of a quadtree structure.
4. The method of claim 1, wherein the information about the model comprises a pointer to an identifier for a geo-located 3D model.
5. The method of claim 1, wherein the leaf nodes represent distinct geographical areas.
6. The method of claim 5, wherein a geographical area encompassing the earth's surface is represented in under twenty levels of nodes.
7. A method of searching for geo-located information, comprising: receiving a representation of a geographic search area; determining a level of a hierarchical multi-level index containing geo-related information to review; traversing levels in the index to a maximum lookup level to locate object information at levels in the index; and returning information relating to at least some of the object information located at levels in the index.
8. The method of claim 7, wherein the object information is associated with a geo-coded model.
9. The method of claim 7, wherein the representation of a geographic search area comprises a bounding box.
10. The method of claim 7, wherein the information relating to the objects is stored at leaf nodes in the index.
11. The method of claim 7, further comprising scoring information relating to object information located at the levels in the index, and returning the object information in an order related to the object scores.
12. The method of claim 11, further comprising excluded information relating to objects from the returned information if the related object scores are below a determined level.

---

*Description*

---

**CROSS REFERENCE TO RELATED APPLICATIONS**

[0001] This application claims priority under 35 USC .ctn.119(e) to U.S. Patent Application Ser. No. 60/794,752, filed on Apr. 25, 2006, the entire contents of which are hereby incorporated by reference.

## TECHNICAL FIELD

[0002] This invention relates to data storage in distributed systems, and more particularly to databases for geographic information systems.

## BACKGROUND

[0003] Geographic information systems (GIS) permit for the archiving, retrieving, and manipulation of data that has been stored and indexed according to the geographic coordinates of its elements.

[0004] GIS systems can provide particular challenges for proper database design. That is because GIS information does not fit nicely into a single data type that can be stored in one type of flat-file or relational database. Rather, GIS systems often involve use a variety of data types, such as imagery, maps, and tables. GIS systems may also be very large, such as when they cover a large area and include expansive amounts of information about various points in an area, or cover a very large area like the entire world. Addressing, topographical, or demographic data for various areas may be stored, and may be fairly large to store. In addition, GIS systems that provide graphical representations of data (such as on a map or a 3D representation of the globe) have even more challenges in organizing data, and storing massive amounts of data. Graphical data, such as 3D structures to be placed on a geographic representation, such as on Google Maps or Google Earth, may be especially large and unwieldy.

[0005] When many different pieces of data are stored for a large area, and those pieces themselves are large, it can be a real challenge to organize, update, and search the data, and to present it quickly and accurately to users of a system.

## SUMMARY

[0006] This document describes systems and methods for storing and accessing geo-located content, such as 3D content to be displayed on a model of the earth. The content is stored in a multi-level hierarchical index having embedded therein information about descendants at each level of the index. Such an arrangement of data may permit for quick retrieval of items even as the number of stored items increases to a large number. In addition, the index structure can generally be updated dynamically as objects are added, without the need for timely rebalancing of the index tree, as may be required with other indexing approaches.

[0007] The described systems and methods, in particular implementations, may provide for one or more of the following features and/or advantages. Users of a system may benefit by being able to quickly and conveniently locate objects associated with a very large geography such as the earth, and within a very large database of objects. The system may permit, for example, bounding box queries in lat/lng space (generated by a graphical interface such as Google Earth or Google Maps), and may also prevent showing too many items, such as when a user is viewing a very large geography (such as an entire country). The system may also permit for the ranking of items so as to show them in a preferred order.

[0008] Geographic information providers may benefit by being able to more easily store information about geo-located objects, and may be able to readily search for and recall such objects for display with a related geographic area. In addition, the system may permit for dynamic updates to the index without requiring a rebuild of the index, in certain implementations. Moreover, the data may be stored in a relatively compact form.

[0009] In one implementation, a method of updating information stored in an index associated with spatially-related objects is disclosed. The method comprises accessing a hierarchical multi-level index having leaf nodes containing information about an object and non-leaf nodes storing information about a number of nodes related to the non-leaf nodes, adding a representation of the object at a leaf node in the index, and traversing parents of the leaf node toward a root node, and incrementing counts of each node in the traversal path. The incrementing the count of a parent may comprise creating a parent node and setting its count to a primary value. In addition, the multi-level index may be in the form of a quadtree structure, and the the information about an object may comprise a pointer to an identifier for a geo-located 3D model.

[0010] In some implementations, the leaf nodes may represent distinct geographical areas, and a geographical area encompassing the earth's surface may be represented in under twenty levels of nodes.

[0011] In yet another aspect, a method of searching for geo-located information is disclosed. The method comprises receiving a representation of a geographic search area, determining a level of a hierarchical multi-level index containing geo-related information to review, traversing levels in the index to a maximum lookup level to locate objects at levels in the index, and returning information relating to some of the objects located at levels in the index. The representation of a geographic search area may comprise a bounding box, and the information relating to the objects may be stored at leaf nodes in the index.

[0012] In some implementations, the method may further comprise scoring the information relating to objects located at the levels in the index, and returning the information in an order related to the object scores. In addition, the method may further comprise excluded information relating to objects from the returned information if the related object scores are too low.

[0013] The details of one or more embodiments are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description and drawings, and from the claims.

## DESCRIPTION OF DRAWINGS

[0014] FIG. 1 is a flow chart showing the steps of an insertion mechanism used when objects are added to an index.

[0015] FIG. 2 is a flow chart showing steps for searching within a quadtree data structure.

[0016] FIG. 3 is a schematic diagram showing components in a computer system suitable to be used with the systems and methods described in this document.

[0017] Like reference symbols in the various drawings indicate like elements.

## DETAILED DESCRIPTION

[0018] In the database implementation described here, a quadtree data structure using two distinct node types may be employed. A quadtree structure is generally a tree data structure in which each internal node has up to four children. A quadtree structure may be used, for example, to repeatedly subdivide space into quadrants. The structure has two distinct node types: leaf nodes and grid (or non-leaf) nodes. The leaf nodes are located at the bottom level (max\_level) of the structure, and contain data entries (i.e., geo-located data) to be found in a bounding box associated with that node. Grid nodes lie at levels 0 to

max\_level-1, and contain the total number of data entries in all leaf nodes descendant of the node.

[0019] The earth is represented using latitude/longitude space. At the top level in the index (i.e., level 0), a single grid node (root node) covers the entire space (i.e., (-180,-90) to (180,90)), using the longitude/latitude coordinate system. At each level, the space is divided into  $2^{**}level$  cells ( $2^{sup.level}$ ), and uses a row/column coordinate system in that division at that level. Across the entire grid, a unique coordinate system is then (row, column, level), where row and column are both in the interval  $[0, 2^{**}level-1]$ . The root node is thus at coordinates (0,0,0).

[0020] A compact representation of each node in this coordinate system can be fit on 64 bits, assuming there are no more than 31 levels, which should be more than sufficient (at level 31, each cell would cover an area of  $360/2^{**}31$  degrees in longitude space, or .about. $1.67e-7$  degrees, and  $180/2^{**}31$  degrees in latitude, or .about. $8.31e-8$  degrees). A maximum level of 15 is generally sufficient, and row and column can be compacted on 31 bits each, with level encoded by packing stop bits in this fashion as follows:

[0021] bits:|row|col|11|000000|

[0022] size:|11|2|62-2\*1|

where `l` represents the level in the index. The two `11` bits after row and column are the stop bits, which permits access back and forth between row/col/level and this representation. This may be termed the CellId for a given node.

[0023] Going from parent to children consists of adding 1 to the level, multiplying the row and column by two to find the top/left children and adding 1 to both values for the bottom/right one. To go from a node to its parent, one simply needs to divide both the row and the column by 2 and decrement the level by 1, assuming the node level is strictly positive (i.e., the root node has no parent).

[0024] Using a compact representation of each node may permit for certain advantages, particularly where the model is very large. By nature, the further down one goes in the index, the sparser it should be. Memory usage could be prohibitive if data for each cell at each level is stored (at level 15, there are  $4^{**}15$ , or over 1 billion cells, so storing information for each of these cells may be unrealistic), so the system here may be established to only keep data for populated cells, which may be stored in a hash table.

[0025] The key of this hash table is the CellId of each node. The value type will depend on the node type: grid nodes demand an understanding of how many data entries are present in all leaf nodes covered by a particular leaf node. Therefore, one can simply store an integer quantity, which may represent a number of descendants, and an extent for the bounding box covered by data entries covered by the node. The combination of these two quantities may form an object collection. For leaf nodes, the system may need to store a list of data entries.

[0026] FIG. 1 is a flow chart 100 showing the steps of an insertion mechanism used when objects are added to an index. At act 102, the data is analyzed to identify a leaf node to contain the desired data entry. At act 104, it is determined whether that node exists; if it does exist, it is filled (act 106), and if it does not exist, it is created (act 108) and then filled (act 106). The tree is then traversed from the leaf node up to the root node-creating new nodes where they do not yet exist, and increasing the count of any descendant nodes by one (act 110).

[0027] FIG. 2 is a flow chart 200 showing steps for searching within a quadtree data structure. A

bounding box is initially received that covers a lat/long space (act 202). A result for the search may consist of two parts: a list of data entries, and a list of object collections. Data entries represent single data points, while object collections represent clusters of objects.

[0028] At act 204, a natural level in an index is determined for the bounding box, e.g., a level at which the bounding box contains at most 2 cells in each dimension (row and column). Taking the first cell which completely contains the bounding box causes problems at certain boundaries, e.g., around the point at 0,0. If a query bounding box spans both north, west, south and east of that point, only a root node contains that box entirely, and having to reach all the way up to root level could yield wrong results. By taking a small square in a grid rather than a single cell, one finds the level in the index which contains cells which are comparable in size to the query bounding box and thus may minimize the number of cells to explore in the index.

[0029] At act 206, a determination is made of which maximum level to reach down to (typically, four levels down from the level calculated above, but not more than max-level): max\_lookup\_level

[0030] For each traversed node(act 208): [0031] if it is a grid node which contains fewer than N items (where N is the maximum number of items to output at max\_lookup\_level), directly reach down to descendants at max\_level for this node and traverse them. There can be no more than N such nodes (act 210). [0032] if it is a grid node which contains more than N items (act 212): [0033] if the process is at max\_lookup\_level, output the object collection for this node to result. [0034] otherwise, explore children of this node [0035] if it is a leaf node, iterate the list of data entries and append those which fall into the query bounding box to the result (act 214).

[0036] When the nodes are completed, the data entries in the result may be scored (based, e.g., on popularity, last modification, or other user controlled parameters) (act 216), and the most relevant ones may be output to limit the size of the output (act 218).

[0037] This design may imply that a query run at a sufficiently high zoom level will cause only data points to be emitted, e.g., if the level for a query bounding box reaches max level. Performance of this approach is generally independent of the number of items in the data set or the size of the query bounding box. For any query, a finite number of cells will be explored, which is controlled by N and the number of levels one wants to reach down to. Adding more data to the data set will result in more object collections being returned as results and will require a user to zoom closer to see actual data points but not impact performance of the system.

[0038] FIG. 3 is a schematic diagram showing components in a computer system suitable to be used with the systems and methods described in this document. The system 400 includes a processor 410, a memory 420, a storage device 430, and an input/output device 440. Each of the components 410, 420, 430, and 440 are interconnected using a system bus 450. The processor 410 is capable of processing instructions for execution within the system 2300. In one implementation, the processor 410 is a single-threaded processor. In another implementation, the processor 410 is a multi-threaded processor. The processor 410 is capable of processing instructions stored in the memory 420 or on the storage device 430 to display graphical information for a user interface on the input/output device 440.

[0039] The memory 420 stores information within the system 2300. In one implementation, the memory 420 is a computer-readable medium. In one implementation, the memory 420 is a volatile memory unit. In another implementation, the memory 420 is a non-volatile memory unit.

[0040] The storage device 430 is capable of providing mass storage for the system 2300. In one implementation, the storage device 430 is a computer-readable medium. In various different

implementations, the storage device 430 may be a floppy disk device, a hard disk device, an optical disk device, or a tape device.

[0041] The input/output device 440 provides input/output operations for the system 2300. In one implementation, the input/output device 440 includes a keyboard and/or pointing device. In another implementation, the input/output device 440 includes a display unit for displaying graphical user interfaces.

[0042] The features described can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. The apparatus can be implemented in a computer program product tangibly embodied in an information carrier, e.g., in a machine-readable storage device or in a propagated signal, for execution by a programmable processor; and method steps can be performed by a programmable processor executing a program of instructions to perform functions of the described implementations by operating on input data and generating output. The described features can be implemented advantageously in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. A computer program is a set of instructions that can be used, directly or indirectly, in a computer to perform a certain activity or bring about a certain result. A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment.

[0043] Suitable processors for the execution of a program of instructions include, by way of example, both general and special purpose microprocessors, and the sole processor or one of multiple processors of any kind of computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer are a processor for executing instructions and one or more memories for storing instructions and data. Generally, a computer will also include, or be operatively coupled to communicate with, one or more mass storage devices for storing data files; such devices include magnetic disks, such as internal hard disks and removable disks; magneto-optical disks; and optical disks. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, ASICs (application-specific integrated circuits).

[0044] To provide for interaction with a user, the features can be implemented on a computer having a display device such as a CRT (cathode ray tube) or LCD (liquid crystal display) monitor for displaying information to the user and a keyboard and a pointing device such as a mouse or a trackball by which the user can provide input to the computer.

[0045] The features can be implemented in a computer system that includes a back-end component, such as a data server, or that includes a middleware component, such as an application server or an Internet server, or that includes a front-end component, such as a client computer having a graphical user interface or an Internet browser, or any combination of them. The components of the system can be connected by any form or medium of digital data communication such as a communication network. Examples of communication networks include, e.g., a LAN, a WAN, and the computers and networks forming the Internet.

[0046] The computer system can include clients and servers. A client and server are generally remote

from each other and typically interact through a network, such as the described one. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

[0047] A number of embodiments of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. Accordingly, other embodiments are within the scope of the following claims.

\* \* \* \* \*

---

