I+MON-TREE: Índice Espacio-Temporal para Objetos en Movimiento

María L. Correa; Natalia J. Ortiz; Edilma O. Gagliardi;
Facultad de Ciencias Físico, Matemáticas y Naturales
Departamento de Informática, Universidad Nacional de San Luis
San Luis, D5700HHW, Argentina
{ mlcorrea, njortiz, oli }@unsl.edu.ar

Abstract

With the proliferation of the mobile computing, the ability to index efficiently the movements of mobile objects becomes important. Actually, varied applications to use databases that to maintain information of objects in movement. The main objective of the same is to store and to recover of efficient way the movements realized, for which it is need to count with the indexes. The objects in movement can to move in divers scenes, with or without restrictions. The restrict stage to fixed networks is a special case, where the interest to reside particularly in the positions of the objects in the network and not in a two-dimensional reference. For example, some applications are cars that the move in routs, trains in railway amongst.

In this article, we present our proposal of spatiotemporal access method restrict to fixed networks, called I+MON-Tree. The same account with the ability of to maintain the historic and present information about the positions of the objects that move on the fixed networks. Besides, it is can to solve the types of queries more required in this cases, just as Time Slice, Range, Window and Trajectory, with a good perform in the use of the resources and fundamentally can synthesize all in a method.

Keywords: spatio-temporal databases, moving objects in networks, index structures

Resumen

Con la proliferación de la computación móvil, la habilidad de indexar eficientemente los movimientos de los objetos móviles es cada vez más importante. Actualmente, variadas aplicaciones utilizan bases de datos que mantienen información de objetos en movimiento. El principal objetivo de las mismas es almacenar y recuperar de manera eficiente los movimientos realizados, para lo cual se necesita contar con índices. Los objetos en movimiento pueden desplazarse en escenarios diversos, con y sin restricciones. El escenario restringido a redes fijas es un caso especial, donde el interés reside particularmente en las posiciones de los objetos en la red y no en una referencia bidimensional. Por ejemplo, algunas aplicaciones son autos que se mueven en rutas, trenes en vías férreas, entre otras.

En este artículo presentamos nuestra propuesta de un método de acceso espacio-temporal restringido a redes fijas, llamado I+MON-Tree. El mismo cuenta con la habilidad de mantener información histórica y actual acerca de las posiciones de los objetos que se mueven sobre las redes fijas. Además, se pueden resolver los tipos de consultas más requeridos en estos casos, tales como Time Slice, Rango, Ventana y Trayectoria, con un buen desempeño en el uso de los recursos, y fundamentalmente, se puede sintetizar todo en un método.

Palabras claves: bases de datos espacio-temporales, objetos en movimiento sobre redes, estructuras de datos, índices, algoritmos.

1. INTRODUCCIÓN

La manera en que se deben representar, almacenar y consultar los objetos en movimiento constituye hoy en día una de las áreas de investigación más amplia y reciente, dado que hay una gran gama de aplicaciones actuales que deben tratar con objetos que cambian su posición frecuentemente en el

tiempo, tales como, servicios basados en ubicación, servicios turísticos, comercio electrónico móvil, campos de batalla digitales, entre otros.

Las *bases de datos para objetos en movimiento* atraen tanto al desarrollo científico como práctico de sistemas de información que pretenden capturar y documentar la posición de objetos que están moviéndose en el espacio a través del tiempo. Estas bases de datos son un caso particular de las bases de datos espacio-temporales, que sólo representan cambios relacionados a los movimientos de los objetos, a diferencia de las aplicaciones espacio-temporales, donde la identidad de las componentes geográficas cambia en el tiempo. En las bases de datos de objetos en movimiento ocurre que los objetos sólo cambian sus ubicaciones o formas a través del tiempo; es decir, lo que se modifica es la posición geográfica del objeto en sí. Dentro de este contexto, las aplicaciones que tienen mayor desarrollo como sistemas de información, son aquellas en las que los objetos tienen movimientos regulares [12].

El incremento en el uso de la computación móvil acrecentó la importancia de indexar eficientemente los movimientos de objetos móviles. Los objetos normalmente se mueven en un espacio bidimensional (x, y), donde x e y son las coordenadas espaciales; y se pueden representar sus movimientos a través del tiempo en el espacio tridimensional (x, y, t), donde x e y tiene el mismo sentido, y t representa la coordenada temporal. Además, los movimientos se representan como secuencias de segmentos de líneas conectados, comúnmente denominada trayectoria [10].

En cuanto a las consultas, son de la forma: "informar todos los objetos que pasaron o están pasando por una ventana dada sobre el plano, en un instante de tiempo dado", llamada consulta *Instante de tiempo* (Time Slice Query). Otra consulta podría ser "informar todos los objetos que pasaron o están pasando por una ventana dada sobre el plano, en un intervalo de tiempo dado", llamada consulta *Rango* o *Intervalo* (Range Query). Similar a la consulta antes descripta es la consulta *Ventana* (Window Query), pues toma los mismos argumentos que dicha consulta. La diferencia entre éstas consultas reside en el resultado, ya que mientras Rango o Intervalo devuelve todos los objetos cuyo movimiento es recubierto por la ventana espacial en el intervalo de tiempo dado, Ventana es más precisa y devuelve piezas de los estados pasados de los objetos, que responden a la solicitud. Otra consulta es *Trayectoria* (Trajectory Query), que consiste en dado un objeto y un intervalo de tiempo, se pide informar las posiciones que dicho objeto ocupó durante esas instancias.

Con el propósito de indexar y recuperar eficientemente datos que respondan los distintos tipos de consultas, antes mencionados, se han realizado investigaciones en el campo de los métodos de acceso espacio-temporal y diseñado nuevas estructuras capaces de soportar consultas espacio-temporales que refieren objetos en movimiento.

Nuestro trabajo propone un nuevo método, I+MON-Tree [4], con la capacidad de almacenar y responder consultas espacio-temporales que referencian objetos cuyo movimiento esta restringido a redes fijas. Está basado en el modelo propuesto en [3], con el fin de aprovechar las ventajas del mismo respecto al almacenamiento eficiente de información histórica de los objetos. Además, se utiliza una estructura adicional donde se almacenan los objetos cuyo tiempo final de permanencia en una posición aún no se conoce. Es decir, cuando un objeto arribó a una nueva posición en un instante de tiempo dado, ocurre que se desconoce hasta el próximo arribo a la siguiente posición, el tiempo transcurrido entre las dos posiciones, y mientras tanto, no se puede definir el intervalo de tiempo.

Además de los agregados estructurales mencionados, se buscó resolver un amplio conjunto de consultas espacio-temporales, de modo tal que el método fuese eficiente en el uso de los recursos para almacenar información y resolver las mismas. Así, a las consultas por ventana y rango contempladas en [3], se le sumaron las consultas por instante de tiempo y por trayectoria, las cuales son de importancia debido al uso que tienen en las aplicaciones. Con ello, nuestra propuesta se

diferenció de los métodos espacio-temporal existentes, en el sentido que, por lo general, los otros apuntan a resolver, como máximo, uno o dos tipos de consultas en forma eficiente.

En la siguiente sección se presentan los antecedentes del tema, describiendo aquellos métodos relacionados a nuestro trabajo. En la Sección 3, se presenta I+MON-Tree, incluyendo una descripción de la estructura de datos y de las clases de actualizaciones de la misma, como así también de los algoritmos que permiten responder los distintos tipos de consultas mencionados. Finalmente, en la Sección 4, se describe el trabajo realizado hasta el momento y las propuestas futuras.

2. ANTECEDENTES

Actualmente, existen algunas propuestas para indexar objetos en movimiento. Algunas asumen el movimiento libre de los objetos en un espacio de dos dimensiones [1, 2, 6, 9, 11, 13,14]; mientras que en otras, el movimiento de los objetos se restringe a redes como carreteras, vías férreas, entre otras. Dos estructuras índices para indexar los objetos en movimiento en redes se propusieron en [5, 10], donde ambas usan la misma idea de convertir un problema tridimensional en dos subproblemas con menos dimensiones, utilizando una estructura para indexar los arcos de la red y otra estructura para indexar los objetos en movimiento.

La primera de estas estructuras, denominada FNR-Tree, propuesta en [5], considera la indexación de objetos, que se mueven en forma restringida, en una red fija, en un espacio bidimensional, utilizando un 2DR-Tree. Ésta es una extensión del R-Tree [7] y la idea principal es que para una red de carreteras consistente de *n* arcos, FNR-Tree se considera como una foresta de varios 1DR-Tree encima de un 2DR-Tree. El 2DR-Tree se usa para indexar los datos espaciales de la red; es decir, la carretera que consiste de arcos. Mientras que cada uno de los 1DR-Trees corresponde a un nodo hoja del 2DR-Tree y se usa para indexar los intervalos de tiempo durante los cuales cualquier objeto móvil se movió a través de un arco de la red. De esta manera, el 2DR-Tree permanece estático durante el tiempo de vida de la estructura, mientras no haya cambios en la red [5].

La segunda estructura, propuesta en [10], también almacena los arcos de la red como segmentos de poligonales dentro de un 2DR-Tree y los objetos en movimiento dentro de otro 2DR-Tree. La diferencia entre ambas estructuras reside en la forma en que se indexan los objetos en movimiento. El espacio de movimiento 2-dimensional, sin la extensión temporal, es transformado en un espacio 1-dimensional usando una curva de Hilbert para *linealizar* los segmentos de la red, y en consecuencia las posibles posiciones de los objetos. [8]

Sin embargo, ambos enfoques presentan como principal desventaja el modelo de red usado, donde cada arco en la red puede representarse sólo como un único segmento de la poligonal. Esto provoca que se disminuya el buen desempeño debido al gran número actualizaciones en la estructura que se requiere.

El siguiente método, llamado MON-Tree [3], es un método espacio-temporal capaz de almacenar y recuperar eficientemente los estados pasados de los objetos que se mueven en redes, tomando como ventaja el conocimiento previo de la región, que en este caso es una red, sobre la que se mueven los objetos. Esta estructura tiene la habilidad de indexar dos modelos diferentes de red. El primero, orientado a arcos, compuesto por arcos y nodos, donde cada arco tiene una poligonal asociada. El segundo, orientado a rutas, compuesto por rutas y un conjunto de junturas o uniones entre estas rutas. Esta estructura es capaz de resolver dos clases de consultas sobre estados pasados de la base de datos, Rango o Intervalo y Ventana.

El índice está compuesto por un 2DR-Tree en el nivel superior, que indexa la mínima caja que delimita una poligonal; luego, hay un conjunto de 2DR-Trees en los niveles inferiores, que indexan los movimientos de los objetos a lo largo de la poligonal; y, finalmente, una estructura *Hashing* que determina para una poligonal dada, cuál es el 2D R-Tree inferior que le corresponde.

Se realizan dos tipos de inserciones: la inserción de la poligonal y la inserción del movimiento. La primera inserción se realiza para construir la red básica. La inserción del movimiento se necesita cuando el objeto se crea o cuando varía su vector de movimiento, ya sea su velocidad y/o su dirección. Además, es necesario hacer una inserción del movimiento de un objeto cuando este cambia de poligonal.

3. I+MON-TREE

I+MON-Tree es un índice espacio-temporal que tiene capacidad para almacenar y recuperar información histórica y actual de objetos en movimiento sobre redes fijas, basado en las ideas antes mencionadas. Nuestra propuesta tiene las siguientes características:

- Utiliza un conjunto de estructuras, que almacenan el movimiento de los objetos, en donde los intervalos de posición y tiempo están cerrados. Es decir, mantienen los estados pasados de los objetos en movimiento.
- Utiliza una estructura auxiliar cuya finalidad es almacenar la posición actual de los objetos, en donde tanto los intervalos de posición y tiempo están abiertos; es decir, se desconoce el tiempo final de permanencia de dichos objetos en una posición.

Hay dos tipos diferentes de modelos de redes que indexa I+MON-Tree; el primer modelo es *orientado a arcos*, donde la red es un grafo $G=(N, E)$ compuesto por N, un conjunto de nodos, y E, un conjunto de arcos, donde cada nodo $n \in N$ es la representación de un punto $p_n=(x, y)$ en el espacio bidimensional y donde cada arco $e \in E$ conecta dos nodos n_{1e} y n_{2e} y se le asocia una poligonal $l_e=p_1, \ldots, p_k$ donde cada p_i es un punto bidimensional $1<i<k$, donde k es el tamaño del arco, $p_1=p_{n1}$ y $p_2=p_{n2}$. Cada objeto se mueve a lo largo de la poligonal y su posición *epos* se representa como un número entre 0 y 1, donde 0 indica que el objeto está en el nodo n_{1e} y 1 que está en el nodo n_{2e} del arco. El dominio de la posición de un objeto en movimiento dentro del grafo G es $D(G)=E \times epos$. El dominio del tiempo T es el mismo, luego un objeto en movimiento en este modelo es una función parcial $f: T \rightarrow D(G)$. Si bien este modelo es simple y transparente, no es eficiente para representar redes de transportación, que son las más requeridas por las aplicaciones. Por ello, el segundo modelo se extiende para representar el movimiento de objetos restringido a redes, manejando un modelo *orientado a rutas*. Aquí, la red se representa como un conjunto de rutas y un conjunto de junturas que vinculan esas rutas, es decir, $G'=(R, J)$ donde R es el conjunto de rutas y J el conjunto de junturas. A cada ruta $r \in R$ se le asocia una poligonal $l_r=p_1,\ldots, p_k$, donde cada p_i es un punto bidimensional y $1<i<k$ y k es el tamaño de la ruta. Una posición *rpos* dentro de la ruta se representa como un número real entre 0 y 1, donde 0 significa que la posición es p_1 y 1 que la posición es el punto p_k de la ruta. Una juntura $j \in J$ está representado por dos rutas r_1 y r_2. El dominio de la posición de un objeto en movimiento dentro del grafo G' es $D'(G')=R \times rpos$. El dominio del tiempo T es el mismo, luego un objeto en movimiento en este segundo modelo es una función parcial $f: T \rightarrow D'(G')$ [3].

En la Figura 1 se muestra un ejemplo de cada modelo de red, basados en [3].

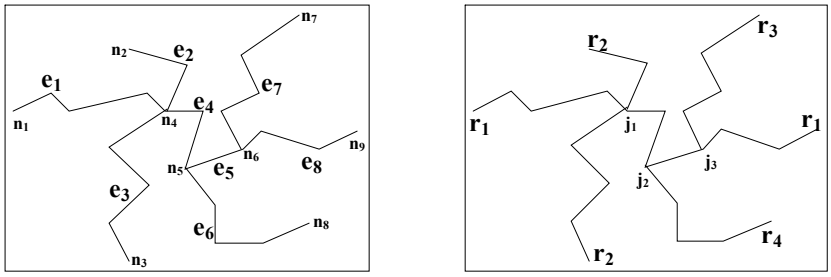

Figura 1: Ejemplo de una red (a) en el primer modelo y (b) en el segundo modelo

En nuestro método se asume que los objetos se mueven a lo largo de poligonales, ya sea asociada a arcos, o asociadas a rutas, según mostramos en los modelos de red.

A continuación, detallamos las estructuras que conforman el índice:

Para la información histórica, se cuenta con dos niveles de estructuras. En el nivel superior, se hallan dos estructuras, un 2DR-Tree y una estructura de hashing; en el nivel inferior se encuentra un conjunto de 2DR-Tree. Cabe aclarar que al detallar estas estructuras se hace referencia al término MBB, el cual se utiliza para denotar a la mínima caja que delimita o contiene un objeto, ya sea poligonal o movimiento de un objeto.

El 2DR-Tree del nivel superior, denominado R_S, indexa la mínima caja que recubre cada poligonal. La información almacenada en los nodos de la estructura es la siguiente:

- Nodos Internos <*mbb, childpt*>, donde *mbb* es la MBB que contiene a todas las MBB`s de las entradas en el nodo hijo, y *childpt* es un puntero al nodo hijo.
- Nodos Hojas <*mbb, polypt, treept*>, almacena la información de la poligonal, donde *mbb* es la MBB que la contiene, *polypt* apunta a la representación real de la misma y *treept* es un apuntador al respectivo R-Tree de aquella.

El Hashing, denominado *H*, almacena el identificador de la poligonal y el puntero al 2DR-Tree del nivel inferior que le corresponde a la misma.

El conjunto del nivel inferior, de 2DR-Tree's, denominados R_{In}, donde *n* es el número de R-Tree's, indexan el movimiento de los objetos a lo largo de la poligonal. La información almacenada en los nodos de la estructura es la siguiente:

- Nodos Internos <*mbb, childpt*>, donde el *mbb* es MBB que cubre los MBB de los nodos hoja, childpt es un puntero a los nodos hijos.
- Nodos Hojas <*moid, mbb, polyid,* $[p_1, p_2]$, $[t_1, t_2]$, *p_tray_back, p_tray_next*>, donde *moid* es el identificador del objeto móvil, *mbb* es la MBB que cubre el movimiento, *poliyd* es el identificador de la poligonal sobre la cual se produjo el movimiento, los intervalos $[p_1, p_2]$ y $[t_1, t_2]$ corresponden a la posición y tiempo respectivamente, en el cual se produjo el movimiento, p*_tray_back* almacena la información relacionada al lugar dentro de la estructura donde se guarda la información correspondiente al movimiento anterior del objeto y finalmente *p_tray_next* almacena la información relacionada al lugar, dentro de la estructura, donde se guarda la información correspondiente al movimiento posterior del objeto.

Por último, el índice de información actual, denominado *I*, es una estructura que almacena cubos abiertos, que hacen referencias a objetos cuyo instante final en una posición aún no está definido, por lo que el cubo está dado por coordenadas espaciales *x* e *y* y un instante de tiempo *t*; en donde *x* e

y constituyen la posición a la que arribo el objeto y *t* el instante de tiempo en que lo hizo. También se mantiene información que hace referencia a los cubos anteriores, los cuales son de determinados por un intervalo de coordenadas espaciales *x* e *y* y un intervalo de tiempo, que corresponden al movimiento realizado desde una posición en instante de tiempo a otra posición en un instante de tiempo posterior; y que nos permite recuperar la trayectoria del objeto.

Además de la información antes mencionada, en la estructura de información actual se hallan los siguientes datos: *moid* es el identificador del objeto, *polyid* es el identificador de la poligonal en la que se encuentra el objeto, *p_next* es el puntero al objeto con tiempo posterior inmediato de arribo a una posición, que se encuentra en la estructura, *p_back* es el puntero al objeto con tiempo anterior inmediato de arribo a una posición, que se encuentra en la estructura, *y p_tray* puntero al cubo anterior, almacenado en estructuras de información histórica, utilizado para mantener la trayectoria del objeto.

En la Figura 2 se presenta un ejemplo de la estructura propuesta, basada en [3].

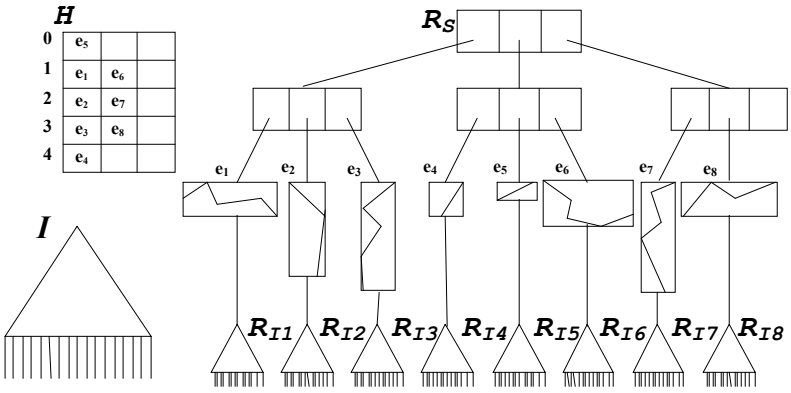

Figura 2: Ejemplo de la estructura I+MON-Tree

Existen dos maneras de actualizar nuestra estructura. La primera se produce al momento de insertar una poligonal y la segunda al momento de realizar la inserción del movimiento de un objeto. En este sentido se siguió la misma política que en [3].

La inserción de la poligonal inserta el identificador de la misma en la estructura Hashing con un puntero a *null*. La inserción de una poligonal en el R-Tree superior es demorada hasta que se realiza la primera inserción del movimiento de un objeto que ha atravesado dicha poligonal. Este enfoque se utiliza para evitar tener en el R-Tree superior una poligonal en las que no se han producido movimientos, manteniendo de esta forma dicho árbol tan pequeño como se posible.

Para la inserción de movimiento se toma como argumento el identificador del objeto *moid*, el identificador de la poligonal *polyid*, la caja construida con las coordenadas de la posición informada *mbb*, la posición a la que ha arribado el objeto p_i y el tiempo t_i en el que se produjo dicho arribo y se procede de la siguiente manera: se verifica si es el primer movimiento que informa el objeto; en caso de que ya se haya registrado con anterioridad algún movimiento de este objeto en la estructura que almacena la información actual, se arma un intervalo de posición con la posición almacenada en el índice y la nueva posición informada, además se arma un intervalo de tiempo con el tiempo anteriormente informado y el nuevo tiempo a registrarse. Con los intervalos construidos y la información de *moid* y *polyid* se produce la inserción del movimiento en las estructuras que mantienen los estados pasados de los objetos. Una vez realizada esta tarea se almacena la última información provista por el objeto en la estructura de información actual. Por el contrario, si es el

primer movimiento del objeto, lo único que se hace es almacenar en el índice de información actual la información provista sobre de dicho movimiento.

4. CONSULTAS EN I+MON-TREE

A continuación explicamos los distintos tipos de consultas soportados por I+MON-Tree, los cuales están acompañados por sus respectivos algoritmos.

4.1 Ventana (Window Query)

Para procesar este tipo de consulta primeramente se debe verificar si la información solicitada se encuentra en la información histórica de los objetos, en cuyo caso se procede con la búsqueda dentro de las estructuras destinadas para dicha información. Una vez finalizada dicha búsqueda se procede con la búsqueda dentro de la estructura que almacena información actual, verificando previamente si es posible que dentro de la información actual se encuentre lo solicitado.

Algoritmo *WindowQueryI+MON-Tree*(R,t_i,t_f,O)
Entrada: R es el rectángulo de consulta, t_i y t_f son el límite inferior y superior respectivamente del intervalo de tiempo y O es el nº de objetos hallados, inicialmente 0.
Salida: Q es el conjunto de objetos que responden la consulta.

1. Si *ÚtilBuscarEnRTree'sInferiores*(t_i,t_f) entonces
2. $Q1 \leftarrow$ *BuscarEnR-Tree'sInferiores*(R,t_i,t_f,O)
3. finsi
4. Si *ÚtilBuscarEnÍndice*(t_i,t_f) entonces
5. $f \leftarrow$ *BuscarPrimeroEnÍndice*(R,t_i,t_f,O)
6. $Q2 \leftarrow f$ *BuscarSiguientesEnIndice*($f.moid$,R,t_i,t_f,O)
7. $Q2 \leftarrow Q2$ *BuscarAnterioresEnÍndice*($f.moid$,R,t_i,t_f,O)
8. finsi
9. $Q \leftarrow Q1$ $Q2$
10. Retornar Q

Donde:

ÚtilBuscarEnR-TreesInferiores(t_i,t_f) verifica si puede haber objetos que respondan a la consulta en los R-Tres inferiores que almacenan información histórica, comparando intervalo de tiempo [t_i,t_f] con tiempos mínimo y máximo obtenidos a partir de los distintos almacenamientos.

BuscarEnR-TreesInferiores(R,t_i,t_f,O) recupera los objetos almacenados en los distintos R-Tree's inferiores que intersecan el rectángulo R e intervalo [t_i,t_f].

ÚtilBuscarEnÍndice(t_i,t_f) verifica si puede haber objetos que respondan a la consulta en índice, comparando intervalo de tiempo [t_i,t_f] con tiempos mínimo y máximo guardados en índice.

BuscarPrimeroEnÍndice(R,t_i,t_f,O) recupera el primer objeto en índice que interseca el rectángulo R e intervalo [t_i,t_f], mediante una búsqueda secuencial.

BuscarSiguientesEnÍndice($f.moid$,R,t_i,t_f,O) recupera los objetos con tiempo mayor al primero hallado, que responden la consulta, siguiendo los punteros *p_next*.

BuscarAnterioresEnÍndice($f.moid$,R,t_i,t_f,O) recupera los objetos con tiempo menor al primero hallado, que responden la consulta, siguiendo los punteros *p_back*.

4.2 Rango o Intervalo (Range Query)

El procesamiento de este tipo de consulta es muy sencillo, ya que a los pasos que se siguen para procesar la consulta por ventana, se incorpora aquel que permite la remoción de duplicados y devolución solo de identificadores de objetos.

Algoritmo *RangeQueryI+MON-Tree*(R,t_i,t_j,O)
Entrada: R es el intervalo de consulta, t_i y t_j son el límite inferior y superior del intervalo de tiempo y O es el nº de hallado, inicialmente 0.
Salida: Q es el conjunto de identificadores de objetos que responden a la consulta.
1. $Q1 \leftarrow$ *WindowQueryI+MON-Tree*(R,t_i,t_j,O)
2. $Q \leftarrow$ *EliminarDuplicados*($Q1$,O)
3. Retornar Q

Donde:

WindowQueryI+MON-Tree(R,t_i,t_j,O) permite recuperar información completa de objeto que responden la consulta.

EliminarDuplicados($Q1$,O) permite eliminar aquellos objetos que aparecen mas de una vez, retornando solo sus identificadores

4.3 Instante de Tiempo (Time Slice Quero)

La consulta se procesa de la siguiente manera, primeramente se verifica si la información solicitada se encuentra entre la información histórica de los objetos, en cuyo caso se realiza una consulta de manera similar a Ventana, pero preguntando por el instante de tiempo dado. La verificación se realiza también en la estructura que indexa la información actual, para determinar la existencia de objetos que intersequen el espacio consultado en el instante de tiempo dado.

Algoritmo *TimeSliceQueryI+MON-Tree*(R,t,O)
Entrada: R es el rectángulo de consulta, t es un instante de tiempo y O es el nº de objetos hallados, inicialmente 0.
Salida: Q es el conjunto de objetos que responden la consulta.
1. Si *ÚtilBuscarEnR-Tree'sInferiores*(t) entonces
2. Si *PrimeroEnInfoHist*(R,t,$Q1$) entonces
3. $O \leftarrow O + 1$
4. Mientras *ProximoEnInfoHist*($Q1$) hacer
5. $O \leftarrow O + 1$
6. finmientras
7. finsi
8. finsi
9. Si *ÚtilBuscarEnÍndice*(t) entonces
10. Si $I(l).t < t$ //l es id de ultimo objeto actualizado
11. $Q2 \leftarrow$ *BuscarMayoresEnÍndice*(R,t,O)
12. sino
13. $Q2 \leftarrow$ *BuscarMenoresEnÍndice*(R,t,O)
14. finsi
15. finsi
16. $Q \leftarrow Q1 \cap Q2$
17. Retornar Q

Donde:

ÚtilBuscarEnR-Tree'sInferiores(*t*) verifica si puede haber objetos que respondan a la consulta en la estructura de estados pasados, comparando el tiempo *t* con tiempos mínimo y máximo guardados en ella.

PrimeroEnInfoHist(*R*,*t*,*Q*1) recupera el 1º objeto en la estructura de estados pasados que interseca *R* y tiempo *t*, retornando true si lo halló.

ProximoEnInfoHist(*Q*1) recupera el próximo objeto en la estructura de estados pasados que responde la consulta, retornando true si lo halló.

ÚtilBuscarEnÍndice(*t*) verifica si puede haber objetos que respondan a la consulta en índice, comparando tiempo *t* con tiempos mínimo y máximo guardados en índice.

BuscarMayoresEnÍndice(*R*,*t*,*O*) recupera todos los objetos que responden la consulta, siguiendo los punteros *p_next* a partir del último objeto actualizado, para evitar búsqueda secuencial.

BuscarMenoresEnÍndice(R,*t*,*O*) recupera todos los objetos que responden la consulta, siguiendo los punteros *p_back* a partir del último objeto actualizado, que es incluido en salida si satisface la consulta, para evitar búsqueda secuencial.

4.4 Trajectory Query

El algoritmo para esta consulta recibe como argumento el identificador del objeto del que se desea conocer la trayectoria y procede de la siguiente manera, se recupera de la estructura con información actual la posición en la estructura que guarda la trayectoria de los objetos donde se encuentra el último movimiento almacenado para dicho objeto. Una vez culminado este paso se accede a tal estructura y se comienzan a recuperar todos los movimientos que conforman la trayectoria del objeto, recorriendo las referencias dentro de dicha estructura para obtener las posiciones correspondientes.

Algoritmo *TrajectoryQuery*(*moid*)
Entrada: *moid* es el identificador del objeto del que se desea recuperar trayectoria.
Salida: *Q* es el conjunto de movimientos que conforman la trayectoria del objeto.
1. *f* ← *PrimeroEnTrayectoria*(*moid*)
2. *Q* ← Ř
3. Mientras *f* □□-1 hacer //*f* = -1 si no hay mas mov. en trayectoria
4. *Q* ← *Q* □ *BuscarMovimientoEnInfoHist*(*f*)
5. finmientras
6. Retornar *Q*

Donde:

PrimeroEnTrayectoria(*moid*) permite recuperar desde Índice de Información Actual, la posición del primer movimiento que debe ser recuperado, para armar trayectoria.

BuscarMovimientoEnInfoHist(*f*) recupera el movimiento almacenado en posición f y actualiza f con el próximo movimiento que debe ser recuperado.

5. CONCLUSIONES Y TRABAJO FUTURO

En este artículo proponemos el diseño de un nuevo índice, I+MON-Tree, que permite almacenar y recuperar información histórica y actual de objetos cuyo movimiento está restringido a redes. El

método se basa en el índice propuesto en [3], lo cual permite aprovechar las ventajas que este posee respecto de métodos anteriores en cuanto a los modelos de red usados para la indexación.

Otro de los beneficios que brinda I+MON-Tree es la ampliación del número de consultas que fueron originalmente contemplados en [3], incorporando dos tipos de consultas importantes como: Instante de Tiempo y Trayectoria.

Nuestra propuesta se implementó en lenguaje *C++* y actualmente se encuentra en la etapa de evaluación experimental, en donde se realiza la comparación de la misma contra el método propuesto en [3]. Dicha comparación está basada en el número de accesos a disco y el tiempo que ambas estructuras insumen durante el almacenamiento de la información, como así también en el desempeño de ambas propuestas al momento de la implementación de las consultas Rango, Ventana, Instante de Tiempo y Trayectoria. La principal motivación para llevar a cabo dicha experimentación es reflejar que la propuesta presentada en este artículo conserva las ventajas del método [3] y que las extensiones realizadas no degradan el desempeño del mismo.

Dado que es un área de investigación abierta en donde hay poco trabajo realizado hasta el momento, se pretende continuar con la investigación con el afán de lograr, en caso de ser posible, una optimización en la estructura y la ampliación del espectro de consultas realizables sobre dicha estructura.

PARTICPACIONES

Estos trabajos están enmarcados dentro de la Línea de investigación Geometría Computacional y Bases de Datos Espacio-Temporales, perteneciente al Proyecto Tecnologías Avanzadas de Bases de Datos 22/F314, Departamento de Informática, Universidad Nacional de San Luis; en el Proyecto AL06_PF_013 Geometría Computacional, subvencionado por la Universidad Politécnica de Madrid; y en el marco de la Red Iberoamericana de Tecnologías del Software (RITOS2), subvencionado por CYTED. Por todo ello, se ha establecido un grupo de interés en el tema conformado por docentes investigadores y alumnos avanzados de la Universidad Nacional de San Luis.

REFERENCIAS BIBLIOGRÁFICAS

[1] Abbadi, A., Agrawal, D. y Chon, H. Query processing for moving objects with space-time grid storage model. *In Proc. of the 3rd Intl. Conf. on Mobile Data Management (MDM)*, pages 121-, 2002.

[2] Abbadi, A., Agrawal, D. y Chon, H. Using space-time grid for efficient management of moving objects. *In 2nd ACM Intl. Workshop on Data Engineering for Wireless and Mobile Access (MobiDE)*, pages 59-65, 2001.

[3] Almeida, V. y Güting, R. Indexing the trajectories of moving objects in networks. *Technical Report 309, Fernuniversität Hagen, Fachbereich Informatic,* 2004.

[4] Correa, L. y Ortiz, N. I+MON-Tree: índice espacio-temporal para objetos en movimiento. *Informe Trabajo Final Licenciatura, Univ. Nac. de San Luis, Argentina,* 2006. Gagliardi, O., Directora.

[5] Frentzos, E. Indexing objects moving on fixed networks. *In Proc. of the 8th Intl. Symp. on Spatial and Temporal Databases (SSTD)*, pages 289-305, 2003.

[6] Gunopoulos, G., Hadjieleftheriou, M., Kollios, G. y Tsotras, V. Efficient indexing of spatiotemporal objects. *In Proc. of the 8th Intl. Conf. on Extending Database Technology (EDBT)*, pages 251-268, 2002.

[7] Guttman, A. R-Trees: A dynamic index structure for spatial searching. *In ACM SIGMOD Conference on Management of Data*, pages 47-57, Boston, ACM, 1984.

[8] Hilbert, D. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38:59-60, 1891.

[9] Jensen, C. y Pfoser, D. Capturing the uncertainty of moving objects representations. *In Proc. of Advances Spatial Databases, 6th Intl. Symp. (SSD)*, pages 111-132, 1999.

[10] Jensen, C. y Pfoser, D. Indexing of network constrained moving objects. *In Proc. of the 11th Intl. Symp. on Advances in Geographic Information Systems (ACM-GIS)*, 2003.

[11]Lazaridis, L., Mehrotra, S. y Portkaew, K. Querying mobile objects in spatio-temporal databases. *In Proc. Of the 7th Intl. Symp. on Spatial and Temporal Databases (SSTD)*, pages 59-78, 2001.

[12]Rodriguez, A. *Una* revisión de bases de datos de objetos en movimiento. *Revista de Ingeniería Informática, Edición Nº 11*, 2005.

[13]Roussopoulos, N. y Song, Z. Hashing moving objects. *In Proc. of the 2nd Intl. Conf. on Mobile Data Management (MDM)*, pages 161-172, 2001.

[14]Roussopoulos, N. y Song, Z. SEB-tree: An approach to index continuously moving objects. *In Proc. of the 4th Intl. Conf. on Mobile Data Management (MDM)*, pages 340-344, 2003.