D*R-Tree: un método de acceso espacio-temporal

Edilma O. Gagliardi [(1)], María G. Dorzán [(1)]
Juan G. Gómez Barroso
Departamento de Informática
Facultad de Ciencias Físico, Matemáticas y Naturales
Universidad Nacional de San Luis, Argentina
{oli, mgdorzan, jggomez}@unsl.edu.ar

Gilberto Gutiérrez Retamal
Departamento de Auditoría e Informática
Facultad de Ciencias Empresariales
Universidad del Bío-Bío, Chile
ggutierr@ubiobio.cl

Resumen

Las Bases de Datos Espacio-temporales permiten almacenar y consultar los cambios de posición, forma y/o tamaño de objetos a lo largo del tiempo. Para responder en forma eficiente las consultas que involucran predicados espacio-temporales, es fundamental contar con métodos de acceso que permitan seleccionar los objetos que forman parte de la respuesta, en lugar de recorrer la Base de Datos completa. Se han propuesto diversos métodos en la literatura, sin embargo en ninguno de ellos se pueden resolver eficientemente los cuatro tipos de consulta más conocidos: *TimeSlice*, *Evento*, *Intervalo* y *Trayectoria*. *D*R-Tree* es un método de acceso espacio-temporal que mantiene puntos de referencia para ciertos instantes de tiempo en la base de datos, los objetos en esos instantes de tiempo se almacenan en una estructura de datos para acceso espacial llamada *R-Tree*. Los movimientos ocurridos entre puntos de referencia consecutivos se mantienen en una estructura llamada bitácora. Con esta estructura se logran responder los cuatro tipos de consulta nombrados anteriormente, con un buen desempeño.

Palabras claves: bases de datos espacio-temporales, métodos de acceso espacio-temporal.

1. Introducción

Las investigaciones en torno a los modelos de Bases de Datos Espaciales y Temporales se realizaron, en su mayoría, de manera independiente. Así, los trabajos de investigación en Bases de Datos espaciales se centralizaron en el modelado y la resolución de consultas, basándose en la geometría asociada a los objetos almacenados en una base de datos [6]. Y para el caso de las Bases de Datos temporales, se procuró incluir información acerca del pasado como atributos adicionales de los objetos. Actualmente, las Bases de Datos Espacio-Temporales (*BDET*) en conjunto representan un tópico de estudio reciente y de mucho interés debido a la estrecha relación existente

[(1)] Proyecto Tecnologías Avanzadas de Bases de Datos 22/F314, Departamento de Informática, UNSL; y Proyecto AL05_PF_0042 Geometría Computacional, UPM.
Este trabajo es parcialmente subvencionado por la Red Iberoamericana de Tecnologías del Software (RITOS2), enmarcada en CYTED.

entre espacio y tiempo en una gran cantidad de aplicaciones en donde se debe modelar datos con componentes tanto espaciales como temporales.

Por ello, es necesario que una BDET sea capaz de representar modelos muy cercanos al mundo real, con todo el dinamismo que él implica, y administrar objetos que, básicamente, cambian su ubicación y/o forma a través del tiempo [12]. Se pueden encontrar ejemplos claros en áreas como las de transporte (vigilancia de tráfico), Ciencias Sociales (demografía), telecomunicaciones (telefonía celular), multimedia (películas animadas) e información geográfica (cambios de límites en terrenos), entre otras [27].

Por ejemplo, en un sistema de vigilancia satelital de una empresa de transportes puede ser interesante conocer la ubicación de las unidades a lo largo del tiempo. Para una aplicación como ésta resulta fundamental poder responder consultas con predicados espaciales y temporales, tales como: *¿Dónde se encuentra la unidad nº* 12 *ahora*? haciendo preciso el instante *ahora*; o *¿Qué unidades estaban más cercanas a la intersección de las rutas A y B ayer a las* 14:00 *hrs?* también haciendo precisos los conceptos *más cercanas* y *ayer*. Otras podrían ser *¿Cuál fue el recorrido de la unidad nº* 8 *en la última semana*?, *¿Cuáles unidades se encontraban en la ruta A entre* 04:00 *hrs. y las* 08:00 *hrs. el día* 14/02/05?, *¿Cuáles unidades llegaron y cuáles salieron en la zona Z a las* 20:00 *hrs?* entre otras.

De esta forma, surgen las BDET's, que permiten modelar en forma adecuada los objetos dinámicos del mundo real. Un Sistema de Administración de Bases de Datos Espacio-Temporales (*SABDET*), debe ofrecer los tipos de datos apropiados y un lenguaje de consulta para soportar datos espaciales dinámicos y estáticos. Además, debe proveer métodos eficientes de indexación y recuperación de datos y explotar modelos de costo de operaciones espacio-temporales para procesamiento de consultas y propósitos de optimización [25].

Se requieren métodos eficientes de indexación y recuperación de datos para responder los distintos tipos de consultas que incluyen predicados espaciales y temporales. De esta manera, no es necesario recorrer la Base de Datos completa sino solamente las entradas de la Base de Datos que indica el índice.

En este sentido, se han realizado investigaciones en el campo de los métodos de acceso espacio-temporal y diseñado estructuras auxiliares para soportar consultas espacio-temporales.

Nuestro trabajo propone un nuevo método de indexación, *D*R-Tree*, para poder almacenar y responder eficientemente consultas de tipo espacio-temporal. Está basado en el modelo propuesto en [5], con el fin de aprovechar sus ventajas respecto de una buena utilización de espacio en disco y un buen tiempo para responder consultas.

D*R-Tree mantiene puntos de referencia para ciertos instantes de tiempo en la base de datos, tal que para esos instantes los objetos se almacenan en una estructura de datos *R-Tree*. Las modificaciones ocurridas entre los puntos de referencia de tiempo consecutivos se mantienen en una estructura llamada *bitácora*.

Si bien, el método original, posee la ventaja de equilibrar tiempo de respuesta de consultas con espacio de almacenamiento, no se pueden realizar eficientemente, o no están previstas, consultas de tipo trayectoria. En general, la mayoría de los métodos existentes no apuntan a responder todos los tipos de consultas más comunes, sino que sus propuestas se basan en estructuras que sólo dan soporte a un subconjunto de ellos. Por consiguiente, y reconociendo la importancia de conocer la trayectoria de un objeto determinado, surge nuestra propuesta, que intenta aprovechar las bondades del resto de los métodos, respondiendo eficientemente los principales tipos de consultas e incluyendo la posibilidad de responder consultas de tipo trayectoria.

En la siguiente sección se detallan los distintos métodos existentes para resolver el problema de la indexación espacio-temporal mencionando las estructuras en las que se basa nuestra propuesta. En la sección 3 se presenta el D*R-Tree, incluyendo una descripción de la estructura de datos y de los algoritmos utilizados para actualizar la estructura y para responder los distintos tipos de consulta. Finalmente se describe el trabajo realizado hasta el momento y los futuros avances del mismo.

2. Aspectos teóricos relacionados

A continuación abordamos el problema de indexación de objetos espacio-temporales y se discuten los principales tipos de consultas espacio-temporales y métodos de acceso hasta ahora conocidos.

En primer lugar presentamos los tipos de consultas espacio-temporales de nuestro interés. Estas consultas incorporan predicados espaciales que involucran instantes o intervalos de tiempo. Los principales [5, 27]:

- *TimeSlice*: El resultado de la consulta consiste de los objetos que se encuentran en una determinada área en un instante de tiempo dado.
- *Intervalo*: A diferencia de la consulta *TimeSlice* se tiene en cuenta un intervalo y no un instante de tiempo.
- *Eventos:* Se recuperan eventos que han sucedido en una región en un instante dado. Estos eventos pueden ser objetos que han *aparecido* o *desaparecido* en una región en un cierto instante.
- *Trayectoria*: Se recupera el conjunto de posiciones espaciales en las que un objeto ha permanecido en un intervalo de tiempo dado.

Los métodos de acceso espacio-temporal se pueden clasificar en tres tipos según [10]:

- *Métodos que indexan el pasado*:
  - *Métodos que incorporan aspectos temporales a un método espacial*: En estos métodos la principal preocupación es manejar el dominio espacial, por lo tanto trabajar con consultas temporales es considerado un asunto secundario. Entre estos métodos podemos mencionar: *RT-Tree* [28], *3D R-Tree* [26] y *STR-Tree* [14].
  - *Métodos que utilizan superposición*: En esta segunda categoría se toman el espacio y el tiempo como dimensiones separadas. El objetivo es mantener juntos todos los datos espaciales que están “vivos” en un instante de tiempo en un índice. Podemos nombrar métodos como *MR-Tree* [28], *HR-Tree* [11], *HR+-Tree* [21], *MV3R-Tree* [22] y *PPR-Tree* [8].
  - *Métodos basados en trayectoria*: Este tercer tipo se concentra en consultas orientadas a las trayectorias. El trabajo con consultas espaciales y que los objetos espacialmente cercanos se almacenen juntos son una preocupación secundaria. Se han planteado estructuras como *TB-Tree* [14], *SETI* [3] y *SEB-Tree* [20].
- *Métodos que indexan la posición actual*: En los métodos de acceso espacio-temporales anteriores se asume que los movimientos son conocidos a priori. De esta forma, sólo se

almacenan trayectorias cerradas. No se almacenan ni son requeridas las posiciones actuales de los objetos. Para resolver este problema se han diseñado estructuras tales como *2+3 R-Tree* [13], *2-3 TR-Tree* [1] y *LUR-Tree* [9].

- *Métodos que indexan la posición actual y futura*: Para predecir la posición futura de los objetos, se debe almacenar información extra, por ejemplo velocidad y destino de los objetos. Existen diversos métodos de acceso espacio temporal que permiten hacer consultas de tipo predictivo tales como *PMR-quadtree for moving objects* [24], *SV-Model* [4], *PSI* [15], *TPR-Tree* [19], *PR-Tree* [2], *NSI* [15], *VCI R-Tree* [16], *STAR-Tree* [17], *Rexp-Tree* [18] y *TPR*-Tree* [23].

A continuación realizamos una presentación de las estructuras en las cuales se basa nuestra propuesta, con el fin de mostrar su diseño.

2.1 R-Tree

R-Tree [7] se ha adoptado como el método de acceso estándar para las bases de datos espaciales y el elegido por la mayoría de los Sistemas de Administración de Bases de Datos. Es el más estudiado con respecto a tópicos tales como procesamiento de consultas, optimización de consultas, modelos de costo, paralelismo, control de concurrencia y recuperación. Además, gran parte de los métodos de acceso espacio-temporal propuestos hoy en día usan como base a R-Tree.

R-Tree es un árbol balanceado por altura, basado en el B-Tree. En un R-Tree no se almacenan los objetos espaciales en forma directa sino que se almacena su *MBR* (*Minimum Bounding Rectangle*), es decir el menor rectángulo que contiene al objeto en cuestión. Cada nodo en el R-Tree corresponde al MBR que contiene a sus hijos. Los nodos hoja de R-Tree contienen punteros a los objetos en la base de datos en vez de punteros a otros nodos. Cada nodo se almacena en una página de disco.

Los nodos hoja de R-Tree contienen entradas de la forma < *Mbr*; *O* > donde *Mbr* es el menor rectángulo que contiene al objeto apuntado por *O*. Los nodos no hoja contienen entradas de la forma < *Mbr*; *P* > donde *P* es un puntero a un hijo del nodo y *Mbr* contiene a todos los MBR's del nodo apuntado por *P*.

En un R-Tree, cada nodo, con la posible excepción del nodo raíz, contiene entre *m* y *M* entradas donde $m <= M/2$ y *M* es el número máximo de entradas por nodo; el nodo raíz tiene al menos dos hijos a menos que sea una hoja; y todas las hojas están al mismo nivel.

En el procedimiento de búsqueda se desciende por el árbol a partir de la raíz; siguiendo por los hijos cuyo MBR se intersecta con el área de consulta y así en forma recursiva, hasta llegar a las hojas. Los MBR's que encierran los diferentes nodos pueden superponerse; además, un MBR puede estar incluido, en el sentido geométrico, en varios nodos, pero está asociado sólo con uno de ellos. Esto implica que una búsqueda puede seguir más de un camino, incluyendo caminos innecesarios.

Para insertar un objeto se desciende recursivamente por el árbol a partir de la raíz; siguiendo por los hijos cuyo MBR crecerá menos, producto de la inserción de un nuevo objeto hasta llegar a un nodo hoja. El objeto se inserta en la hoja si hay espacio, en caso contrario el nodo se divide usando alguna de las técnicas de división conocidas [7]. Posteriores variaciones de R-Tree difieren principalmente en la forma en que se insertan los objetos.

Al eliminar un objeto si el nodo que lo contenía tiene insuficientes entradas estas se eliminan y se reinsertan. Los cambios de MBR, producto de la eliminación, se propagan hacia arriba.

2.2 DR-Tree

El *DR-Tree* es un método de acceso espacio-temporal de tipo histórico [5], que trata de mantener un equilibrio entre el espacio de disco utilizado por la estructura y el tiempo de acceso empleado en responder los distintos tipos de consulta.

La estructura tiene puntos de referencia para ciertos instantes de tiempo, tal que los objetos "vivos" en esos instantes se almacenan en una estructura de datos R-Tree. Las modificaciones producto del movimiento de los objetos entre puntos de referencia temporal consecutivos se mantienen en una lista de bloques denominada bitácora. Ésta se encuentra ordenada de acuerdo al tiempo y permite reconstruir cualquier estado de la base de datos entre dos puntos de referencia consecutivos.

La bitácora es una lista de bloques. Las entradas en los bloques son tuplas con la siguiente estructura: < *t*; *Mbr anterior*; *Mbr actual*; *Oid* > donde *t:* Tiempo en que se produjo la modificación; *Mbr anterior*: Aproximación al objeto en el tiempo *t-1*; *Mbr actual*: Aproximación al objeto en el tiempo *t*; *Oid*: Identificador del objeto. Las entradas en las la bitácora se encuentran ordenadas por el atributo *t*.

Se proporciona un parámetro *d* correspondiente a un valor de umbral del tamaño de las bitácoras, así la estructura podrá decidir en qué momento debe crearse un nuevo punto de referencia. Éste se crea cada vez que en la bitácora actual sea imposible almacenar los cambios producidos en un instante *t*, ya que al hacerlo el número de bloques de ésta superaría el valor *d*.

3. D*R-Tree

D*R-Tree es un método de acceso espacio-temporal de tipo histórico que se basa en la idea expuesta en [5]. Tanto el método original, como el propuesto en este artículo, pretenden mantener un equilibrio entre el espacio de disco utilizado por la estructura y el tiempo de acceso empleado en responder los distintos tipos de consulta.

Nuestra propuesta consiste de las siguientes características:

- Se utiliza un Índice para almacenar los instantes de tiempo que se establezcan como puntos de referencia.
- Los R-Tree's son utilizados para almacenar la ubicación espacial de los objetos. Cada R-Tree está asociado a un punto de referencia de tiempo, según el índice descrito en el punto anterior.
- Se usan Bitácoras para almacenar los movimientos realizados en instantes intermedios entre puntos de referencia almacenados en el índice de instantes; además, se guardan las referencias a los movimientos anteriores inmediatos por cada objeto.
- Se agrega un Índice para acceder a los últimos registros de movimiento en cada bitácora por cada objeto. Con ello se posibilita la entrada directa a las bitácoras involucradas en la trayectoria de los objetos.

Para realizar la actualización de la estructura suponemos que disponemos de una "imagen" de los objetos en el espacio. Se creará un nuevo punto de referencia cuando, al querer ingresar un nuevo conjunto de movimientos en la bitácora, ésta no dispone de espacio suficiente para albergarlos, dando origen a una nueva entrada en el Índice de instantes y un nuevo R-Tree, el que se construye de acuerdo a las diferencias entre el R-Tree y la bitácora del último punto de referencia almacenado. Estas diferencias se refieren a los cambios que se deben hacer en la imagen inmediatamente anterior para alcanzar la imagen actual. En la figura 1 se presenta un esquema general de nuestra propuesta

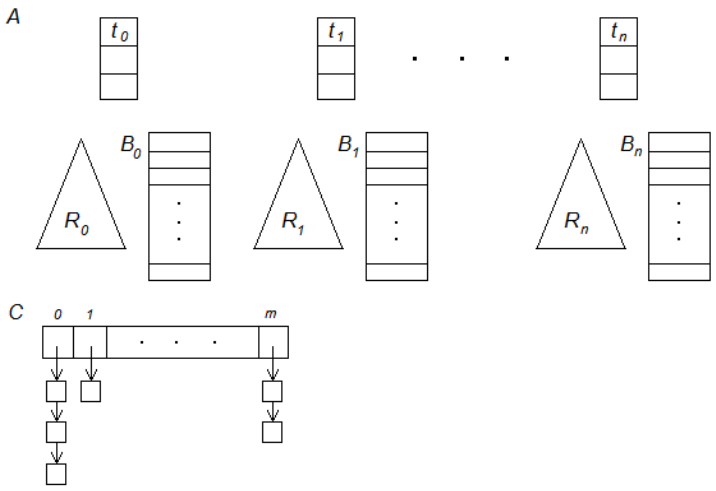

Figura 1: Esquema general de D*R-tree

Descripción de la estructura:

- *Índice de tiempos* (*A*): es una lista secuencial que almacena tuplas de la forma <*t*, *R*, *B*> donde:

  t: tiempo del punto de referencia;

  R: mantiene la raíz del R-Tree asociado a *t*;

  B: mantiene la cabecera de la bitácora asociada a *t*.

- *R-Tree* (*R*): descripto anteriormente. Este mantiene la misma estructura que en la versión original.

- *Bitácora* (*B*): es una lista secuencial ordenada por el tiempo que almacena tuplas de la forma <*Oid, t, Ref_ant, Mbr_act*> donde:

  Oid: es el identificador del objeto;

  t: es el tiempo en que se produjo el movimiento;

  Ref_ant: mantiene una referencia a la ubicación del movimiento anterior del objeto en la bitácora correspondiente;

  Mbr_act: mantiene la posición actual del objeto por medio de un par de coordenadas (x,y), donde x e y son puntos que corresponden a la diagonal del MBR.

- *Índice de Trayectorias* (*C*): es una lista de dos niveles donde:

  1º nivel: se mantienen todos los objetos que son accedidos por *Oid* y se mantiene una referencia a la cabecera de la lista de segundo nivel.

  2º nivel: es una lista vinculada. Cada nodo hace referencia a una bitácora distinta y almacena información correspondiente a la última entrada en dicha bitácora para el objeto correspondiente al índice de la lista de primer nivel. Cada bitácora tendrá, a lo sumo, una única entrada en la lista.

Así, las principales diferencias con respecto al método original, es que el D*R-Tree posee una estructura adicional, un *Índice de Trayectorias*, que nos brinda una ventaja, permitiéndonos guardar

información para poder recuperar la trayectoria de los objetos en forma eficiente. Esto se complementa con la reestructuración de la información almacenada en las bitácoras.

3.1 Algoritmos de consultas

A continuación explicamos los distintos tipos de consultas soportados por el D*R-Tree con sus respectivos algoritmos donde se tienen en cuenta las siguientes restricciones: la cantidad de objetos en movimiento es fija; el porcentaje de movilidad es alto; los objetos informan los cambios de posición; el universo donde se mueven los objetos es acotado y conocido; se utilizarán distribuciones de probabilidades para generar los movimientos de los objetos y las consultas forman con rectángulos cuyos lados son perpendiculares a los ejes.

3.1.1 Consulta TimeSlice

Para procesar una consulta de este tipo primero se debe ubicar el punto de referencia adecuado, de acuerdo al instante de tiempo especificado en la consulta (t). Luego se hace una búsqueda espacial en el R-Tree del punto de referencia encontrado. El conjunto de objetos obtenidos por esta consulta es actualizado con las entradas de la correspondiente bitácora.

Algoritmo *TimeSlice*(R, t)

Entrada: R es el rectángulo de consulta y t es el instante sobre el cual se realiza la consulta.

Salida: Q es el conjunto de objetos que responden a la consulta.

1. $i \leftarrow$ *BuscarTiempo*(t)
2. $Q \leftarrow$ *BuscarEnRTree*($A(i).R$, R)
3. Mientras $A(i).t \leq t$ hacer
4. $Q \leftarrow$ *ActualizarQconBitácora*($A(i).B$, R, t)
5. finmientras
6. Retornar Q

Donde:

BusarTiempo(t) permite encontrar, mediante una búsqueda binaria, el punto de referencia que corresponde a la posición en A tal que A(i).t ≤ t.

BuscarEnRTree($A(i).R$, R) permite encontrar en el R-Tree, con raíz $A(i).R$, los objetos que intersectan a R. La explicación de este algoritmo se encuentra en [7].

ActualizarQconBitácora($A(i).B$, R, t) actualiza al conjunto resultado Q buscando en la bitácora $A(i).B$ aquellos objetos referenciados en ésta, cuyos tiempos sean menores o iguales a t, para determinar cuáles continúan intersectando a R y cuáles no.

3.1.2 Consulta Intervalo

El procesamiento de este tipo de consulta es muy simple. Primero se establece el estado de los objetos en el límite inferior del intervalo (t_o). Para lograr esto se usa el método propuesto para *TimeSlice*. Una vez establecido este conjunto se recorren todas las entradas de las bitácoras cuyo atributo t es menor o igual al límite superior (t_f) especificado en el intervalo, actualizando el conjunto.

Algoritmo *Intervalo*(R, t_o, t_f)

Entrada: R es el rectángulo de consulta, t_o es el límite inferior del intervalo de tiempo y t_f es el límite superior sobre el cual se realiza la consulta.

Salida: Q es el conjunto de objetos que responden a la consulta.

1. $i \leftarrow$ *BuscarTiempo*(t)
2. $Q \leftarrow$ *BuscarEnRTree*($A(i).R$, R)
3. Mientras $A(i).t \leq t_o$ hacer
4. $Q \leftarrow$ *ActualizarQconBitácora*($A(i).B$, R, t_o)
5. finmientras
6. Si Todas las entradas de la bitácora $A(i).B$ fueron procesadas entonces
7. $i \leftarrow i + 1$
8. finsi
9. Mientras $A(i).t \leq t_f$ hacer
10. $Q \leftarrow$ *ActualizarQconBitácora* ($A(i).B$, R, t_f)
11. $i \leftarrow i + 1$
12. finmientras
13. Retornar Q

Donde:

BusarTiempo(t) permite encontrar, mediante una búsqueda binaria, el punto de referencia que corresponde a la posición en A tal que $A(i).t \leq t$.

ActualizarQconBitácora($A(i).B$, R, t) actualiza al conjunto resultado Q buscando en la bitácora $A(i).B$ aquellos objetos referenciados en ésta, cuyos tiempos sean menores o iguales a t, para determinar cuáles continúan intersectando a R y cuáles no.

3.1.3 Consulta Eventos

Para responder este tipo de consultas primero se debe ubicar la bitácora donde están los cambios del tiempo consultado. Se recorren todas las entradas con tiempo igual a t para indicar cuántos objetos entraron o salieron del área de consulta.

Algoritmo *Eventos*(R, t)

Entrada: R es el rectángulo de consulta y t es el instante de tiempo sobre el cual se realiza la consulta.

Salida: Q es el conjunto de objetos que responden a la consulta.

1. $i \leftarrow$ *BuscarTiempo*(t)
2. $E \leftarrow$ *BuscarEnBitácoraPorT*($A(i).B$, t)
3. Para cada $e \in E$ hacer
4. Si $e.Mbr\_act$ está en R entonces

5. $Q \leftarrow Q \cup \{e\}$ /* entró o se mantuvo en R*/
6. sino
7. Si *e.Ref_ant* está en *R* entonces
8. $Q \leftarrow Q \cup \{e\}$ /* salió de *R* */
9. finsi
10. finsi
11. finpara
12. Retornar *Q*

Donde:

BusarTiempo(*t*) permite encontrar, mediante una búsqueda binaria, el punto de referencia que corresponde a la posición en *A* tal que $A(i).t \leq t$.

BuscarEnBitácoraPorT(*A*(*i*).*B*, *t*) realiza una búsqueda de todos los objetos referenciados en la bitácora *A(i).B* tal que su tiempo sea *t* .

3.1.4 Consulta Trayectoria

Para procesar una consulta de tipo Trayectoria primero se busca el objeto en la estructura Índice de Trayectoria y se recorre la lista secuencialmente considerando el límite superior del intervalo de consulta (t_k), es decir que se detiene en el nodo de la lista de segundo nivel donde el tiempo asociado a la bitácora apuntada sea el menor más cercano a t_k. Luego se comienzan a recorrer las referencias dentro de las bitácoras obteniendo las posiciones correspondientes, teniendo en cuenta el intervalo de entrada.

Algoritmo *Trayectoria*(*Oid*, t_i, t_k)

Entrada: *Oid* es el identificador del objeto a consultar, t_i es el límite inferior del intervalo de tiempo y t_k es el límite superior sobre el cual se realiza la consulta.

Salida: *Q* es el conjunto de objetos que responden a la consulta.

1. $c \leftarrow C(Oid).siguiente$
2. Mientras $t_k \leq c.t$ hacer
3. $c \leftarrow c.siguiente$
4. finmientras
5. $b \leftarrow A(c.t).B(c.i)$
6. $Q \leftarrow Q \cup \{b.MBR\_act\}$
7. $b \leftarrow b.Ref\_ant$
8. Mientras $(b.t > t_i) \wedge (b \neq -1)$ hacer /*si b = -1, si referencia al primer R-Tree*/
9. $Q \leftarrow Q \cup \{b.MBR\_act\}$
10. $b \leftarrow b.Ref\_ant$
11. finmientras

12. Si *b* = -1 entonces

13. $Q \leftarrow$ *BuscarPosiciónInicial*(*Oid*)

14. finsi

15. Retornar *Q*

Donde:

BuscarPosiciónInicial(*Oid*) busca la posición inicial del objeto con oid *Oid*.

4. Trabajo futuro

Esta es un área de investigación abierta sobre la que existe poco trabajo realizado y es de utilidad dado que pocos sistemas tienen totalmente integrado el tipo dato espacio-temporal y más aún la incorporación de plataformas espacio-temporales para la resolución de consultas que involucran operaciones con predicados espacio-temporales.

El método propuesto se ha implementado en lenguaje *Java* y las pruebas de experimentación se están comparando contra el método de acceso DR-Tree, teniendo en cuenta los aspectos de utilización de almacenamiento en disco y tiempo de acceso en las consultas descriptas anteriormente.

Luego, se realizará la implementación de una aplicación como herramienta didáctica, que permita visualizar el desplazamiento de los objetos y el resultado de las consultas, como por ejemplo la recuperación de una trayectoria, los eventos ocurridos en un instante, entre otras.

Este trabajo está enmarcado dentro de la línea de investigación Geometría Computacional y Bases de Datos, perteneciente al Proyecto Tecnologías Avanzadas de Bases de Datos 22/F314, Departamento de Informática, UNSL y en el marco de la Red Iberoamericana de Tecnologías del Software (RITOS2), subvencionado por CYTED, por lo que se ha establecido un grupo de interés en el tema conformado por docentes investigadores y alumnos avanzados de la UNSL y de la UBB.

Referencias bibliográficas

[1]. Abdelguerfi M., Givaudan J., Shaw K., and Ladner R.. The 2-3 TR-tree, A Trajectory-Oriented Index Structure for Fully Evolving Valid-time Spatio-temporal Datasets. In Proc. of the ACM workshop on Adv. in Geographic Info. Sys., ACM GIS, pages 29–34, (2002)

[2]. Cai M. and Revesz P.. Parametric R-Tree: An Index Structure for Moving Objects. In Proc. of the Intl. Conf. on Management of Data, COMAD, (2000)

[3]. Chakka V., Everspaugh A., and Patel J.. Indexing Large Trajectory Data Sets with SETI. In Proc. of the Conf. on Innovative Data Systems Research, CIDR, Asilomar, CA, (2003)

[4]. Chon H., Agrawal D., and Abbadi A.. Storage and Retrieval of Moving Objects. In Mobile Data Management, pages 173–184, (2001)

[5]. Gutiérrez Gilberto. Propuesta de un método de acceso espacio-temporal. II WorkShop de Bases de datos, Chillan/Chile (2003).

[6]. Güting, R.H., An introduction to Spacial Database System. VLDB Journal (1994)

[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]. Kollios G., Tsotras V., Gunopulos D., Delis A., and Hadjieleftheriou M.. Indexing Animated Objects Using Spatiotemporal Access Methods. IEEE Trans. on Knowledge and Data Engineering, TKDE, 13(5):758–777, (2001)

[9]. Kwon D., Lee S., and Lee S.. Indexing the Current Positions of Moving Objects Using the Lazy Update R-tree. In Mobile Data Management, MDM, pages 113–120, (2002)

[10].Mokbel M., Ghanem T., Aref W. Spatio-temporal Access Methods, IEEE Data Engineering Bulletin 26, pp. 40-49. (2003)

[11].Nascimento M. and Silva J.. Towards historical R-trees. In Proc. of the ACM Symp. on Applied Computing,SAC, pages 235–240, (1998)

[12].Nascimento M., Silva J., and Theodoridis Y.. Access structures for moving points, (1998)

[13].Nascimento M., Silva J., and Theodoridis Y.. Evaluation of Access Structures for Discretely Moving Points. In Proc. of the Intl. Workshop on Spatio-Temporal Database Management, STDBM, pages 171–188, (1999)

[14].Pfoser D., Jensen C., and Theodoridis Y. Novel Approaches in Query Processing for Moving Object Trajectories. In Proc. of the Intl. Conf. on Very Large Data Bases, VLDB, pages 395–406, (2000)

[15].Porkaew K., Lazaridis I., and Mehrotra S.. Querying Mobile Objects in Spatio-Temporal Databases. In Proc. of the Intl. Symp. on Advances in Spatial and Temporal Databases, SSTD, pages 59–78, Redondo Beach, CA, (2001)

[16].Prabhakar S., Xia Y., Kalashnikov D., Aref W., and Hambrusch S.. Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects. IEEE Transactions on Computers, 51(10):1124–1140, (2002)

[17].Procopiuc C., Agarwal P., and Har-Peled S.. STAR-Tree: An Efficient Self-Adjusting Index for Moving Objects. In Proc. of the Workshop on Alg. Eng. and Experimentation, ALENEX, pages 178–193, (2002)

[18].Saltenis S. and Jensen C.. Indexing of Moving Objects for Location-Based Services. In Proc. of the Intl. Conf. on Data Engineering, ICDE, (2002)

[19].Saltenis S., Jensen C., Leutenegger S., and Lopez M.. Indexing the Positions of Continuously Moving Objects. In Proc. of the ACM Intl. Conf. on Management of Data, SIGMOD, pages 331–342, (2000)

[20].Song Z. and Roussopoulos N.. SEB-tree: An Approach to Index Continuously Moving Objects. In Mobile Data Management, MDM, pages 340–344, (2003)

[21].Tao Y. and Papadias D. Efficient Historical R-trees. In Proc. of the Intl. Conf. on Scientific and Statistical Database Management, SSDBM, pages 223–232, (2001)

[22].Tao Y. and Papadias D. MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries. In Proc. of the Intl. Conf. on Very Large Data Bases, VLDB, pages 431–440, (2001)

[23].Tao Y., Papadias D., and Sun J.. The TPR*-Tree: An Optimized Spatio-temporal Access Method for Predictive Queries. In Proc. of the Intl. Conf. on Very Large Data Bases, VLDB, (2003)

[24].Tayeb J., Ulusoy Ö., and Wolfson O.. A Quadtree-Based Dynamic Attribute Indexing Method. The Computer Journal, 41(3):185–200, (1998)

[25].Theodoridis Y., Sellis Timos K., Papadopoulos A., and Manolopoulos Y.. Specifications for

efficient indexing in spatiotemporal databases. In Proceedings 10th International Conference on Scientific and Statistical Database Management, Capri, Italy, pages 123-132. (1998)

[26]. Theodoridis Y., Vazirgiannis M., and Sellis T. Spatio-Temporal Indexing for Large Multimedia Applications. In Proc. of the IEEE Conference on Multimedia Computing and Systems, ICMCS, (1996)

[27]. Wang, X., Zhou, X. y Lu, S. Spatiotemporal data modeling and management: a survey. Proceedings 36th International Conference on Technology of Object-Oriented Languages and System, pp. 202-211. (2000)

[28]. Xu X., Han J., and Lu W. RT-Tree: An Improved R-Tree Indexing Structure for Temporal Spatial Databases. In Proc. of the Intl. Symp. on Spatial Data Handling, SDH, pages 1040–1049, (1990)