D*R-Tree: un método eficiente para responder consultas espacio-temporales

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

y

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

Abstract

Spatio-temporal databases deal with objects that change their location and/or shape over time. Numerous researches have been done in developing spatio-temporal access methods as auxiliary structures to support spatio-temporal queries. The main interest of most applications is to efficiently store and query the positions of these objects. We can find a related rich literature on the subject about the methods for supporting a subset of the following *TimeSlice, Events, Interval and Trajectory* queries. In this paper we propose a new index structure, the *D*R-Tree* to efficiently store and retrieve spatio-temporal objects. The main objective of this work is to show a suitable method for supporting all mentioned queries types, with an optimal performance. We propose the index in terms of the basic algorithms for querying. We test our proposal in an extense experimental evaluation with generated data sets. In our tests, the D*R-Tree showed good scalability when increasing the number of objects and time units in the data sets, as well as in query processing, compared with a similar structure.

Keywords: spatio-temporal access methods; spatio-temporal databases, data structures, algorithms.

Resumen

Existen aplicaciones que requieren manejar objetos espacio-temporales, es decir, objetos cuya posición espacial o forma cambia en distintos instantes de tiempo. Para administrar la información referida a tales cambios se requieren de métodos de acceso espacio-temporal, que permitan además procesar en forma eficiente consultas de tipo espacio-temporal. En general, TimeSlice, Intervalo, Eventos y k son los tipos de consultas para las que podemos encontrar una variedad de métodos, los cuales intentan optimizar el desempeño de las consultas, pero por separado, apuntando a un subconjunto de las antes mencionadas. En este artículo presentamos un método de acceso espacio-temporal, llamado D*R-Tree, que resuelve eficientemente los cuatro tipos de consultas mencionados anteriormente, y mostramos su buen desempeño a través de las evaluaciones experimentales realizadas.

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

[(1)] Proyecto Tecnologías Avanzadas de Bases de Datos 22/F314, Departamento de Informática, UNSL; y Proyecto *AL2006_PF_013* Geometría Computacional, UPM.
Este trabajo es parcialmente subvencionado por la Red Iberoamericana de Tecnologías del Software (RITOS2), enmarcada en CYTED.

1 INTRODUCCIÓN

Existen aplicaciones que requieren manejar objetos espacio-temporales, es decir, objetos cuya posición espacial y/o forma cambia en el tiempo. Para ello, los Sistemas de Administración de Bases Datos (SABDs) deben proveer métodos de acceso para procesar en forma eficiente distintos tipos consultas que tengan en cuenta relaciones espacio-temporales.

En un principio, los avances acerca de los modelos de Bases de Datos Espaciales y Temporales se realizaron de manera separada. Por un lado, 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]. Mientras que, por otro lado, las Bases de Datos Temporales, procuraron incluir información acerca del pasado como atributos adicionales de los objetos. Actualmente, las Bases de Datos Espacio-Temporales (BDETs) constituyen una nueva área de estudio y de interés, debido a la cantidad de aplicaciones en donde se deben modelar datos con componentes espaciales y temporales. Por este motivo, los SABDs deben dar soporte a tipos de datos espacio-temporales, que propicien la manipulación de los mismos y faciliten la resolución de consultas espacio-temporales.

Observamos a lo largo de nuestra investigación en la temática que la mayoría de los Métodos de Acceso Espacio-Temporales (MAETs) existentes no están diseñados para responder los tipos de consultas espacio-temporales más comunes, sino que sus propuestas se basan en estructuras que sólo dan soporte eficiente a un subconjunto de ellos. Estos métodos se pueden clasificar en tres tipos según [9]:

• *Métodos que indexan el pasado*, entre ellos encontramos los *Métodos que incorporan aspectos temporales a un método espacial* como es el caso de *RT-Tree* [18]*, 3D R-Tree* [17] y *STR-Tree* [12]; *Métodos que utilizan superposición* donde podemos nombrar métodos como *MR-Tree* [18] y *HR-Tree* [10] y los *Métodos basados en trayectoria* con estructuras como *TB-Tree* [12] y *SETI* [3].

• *Métodos que indexan la posición actual*: en esta clasificación se han diseñado estructuras tales como *2+3 R-Tree* [11] y *2-3 TR-Tree* [1].

• *Métodos que indexan la posición actual y futura* como *PSI* [13]*, NSI* [13]*, VCI R-Tree* [14]*, STAR-Tree* [15] y *Rexp-Tree* [16].

Por lo expuesto anteriormente, nuestro trabajo consistió en desarrollar un método de indexación espacio-temporal, con el objetivo de evaluar y diseñar estructuras de datos y algoritmos que permitan resolver eficientemente los principales tipos de consultas espacio-temporales. Así, nos propusimos diseñar un método de acceso espacio-temporal, basado en uno existente, de modo tal que pudiésemos ampliar el conjunto de consultas que éste soportase, y optimizar aspectos relacionados al uso del espacio y que, en consecuencia, trajeran aparejado mejoras en los tiempos de respuestas.

Por lo tanto, y reconociendo la importancia de mantener un índice que respondiera eficientemente diversos tipos de consultas comúnmente requeridos, nuestra propuesta se basó en aprovechar las ventajas de un método con comprobada eficiencia, respondiendo los tipos de consultas *TimeSlice*, *Intervalo* y *Eventos*, e incluyendo la posibilidad de responder consultas de tipo *Trayectoria*.

En este artículo presentamos el índice D*R-Tree [4], que almacena datos espacio-temporales y responde eficientemente tipos de consultas antes mencionados sobre éstos. Para su diseño y desarrollo, nos basamos en el modelo propuesto en [5], aprovechando sus ventajas respecto de una buena utilización de espacio en disco, y un tiempo estimativamente razonable en las aplicaciones para responder las consultas de tipo TimeSlice, Intervalo y Eventos.

Por tanto, se realizó una optimización de las estructuras subyacentes y se extendió el conjunto de

consultas resueltas por el método base, pudiendo responder consultas de tipo trayectoria de un objeto. Esta extensión trajo aparejado la creación de nuevas estructuras para poder mantener la información necesaria, de forma tal que la recuperación de los datos sea eficiente.

Cabe aclarar que debido a los inconvenientes que la consulta de tipo Trayectoria ocasiona, referente a espacio y tiempo de respuesta, existen MAETs diseñados especialmente para responderla en forma eficiente.

En cuanto a las evaluaciones experimentales realizadas, se comparó con [5] y se comprobó la eficiencia del método. Éstas brindaron resultados satisfactorios, dado que obtuvimos un desempeño equiparable y, fundamentalmente, ampliamos el conjunto de consultas espacio-temporales original agregando un nuevo tipo, y tratamos todas las consultar dentro de una misma estructura.

Además, como objetivos subyacentes a esta investigación, se pretendió consolidar y alimentar una línea de estudio, a fines de brindar un puente a futuros trabajos de investigación.

El presente artículo está organizado de la siguiente manera: en la sección 2 exponemos nuestro método de acceso espacio-temporal, D*R-Tree, con la descripción de las estructuras que lo componen. A continuación, en la sección 3, presentamos las consultas espacio-temporales que trata D*R-Tree y cómo las resuelve. En la sección 4 presentamos los resultados de la evaluación experimental de nuestro índice comparándolo con uno de comprobada eficiencia. Por último, mostramos las conclusiones y la visión a futuro de nuestro trabajo.

2 D*R-TREE

El MAET que presentamos en este trabajo, denominado D*R-Tree, se basa en la idea expuesta en [5], denominado SEST-Index, e intenta mantener un equilibrio entre el espacio utilizado para mantener la estructura y el tiempo de acceso empleado en responder los distintos tipos de consulta.

SEST-Index es un método de acceso espacio-temporal de tipo histórico que mantiene puntos de referencia con el componente espacial de los objetos para ciertos instantes de tiempo. Los objetos que se mueven en esos instantes de tiempo se almacenan en una estructura de datos R-Tree. Los movimientos de los objetos entre puntos de referencia consecutivos se mantienen en una lista de movimientos denominada bitácora, ordenada de acuerdo al tiempo, y que permite reconstruir cualquier estado de la base de datos entre dos puntos de referencia consecutivos. Las entradas de las bitácoras son tuplas que almacenan información referida a: i) el tiempo en que se produjo el moviemiento, ii) el identicador del objeto que realizó el movimiento y iii) la aproximación del objeto en el instante de tiempo t-1 y en el instante t. Este método está diseñado para responder eficientemente las consultas de tipo TimeSlice, Eventos e Intervalo.

Teniendo en cuenta los aspectos generales del funcionamiento del método en el que nos basamos, D*R-Tree mantiene puntos de referencia para ciertos instantes de tiempo donde se almacena la información espacial de los objetos en un R-Tree [2, 7, 8]. Todos los cambios ocurridos entre dos puntos de referencia consecutivos se almacenan en una lista llamada bitácora, la cual se encuentra ordenada de acuerdo al tiempo. De esta manera, este nuevo índice, evita examinar todos los objetos de la base de datos al momento de responder consultas cuyos predicados consideran restricciones espacio-temporales.

2.1 Descripción de la estructura

En forma general, D*R-tree se compone de las siguientes estructuras, las cuales permiten mantener la información necesaria acerca de los objetos y, de esta manera, recuperar el estado de los objetos

para cualquier instante de tiempo:

- *Índice de puntos de referencias*: Se emplea para almacenar los instantes de tiempo que se establezcan como puntos de referencia. Para cada uno de éstos, se genera un R-Tree y se dispone el estado inicial de la bitácora asociada.
- *R-Tree*: Se genera uno para cada punto de referencia en el tiempo, según el índice descripto en el ítem anterior y se utiliza para almacenar la ubicación espacial de los objetos. Por cada objeto se almacena el *Minimum Bounding Rectangle* (MBR), o menor rectángulo que lo contiene; es decir que no se mantiene la representación espacial completa de los objetos. Cada nodo interno corresponde al MBR que contiene a sus hijos. Los nodos hoja contienen referencias a los objetos en sí.
- *Bitácora*: Se emplean para almacenar los movimientos de los objetos en instantes intermedios a puntos de referencia consecutivos; además, se guardan las referencias espacio-temporales de los movimientos inmediatos anteriores correspondientes a cada objeto.
- *Índice de Trayectorias*: Mantiene una referencia al último movimiento almacenado en cada bitácora correspondiente a un objeto. Con ello posibilitamos el acceso directo a las bitácoras involucradas en la trayectoria de los objetos.
- *Índice de Tiempos*: Se utiliza un índice que permite encontrar en las distintas bitácoras el primer movimiento correspondiente a un instante de tiempo.

En la Figura 1 se muestra el esquema general del índice.

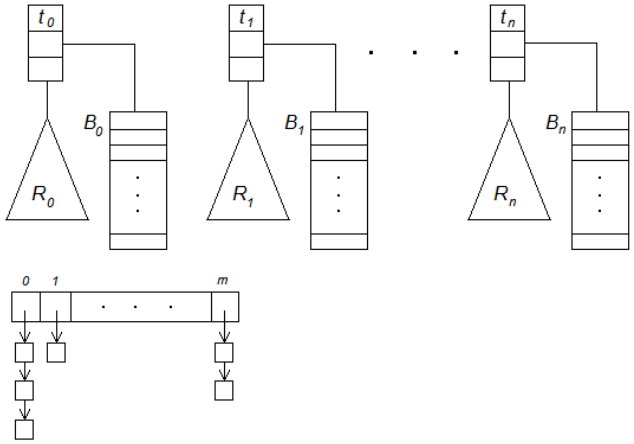

Figura 1 – Esquema general de D*R-tree

2.2 Actualización de la estructura

El registro de los movimientos de los objetos es discreto. Es decir, en un instante de tiempo, los objetos que se mueven, informan un cambio de posición. Esto produce una actualización en la estructura, que debe incorporar la información necesaria de manera tal que refleje el movimiento de los objetos. Considerando un punto de referencia p, un instante de tiempo t y el conjunto m_t de movimientos producidos en dicho instante, puede ocurrir uno de los siguientes casos, al momento de ingresarlos en el índice:

- Caso 1: existen suficientes posiciones libres en la bitácora b_p asociada al punto de referencia actual p para poder albergar los movimientos correspondientes al instante t. Por lo tanto, cada movimiento del conjunto m_t se agrega al final de la bitácora b_p.

- Caso 2: en la bitácora b_p asociada al punto de referencia actual p no existe la cantidad necesaria de posiciones libres para poder almacenar los movimientos del conjunto m_t. En este caso, se crea un nuevo punto de referencia p_t para el instante t, dando origen a una nueva entrada en el *Índice de puntos de referencias*. Esto lleva a la creación una nueva bitácora b_t y un nuevo R-Tree r_t que se asocian a p_t. Este R-Tree se construye teniendo en cuenta el R-Tree r y la bitácora b_p correspondientes al punto de referencia anterior a p_t. Es decir, se actualizan las posiciones de los objetos almacenados en r con la información de los movimientos contenida en la bitácora b_p. Luego, cada movimiento del conjunto m_t se agrega al final de la bitácora b_t.

En ambos casos, cuando se agrega el registro de un movimiento de un objeto o en la bitácora, se asigna la correspondiente referencia a la entrada donde se mantiene el movimiento inmediato anterior de o. Además, se debe actualizar la referencia correspondiente en el *Índice de Trayectorias* para el objeto o y agregar la correspondiente entrada en el *Índice de Tiempos*.

A continuación se expondrán las consultas que soporta D*R-Tree, cuyas ideas principales fueron publicadas en [4].

3 CONSULTAS EN D*R-TREE

Un MAET debe proveer la capacidad de poder responder eficientemente consultas que involucran predicados espacio-temporales, reflejando lo que es de interés para el usuario. El método de acceso determina de qué manera tratar y evaluar la consulta. Al momento de utilizar un índice, es importante determinar qué tipo de consultas se tendrán en cuenta ya que esto establece la forma en que se administra y almacena la información, como así también la manera de recuperar la información necesaria. Los tipos de consultas espacio-temporales considerados en este trabajo y los respondidos por D*R-Tree son los siguientes:

Consulta TimeSlice: consiste en encontrar los objetos que se ubican en una determinada área en un instante de tiempo dado.

Consulta Intervalo: consiste en encontrar los objetos que se ubiquen en una determinada área en un intervalo de tiempo dado.

Consulta Eventos: en éstas se recuperan eventos que sucedieron en una región en un instante dado. Estos eventos pueden ser objetos que han realizado una acción determinada en un cierto instante, como por ejemplo aparecer o desaparecer en una región.

Consulta Trayectoria: son las que recuperan el conjunto de posiciones espaciales por las que un objeto en particular se ha movido en un intervalo de tiempo dado.

A continuación explicaremos el modo en que nuestro índice trabaja para responder las consultas de este tipo.

- *Consulta TimeSlice*: Para procesar una consulta de este tipo primero se debe ubicar el punto de referencia adecuado en el *Índice de puntos de Referencias*, 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.

- *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.

• *Consulta Eventos*: Para responder este tipo de consultas primero se debe ubicar la bitácora donde están los movimientos del tiempo consultado (t), esto se consigue consultando en el *Índice de Tiempos*. Se recorren todas las entradas con tiempo igual a t para indicar cuántos objetos entraron o salieron del área de consulta.

• *Consulta Trayectoria*: Primero se debe acceder en forma directa al *Índice de Trayectorias* utilizando el identificador del objeto que se quiere consultar ya que éste es un direccionamiento directo. Una vez localizada la entrada correspondiente, se recorre la lista vinculada asociada a esta entrada considerando el instante de tiempo t_f que representa el límite superior del intervalo de consulta. Luego de localizar la entrada de la bitácora cuyo tiempo es el menor más cercano a t_f, se comienzan a recorrer las bitácoras por medio de las referencias al movimiento anterior, y se obtienen las posiciones espaciales correspondientes teniendo en cuenta que el tiempo correspondiente al movimiento sea mayor o igual al instante de tiempo t_o que representa el límite inferior del intervalo de consulta.

Se puede encontrar en [4] información más especifica acerca de los algoritmos que resuelven las consultas antes mencionadas y otros detalles.

4 EVALUACIÓN EXPERIMENTAL

A continuación, presentamos la evaluación experimental de nuestro índice comparándolo con el método en el cual nos basamos [5], donde se evaluó la utilización del espacio en ambos índices y el desempeño de los algoritmos utilizados para responder los distintos tipos de consulta.

Los lotes de prueba utilizados para realizar la evaluación experimental son significativamente representativos, ya que se tuvieron en cuenta las propiedades y magnitudes de los lotes de prueba usados en experimentaciones relacionadas a la temática y ocurrentes en aplicaciones reales según se constata en la literatura afín. Además, se utilizaron los mismos lotes de prueba con los que se evaluó SEST-Index, para poder hacer una comparación más certera.

Se utilizaron lotes con 1000, 3000 y 5000 objetos (puntos en el plano), moviéndose en el espacio durante 50 instantes de tiempo consecutivos, con un porcentaje de movilidad por instante de 1, 3, 5, 7, 9, 11, 13 y 15 por ciento. Estos puntos se distribuyeron uniformemente en el instante de tiempo inicial. Luego, se movieron aleatoriamente durante los próximos 50 instantes de tiempo hasta llegar al instante final. Para medir el tiempo de acceso en las consultas se calculó el promedio de bloques leídos tras realizar 100 consultas aleatorias. Las figuras mencionadas a continuación se encuentran en Anexo Figuras del presente artículo.

Los resultados obtenidos en el análisis empírico arrojaron que el método modificado representa realmente una mejora. Si bien desde el punto de vista de la cantidad de cuadros presentados, las mejoras reales se ven en pocos, ocurre que: i) debe considerarse que la estructura original fue modificada ampliándosela en estructuras de almacenamiento, ii) se continuó manteniendo el buen desempeño de la original, y iii) se agregó un tipo de consulta adicional, que, generalmente, requiere índices separados para su tratamiento.

Con todo ello, mostramos una estructura integradora, con al menos igual desempeño que su estructura original.

4.1 Uso de espacio en disco

Se midió el número de bloques de disco usados por SEST-Index y D*R-Tree con bitácoras de 4, 8 y 12 bloques. Se utilizaron archivos de datos con 1000, 3000 y 5000 puntos con un porcentaje de

movilidad de 1, 3, 5, 7, 9, 11, 13 y 15 por ciento.

Podemos observar que para ambas estructuras, a medida que aumenta el porcentaje de movilidad mayor es la cantidad de bloques necesarios para almacenarlas, como muestra la Figura 2. Se observa también que la diferencia en la cantidad de bloques utilizados por ambas estructuras aumenta a medida que se incrementa el porcentaje de movilidad. En el peor de los casos, el costo adicional en D*R-Tree se aproxima a un 9 por ciento del tamaño total de la estructura. El mismo comportamiento se puede observar para otros casos, como muestran las Figuras 3 y 4.

Si bien en los resultados se observa que SEST-Index utiliza, en porcentaje, menor espacio en disco que D*R-Tree, se debe a que nuestra estructura responde eficientemente a la consulta de tipo Trayectoria, lo que implica el uso de estructuras adicionales y modificaciones en la bitácora. Sin embargo, este costo resulta insignificante en comparación a las ventajas obtenidas. La experimentación nos mostró que al aumentar la cantidad de objetos, la cantidad de bloques utilizados aumentó proporcionalmente. También que, mientras la cantidad de bloques por bitácora aumenta, menor es el espacio utilizado por la estructura en general ya que se necesitan mantener menos puntos de referencia (ver Figura 5).

4.2 Consultas

Se midió el promedio de bloques de disco leídos por SEST-Index y D*R-Tree con bitácoras de 4, 8 y 12 bloques en 100 consultas aleatorias de tipo TimeSlice, Intervalo, Eventos y Trayectoria. Se utilizaron archivos de datos con 1000, 3000 y 5000 puntos con un porcentaje de movilidad de 1, 3, 5, 7, 9, 11, 13 y 15 por ciento.

4.2.1 Consultas TimeSlice

Se realizaron 100 consultas con rectángulos que cubren un 5 y 10 por ciento del área total. La posición del rectángulo y el instante de tiempo de la consulta son aleatorios.

Luego de realizar la experimentación notamos que en D*R-Tree no hay significativas variaciones en el número de bloques leídos, para responder este tipo consulta, con respecto a SEST-Index ya que ambos índices trabajan de forma similar para resolverla.

Como podemos ver en la Figura 9, el porcentaje de movilidad no afecta significativamente en la cantidad de bloques leídos. Se observa también, que mientras más grande es la bitácora en ambas estructuras, mayor es el número promedio de bloques leídos. Esto se debe a que se tiene que recorrer secuencialmente más bloques de la bitácora. Con bitácoras más pequeñas se leen, en promedio, menos bloques de disco para responder una consulta de tipo TimeSlice. Sin embargo, como vimos anteriormente esto implica mayor espacio para mantener la estructura total.

4.2.2 Consultas Intervalo

Se realizaron 100 consultas con rectángulos que cubren un 5 y 10 por ciento del área total para intervalos de 5 y 10 unidades de tiempo. La posición del rectángulo y el intervalo de tiempo de consulta son aleatorios.

En los resultados podemos observar que en D*R-Tree no hay variación importante en el número de bloques leídos con respecto a SEST-Index. Esto se debe a que ambas estructuras resuelven de forma similar este tipo de consulta.

Viendo la Figura 10, observamos que mientras mayor es el porcentaje de movilidad en ambas estructuras, mayor es el número promedio de bloques leídos, ya que por cada instante de tiempo se necesitan almacenar mayor cantidad de entradas en la bitácora. Por la misma razón, para mayor cantidad de objetos, mayor es la cantidad de bloques leídos. Si bien el tamaño de la bitácora influye

en el espacio utilizado para almacenar la estructura como mencionamos anteriormente, no afecta en la cantidad de bloques leídos para responder este tipo de consulta (ver Figura 13).

4.2.3 Consultas Eventos

Se realizaron 100 consultas con rectángulos que cubren un 5 y 10 por ciento del área total. La posición del rectángulo y el tiempo de consulta son aleatorios.

En los resultados podemos observar que, en promedio las estructuras difieren en 1 o 2 bloques leídos. Además observamos que mientras más grande es el porcentaje de movilidad y mayor la cantidad de objetos, mayor es el número promedio de bloques leídos, ya que en cada instante de tiempo ocurren más eventos. La cantidad de bloques por bitácora no disminuye el rendimiento de nuestra estructura con respecto a la original. Cabe aclarar que el área de consulta no influye en los resultados ya que en todos los casos, debemos examinar todos los movimientos que se produjeron en el instante consultado. Los resultados se muestran en las Figuras 14 a la 17.

4.2.4 Consultas Trayectoria

Se realizaron 100 consultas para intervalos de 5 y 10 unidades de tiempo. La selección del objeto a consultar es aleatoria, como así también la ubicación del intervalo de tiempo de consulta.

Observamos claramente que siempre en consultas de tipo Trayectoria, D*R-Tree lee un número de bloques significativamente menor que SEST-Index, dado que éste no responde a este tipo de consulta en forma eficiente por no estar diseñada para este fin, por lo tanto, para recuperar la trayectoria debe hacer una búsqueda exhaustiva en las estructuras de almacenamiento. Primero se debe encontrar el punto de referencia que contiene el instante correspondiente al límite superior del intervalo de consulta. Luego, se recorren secuencialmente las bitácoras en forma descendiente con respecto al tiempo hasta encontrar el instante correspondiente al límite inferior del intervalo de consulta. De esta manera, se localizan las entradas de las bitácoras donde se almacenó un movimiento correspondiente al objeto de consulta. Por último se retornan las posiciones encontradas durante dicho recorrido.

D*R-Tree, por otro lado, simplemente utiliza el *Índice de Trayectoria*, el cual está implementado como un direccionamiento directo sobre los identificadores de objetos, para encontrar el último movimiento del objeto de consulta en la bitácora correspondiente al límite superior del intervalo. Luego, se recorren las bitácoras utilizando las referencias a entradas de una bitácora en la cual está almacenado el movimiento inmediatamente anterior correspondiente al objeto de la consulta. De esta manera, D*R-Tree se asegura de leer sólo los bloques donde existe una referencia a un movimiento del objeto. Notamos que SEST-Index puede no retornar un resultado ya que en intervalos de tiempo donde no se han registrado movimientos del objeto, en el recorrido exhaustivo no se encontrará información del objeto consultado. En D*R-Tree esto no sucede ya que para estos casos mantenemos una referencia en el *Índice de Trayectorias*. Es importante destacar que en nuestra propuesta, el porcentaje de movilidad y la cantidad de objetos no afecta el desempeño del algoritmo que responde a este tipo de consulta ya que el diseño de la estructura permite acceder sólo a aquellas posiciones donde el objeto ha registrado un movimiento.

Como muestran las Figuras 18, 19 y 20, D*R-Tree se mantiene estable en la cantidad de bloques leídos, aunque el porcentaje de movilidad y la cantidad de objetos varíen. Los resultados corresponden a pruebas realizadas con 1000 objetos para intervalos de 5 y 10 unidades de tiempo ya que en las consultas no influye la cantidad de objetos. Para poder observar cual es el comportamiento de la estructura propuesta, se realizaron pruebas donde se compara D*R-Tree con distintos tamaños de intervalo de tiempo de consulta. Se observó que al incrementar el intervalo de consulta, la diferencia entre la cantidad de bloques leídos no es significativa. Los resultados que se muestran en la Figura 21 corresponden a pruebas realizadas con 1000 objetos.

5 CONCLUSIONES Y TRABAJO FUTURO

Como mencionamos anteriormente, para satisfacer la creciente demanda de aplicaciones que deben manejar objetos que evolucionan en el tiempo son necesarios los MAETs. En el transcurso de esta investigación, observamos que la mayoría de estos métodos se abocan a responder eficientemente ciertos tipos de consultas, lo que nos motivó a desarrollar un índice capaz de mantener la información necesaria para poder responder los cuatro tipos de consulta más solicitados en general: TimeSlice, Intervalo, Eventos y Trayectoria. De esta forma, surgió el índice D*R-Tree, que permite almacenar datos espacio-temporales y responder eficientemente consultas sobre éstos. Posteriormente, las evaluaciones experimentales nos brindaron resultados satisfactorios, dado que mantuvimos el mismo desempeño que la estructura original, pero además, ampliamos el conjunto de consultas espacio-temporales.

Como trabajo futuro, nos proponemos realizar el análisis de costo de los algoritmos presentados, como así también proponer una implementación real en disco para alguna aplicación en particular. Además, queremos incluir al índice otros tipos de consultas, tales como los de reunión espacio-temporal, la búsqueda de los vecinos más cercanos, entre otras. Con fines didácticos, pretendemos desarrollar un aplicativo que permita visualizar el funcionamiento interno de D*R-Tree, para observar 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.

Por todo lo expuesto, el desarrollo de este trabajo implicó la formación en profundidad de la temática y hemos pretendido, esencialmente, destacar resultados obtenidos, algoritmos y estructuras de datos desarrolladas para Métodos de Acceso Espacio-Temporales, como así también las aplicaciones relacionadas, dejando temas para el estudio e investigación futuros.

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.

ANEXO FIGURAS

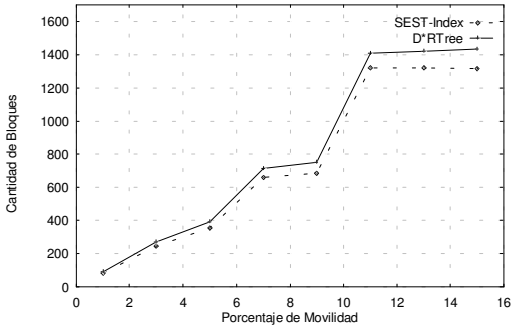

Figura 2. Número de bloques utilizados para 1000 objetos (4 bloques por bitácora)

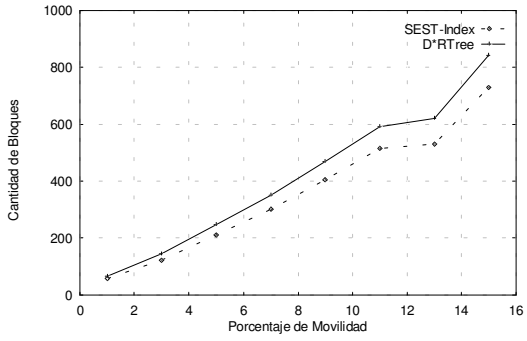

Figura 3. Número de bloques utilizados para 1000 objetos (8 bloques por bitácora)

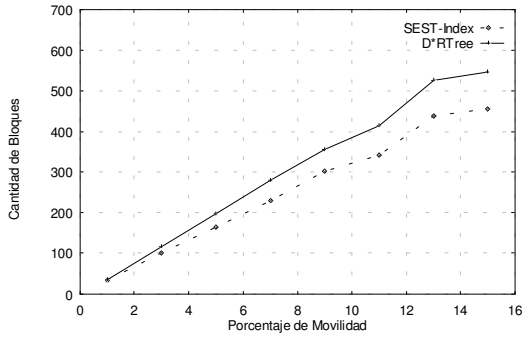

Figura 4. Número de bloques utilizados para 1000 objetos (12 bloques por bitácora)

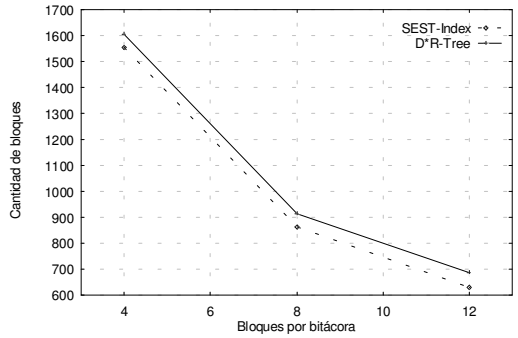

Figura 5: Número de bloques utilizados para 5000 objetos con un porcentaje de movilidad del 1%

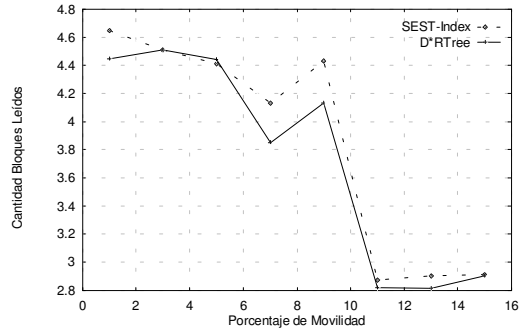

Figura 6: Número de bloques leídos para TimeSlice en un área de 5% para 1000 objetos (4 bloques por bitácora)

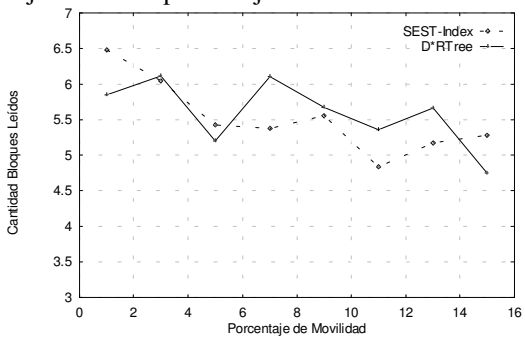

Figura 7: Número de bloques leídos para TimeSlice en un área de 5% para 1000 objetos (8 bloques por bitácora)

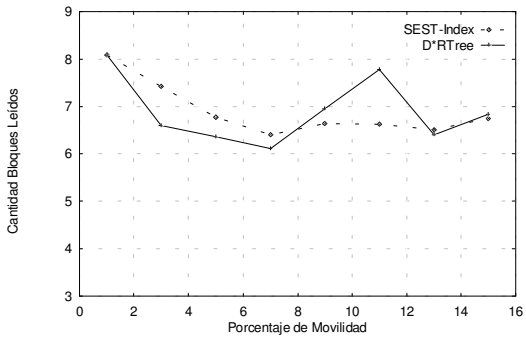

Figura 8. Número de bloques leídos para TimeSlice en un área de 5% para 1000 objetos (12 bloques por bitácora)

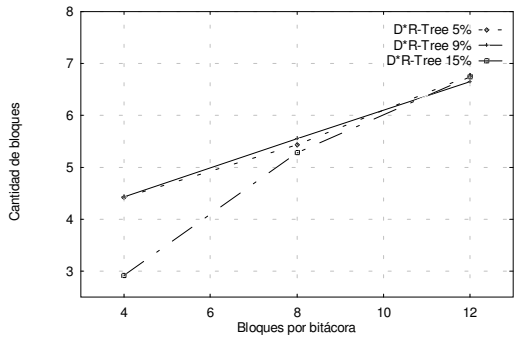

Figura 9: Número de bloques leídos para TimeSlice con 1000 objetos y un porcentaje de movilidad del 5%, 9% y 15% para bitácoras de 4, 8 y 12 bloques

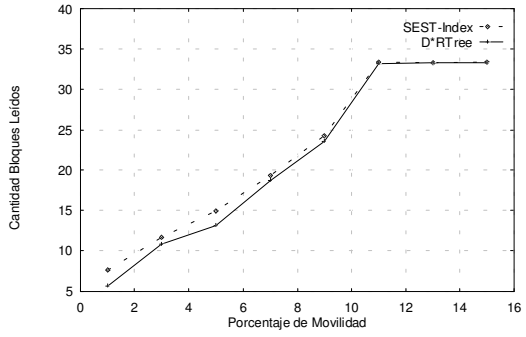

Figura 10. Número de bloques leídos para Intervalo de tamaño 10 en un área de 10% para 1000 objetos (4 bloques por bitácora)

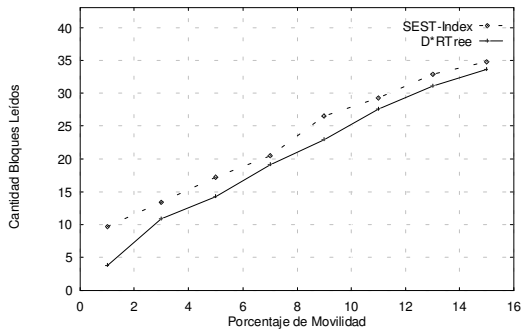

Figura 11. Número de bloques leídos para Intervalo de tamaño 10 en un área de 10% para 1000 objetos (8 bloques por bitácora)

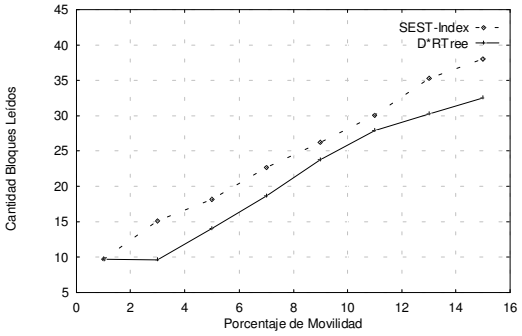

Figura 12. Número de bloques leídos para Intervalo de tamaño 10 en un área de 10% para 1000 objetos (12 bloques por bitácora)

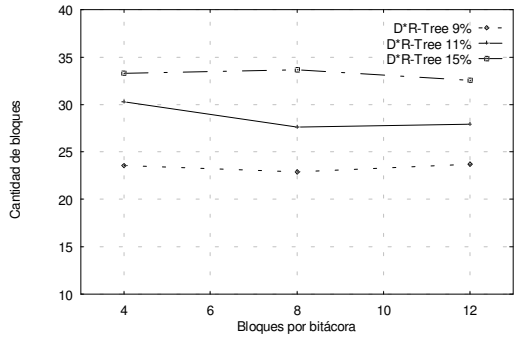

Figura 13: Número de bloques leídos para Intervalo con 1000 objetos y un porcentaje de movilidad del 9%, 11% y 15% para bitácoras de 4, 8 y 12 bloques

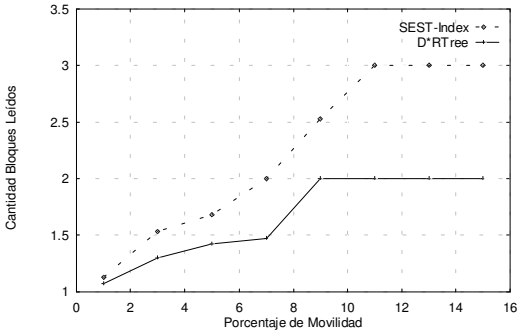

Figura 14. Número de bloques leídos para Eventos en un área de 5% para 1000 objetos (4 bloques por bitácora)

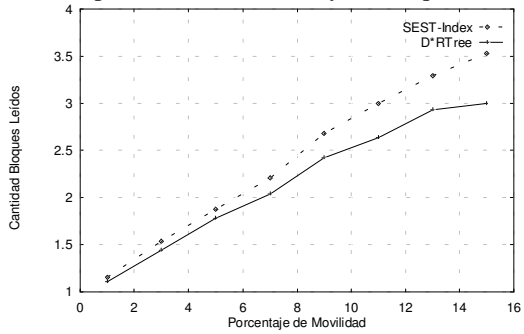

Figura 15. Número de bloques leídos para Eventos en un área de 5% para 1000 objetos (8 bloques por bitácora)

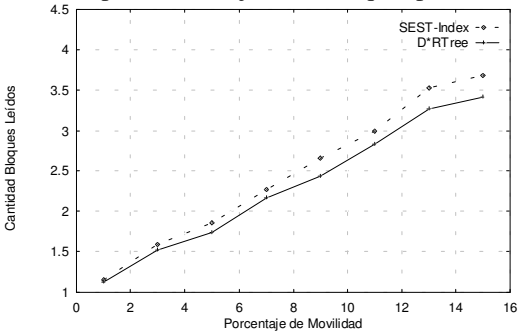

Figura 16. Número de bloques leídos para Eventos en un área de 5% para 1000 objetos (12 bloques por bitácora)

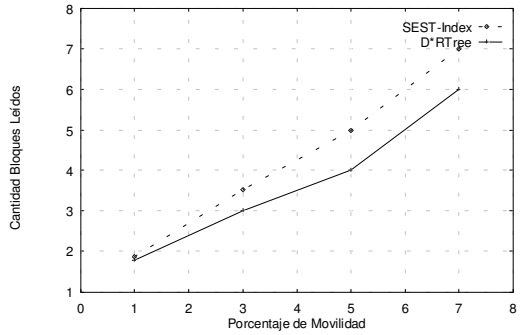

Figura 17. Número de bloques leídos para Eventos en un área de 5% para 5000 objetos (8 bloques por bitácora)

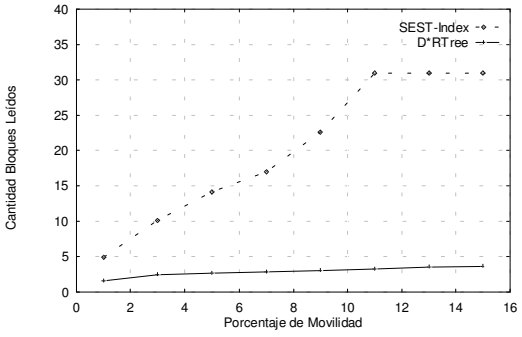

Figura 18. Número de bloques leídos para Trayectoria de tamaño 10 para 1000 objetos (4 bloques por bitácora)

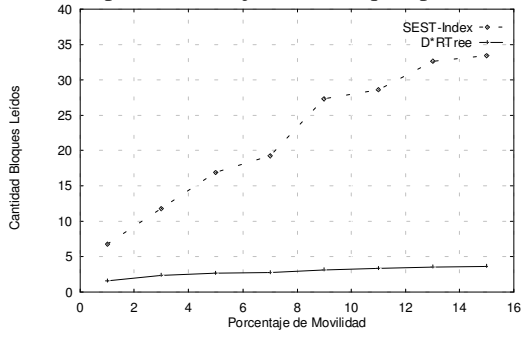

Figura 19. Número de bloques leídos para Trayectoria de tamaño 10 para 1000 objetos (8 bloques por bitácora)

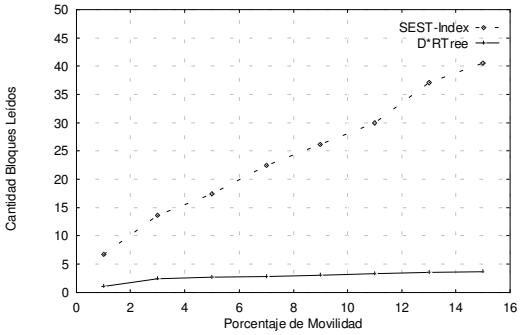

Figura 20. Número de bloques leídos para Trayectoria de tamaño 10 para 1000 objetos (12 bloques por bitácora)

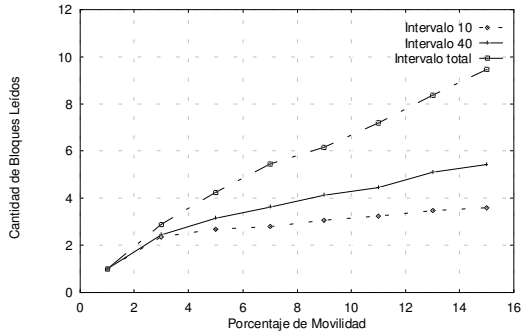

Figura 21. Número de bloques leídos para Trayectoria de tamaño total, 10 y 40 para 1000 objetos (12 bloques por bitácora)

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] Bayer, R. and McCreight, E. Organization and Maintenance of Large Ordered Indexes. Acta Informática 1, 173-189, (1972).

[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] Dorzán M., Gómez Barroso J. Gagliardi E. D*R-Tree: un método de acceso espacio-temporal. CACIC (2005). ISBN 950-698-166-3

[5] Gutiérrez Gilberto, Navarro Gonzalo, Rodriguez Andrea, Gonzales Alejandro, Orellana José. A Spatiotemporal Access Method based on Snapshots and Events. ACM GIS'05, Bremen, Germany. (2005)

[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] Manolopoulos, Yannis, Nanopoulos, Alexandros, Papadopoulos, Apostolos N. y Theodoridis, Yannis. R-trees have grown everywhere. (2003).

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

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

[11] 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, (1999)

[12] 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)

[13] 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,Redondo Beach, CA, (2001)

[14] 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)

[15] 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)

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

[17] 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)

[18] 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. (1990)