I+3 R-Tree: un método de acceso espacio-temporal

Edilma O. Gagliardi (1), Fernando D. Carrasco, Juan C. García Sosa
Facultad de Ciencias Físico Matemáticas y Naturales,
Universidad Nacional de San Luis, Argentina
{oli, fdcarras, jcgarcia}@unsl.edu.ar

Gilberto Gutierrez
Universidad del Bío Bío, Chile
ggutierr@ubiobio.cl

Resumen: Este artículo presenta una estructura de indexación de datos espacio-temporales para objetos que se mueven en una región fija a lo largo del tiempo. Este método de acceso permite procesar consultas que contemplen relaciones espacio-temporales, considerando el pasado y el presente. El método considera el procesamiento de cuatro tipos de consultas, a) timeslice, b) intervalo, c) evento y d) trayectoria. Mostramos resultados de la evaluación experimental que nos permitieron observar el comportamiento de esta estructura respecto de las consultas mencionadas.

Palabras claves: Bases de Datos Espacio-temporales, Métodos de acceso Espacio-temporal.

1 Introducción

Hay aplicaciones del mundo real donde los objetos de interés se caracterizan por mantener dos atributos comunes: el espacio y el tiempo. De esta forma, un objeto se puede identificar por la posición y el área que ocupa en cualquier instante de tiempo. Estos objetos dan origen al tipo de dato espacio-temporal. Así, aquellas aplicaciones informáticas cuya función sea manejar este tipo de objetos deberán proveer un soporte para manejar este tipo de datos, dando origen a los Sistemas de Administración de Bases de Datos Espacio-Temporales (STDBMS).
En particular, para la indexación de información espacio-temporal, un STDBMS debe proveer Métodos de Acceso Espacio-Temporal (MAETs) que eviten examinar todos los objetos al momento de responder consultas, y que sean eficientes frente a una alta asiduidad de consultas. La indexación se utiliza para optimizar la recuperación de datos almacenados en memoria secundaria y el tiempo de procesamiento de las consultas.
Esta propuesta consiste en una estructura I+3 R-Tree, compuesta por un R-Tree de tres dimensiones, llamado 3D R-Tree, para almacenar datos históricos, y el de una nueva estructura denominada I para guardar datos correspondientes a las posiciones actuales de los objetos. El uso de un 3D R-Tree nos permitirá recuperar información

(1) Proyecto Tecnologías Avanzadas de Bases de Datos. FCFMyN. UNSL.

relacionada a la permanencia de los objetos en determinadas posiciones o regiones, en instantes de tiempo pasados. Mientras que la segunda estructura se propone con el objeto de recuperar las posiciones actuales de los objetos. Así, con este método de acceso es posible procesar consultas cuyos predicados contemplen relaciones espacio-temporales, considerando no sólo el pasado sino también el presente. Nuestro método permite procesar consultas espacio/temporales de cuatro tipos: a) timeslice, que permite recuperar todos los objetos presentes en una cierta región en un instante dado t, b) intervalo, la cual permite recuperar todos los objetos existentes en una cierta región en el intervalo de tiempo [t1,t2], c) evento, que permite recuperar todos los objetos que entraron-salieron en una determinada región y d) trayectoria que permite, recuperar el camino que ha seguido un objeto.

2 Índices Relacionados

En esta sección se hace una presentación de los MAETs en los cuales se basa nuestra propuesta, con el objeto de conocer sus características principales y funcionamiento.

2.1 R-Tree

El R-Tree [1] es una extensión del B-Tree, que sirve para almacenar objetos multidimensionales, por ejemplo puntos y regiones en el espacio. Un R-Tree organiza una colección de objetos espaciales en forma jerárquica donde las hojas contienen punteros a los datos en sí, y los nodos intermedios contienen el *mínimo rectángulo envolvente* (MBR) al objeto espacial que contiene a sus hijos.

2.2 3D R-Tree

El 3D R-Tree [3] es un R-Tree para tres dimensiones, donde se consideran dos dimensiones espaciales y la tercera dimensión temporal.
Cuando un objeto se mueve a otra posición o cambia de forma, se crea un nuevo MBR para representar el cambio del objeto, y dicho MBR conteniendo tanto la extensión espacial del objeto como así también su vida útil (tiempo), es insertado en el 3D R-Tree.
Una consulta espacio-temporal se modela como un cubo, donde la superficie de su base representa la región de consulta y su altura corresponde al intervalo temporal de la consulta. Usando el cubo especificado en la consulta, se busca en el 3D R-Tree todos los objetos que se interceptan con él utilizando el mismo procedimiento de búsqueda especificado en la propuesta original del R-Tree.

2.3 2+3 R-Tree

El objetivo del 2+3 R-Tree [2] es indexar información actual y pasada de objetos que se encuentran en movimiento. La idea principal es tener dos R-Tree's de manera

separada. Un R-Tree de dos dimensiones para almacenar los objetos que tienen indefinido el tiempo final de su estancia o permanencia en un lugar porque aún no se han movido de tal sitio. Y un segundo R-Tree tridimensional, de dos dimensiones espaciales y una dimensión temporal, para almacenar aquellos objetos que tienen definidos el tiempo inicial y final de permanencia en una determinada posición. Mientras el tiempo final de un objeto es desconocido, el objeto se almacena en el R-Tree bidimensional guardando su tiempo inicial su id y su posición. Una vez que el objeto cambia su posición, se construye un MBR tridimensional, el cual es insertado en el 3D R-Tree y es eliminado del R-Tree bidimensional. Dependiendo del tiempo de la consulta, puede ser necesario hacer la búsqueda en ambos árboles.

3 I+3 R-Tree

La estructura que proponemos, llamada I+3 R-Tree, es una alternativa al 2+3 R-Tree, con la cual pretendemos superar algunas desventajas o carencias que posee la misma, por ejemplo el 2+3 R-Tree no resuelve consultas de eventos ni de trayectoria.
El I+3 R-Tree reemplaza el R-Tree bidimensional de la estructura original, por una estructura de datos que actúa como un índice de acceso a todos los objetos con sus respectivos últimos movimientos. De esta manera, la información actual se mantiene en el índice, mientras que al igual que en el 2+3 R-Tree la información histórica se mantiene en el 3D R-Tree.
El I+3 R-Tree está enfocado para apoyar aplicaciones en las cuales la cantidad de objetos en movimiento es fija y el espacio donde los objetos se mueven es acotado y conocido, y, además, hay una alta frecuencia de movimientos. Se asume que los objetos son capaces de informar, en forma discreta las coordenadas y el tiempo en que el objeto alcanzó una nueva posición espacial.
La estructura permite procesar consultas de tipo timeslice e intervalo de la misma manera que en el 2+3 R-Tree, además permite resolver consultas de tipo evento y trayectoria lo cual se considera una mejora sobre la estructura de datos en la cual nos basamos.
Por otro lado, ninguna de las propuestas anteriores resuelve consultas de trayectoria. Este punto sí es resuelto en el I+3 R-Tree, agregando una modificación a la estructura 3D R-Tree. Dicha modificación consiste en mantener una lista para cada objeto enlazando en el 3D R-Tree los cubos correspondientes a las diferentes posiciones por las cuales ha pasado el objeto. El acceso a estas listas se realiza a través del Índice desde la posición actual de cada objeto mediante un direccionamiento directo, ya que la cantidad de objetos es fija.
En el 3D R-Tree se almacenan los cubos definidos, cada uno de los cuales representa la estadía de un objeto en una posición durante un intervalo definido de tiempo. Es decir, toda la información espacio – temporal histórica se mantiene en el 3D R-Tree. Las nuplas almacenadas en el 3D R-Tree son del tipo <oid, mbr_3D, p_tray> donde:

- oid: código identificador del objeto;
- mbr_3D: región tridimensional cuya altura representa el intervalo temporal durante el cual el objeto se mantuvo en la posición espacial definida por su base;

- p_tray: puntero al cubo anterior correspondiente al mismo oid, utilizado para mantener un historial de trayectoria.

En el Índice se almacenan los cubos abiertos. Esto es aquellos cubos para los cuales su instante final en una determinada posición aún no está definido. También se guardan las referencias necesarias a los cubos anteriores que describen la trayectoria del objeto. El índice es una lista secuencial de N elementos, donde N es la cantidad de objetos considerados, que almacena nuplas de la forma <oid, mbr, t, p3D, pa, ps> donde:

- mbr: región aproximada que ocupa actualmente el objeto;
- t: tiempo de llegada del objeto a su ubicación actual;
- p3D: puntero al cubo anterior correspondiente al mismo oid, utilizado para mantener un historial de trayectoria;
- pa: puntero al objeto insertado en el instante de tiempo inmediatamente anterior; y
- ps: puntero al objeto insertado en el instante de tiempo siguiente.

La lista secuencial se encuentra ordenada por el oid; como la cantidad de objetos que se almacenan es fija, esta lista puede ser accedida como un direccionamiento directo, lo cual es una ventaja a la hora de realizar búsquedas por oid para resolver consultas de tipo trayectoria. Esta lista se mantiene en memoria principal.

Los punteros pa y ps son utilizados para enlazar los tiempos en orden creciente. Este ordenamiento por tiempo se realiza con el objetivo de poder acceder de manera menos costosa a un instante determinado en el índice para resolver consultas de tipo timeslice, intervalo o evento.

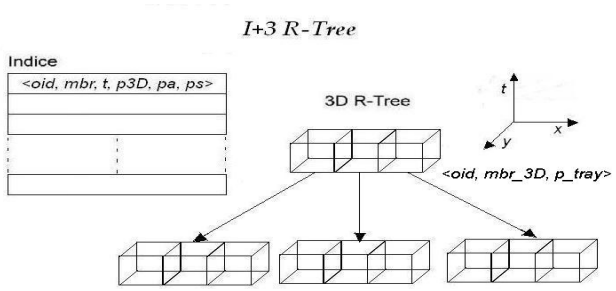


Figura 1: Estructura del I+3 R-Tree

4 Los Algoritmos de Consultas en el I+3 R-Tree

4.1 Consulta Timeslice

El algoritmo para el procesamiento de consultas timeslice hace uso de un algoritmo de búsqueda en el índice, y de un algoritmo de búsqueda en el 3D R-Tree y devuelve los resultados en un conjunto Q de objetos que conforman la respuesta.

El algoritmo armar_region() construye la región de consulta a partir de los puntos que representan los extremos opuestos de la misma y el tiempo de consulta.
El algoritmo BuscarEn3DR() es simplemente el algoritmo de búsqueda del 3D R-Tree, que recibe como parámetro una región plana de consulta, y devuelve todos los objetos que se intersectan con esta región de consulta en el tiempo ti.

Algoritmo *Timeslice(*$p1$, $p2$,, t_i*)*

Entrada: $p1$, $p2$ son los puntos utilizados para armar el rectángulo de consulta, t_i 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. Si t_i > *TMax3D* entonces
2. $Q \leftarrow$ *BuscarEnIndice(*$p1$, $p2$, t_i*)* *// La respuesta está solo en el índice*
3. Sino
4. *region* $\leftarrow$ *armar_region(*$p1$, $p2$, t_i, t_i*)*
5. Si t_i < *TMinInd* entonces
6. $Q \leftarrow$ *BuscarEn3DR(region)*
7. Sino
8. $Q \leftarrow$ *BuscarEnIndice(*$p1$, $p2$, t_i*)*
9. $Q \leftarrow Q \cup$ *BuscarEn3DR(region)*
10. Retornar *Q*

Algoritmo *BuscarEnIndice(*$p1$, $p2$, t_i*)*

Entrada: $p1$, $p2$, son los puntos utilizados para armar el rectángulo de consulta, t_i es el instante de tiempo sobre el cual se realiza la consulta.
Salida: Q es el conjunto de objetos que responden a la consulta.
Nupla es un objeto de la forma <*oid*, *pos*, *tiempo*, *Psig*, *Pant, Ptray*> donde *oid* es el identificador del objeto, *pos* es la posición del objeto, *tiempo* es el instante en el cual el objeto llegó a *pos*, *Psig* es el puntero a la siguiente nupla ordenada por tiempo, *Pant* es el puntero a la nupla anterior ordenada por tiempo y *Ptray* es puntero a la pos anterior del objeto para mantener la trayectoria.

1. *Nupla* $\leftarrow$ *primero*
2. *fin* $\leftarrow$ *false*
3. *region* $\leftarrow$ *armar_region(*$p1$, $p2$, t_i, t_i*)*
4. Mientras *Nupla*(*tiempo*) <= t_i and not *fin*
5. Si *Nupla*(*pos*) $\cap$ *region*
6. $Q \leftarrow Q \cup$ *Nupla*
7. Si *Nupla*(*Psig*) <> Null
8. *Nupla* $\leftarrow$ *Nupla*(*Psig*)
9. Sino
10. *fin* $\leftarrow$ *true*
11. Retornar *Q*

4.2 Consulta Intervalo

El algoritmo para el procesamiento de consultas de intervalo utiliza los mismos algoritmos de búsqueda que el algoritmo para procesar consultas timeslice. La región construida por armar_region() es un cubo con un piso y un techo dados por los limites inferior (ti) y superior (tk) del intervalo de consulta respectivamente. El algoritmo

BuscarEn3DR() recibe en este caso, una región de consulta cúbica como parámetro, y devuelve todos los objetos que se intersecan con éste cubo de consulta.

Algoritmo *Intervalo(*$p1, p2,, t_i, t_k$*)*

Entrada: p1, *p2* son los puntos utilizados para armar el rectángulo de consulta, 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. Si $t_i > TMax3D$ entonces
2. $Q \leftarrow BuscarEnIndice(p1, p2, t_i)$
3. Sino
4. $region \leftarrow armar\_region(p1, p2, t_i, t_k)$
5. Si $t_i < TMinInd$ entonces
6. $Q \leftarrow BuscarEn3DR(region)$
7. Sino
8. $Q \leftarrow BuscarEnIndice(p1, p2, t_i)$
9. $Q \leftarrow Q \cup BuscarEn3DR(region)$
10. Retornar *Q*

4.3 Consulta Evento

El algoritmo para el procesamiento de consultas de evento utiliza los algoritmos BuscarEnIndice_evento( ) y BuscarEn3DR_evento( ).

Algoritmo *Evento(*$p1, p2,, t_i$*)*

Entrada: $p1, p2,$ son los puntos utilizados para armar el rectángulo de consulta, t_i 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. Si $t_i > TMax3D$ entonces
2. $Q \leftarrow BuscarEnIndice\_evento(p1, p2, t_i)$
3. Sino
4. $region \leftarrow armar\_region(p1, p2, t_i, t_i)$
5. Si $t_i < TMinInd$ entonces
6. $Q \leftarrow BuscarEn3DR\_evento(region)$
7. Sino
8. $Q \leftarrow BuscarEnIndice\_evento(region, t_i)$
9. $Q \leftarrow Q \cup BuscarEn3DR\_evento(region)$
10. Retornar *Q*

Algoritmo *BuscarEnIndice_evento(*$p1, p2, t_i$*)*

Entrada: $p1, p2,$ son los puntos utilizados para armar el rectángulo de consulta, t_i es el instante de tiempo sobre el cual se realiza la consulta.

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

Nupla es un objeto de la forma <*oid*, *pos*, *tiempo*, *Psig*, *Pant, Ptray*> donde *oid* es el identificador del objeto, *pos* es la posición del objeto, *tiempo* es el instante en el cual el objeto llegó a *pos*, *Psig* es el puntero a la siguiente nupla ordenada por tiempo, *Pant* es el puntero a la nupla anterior ordenada por tiempo y *Ptray* es puntero a la pos anterior del objeto para mantener la trayectoria.

1. *Nupla* ← *primero*
2. *fin* ← *false*

3. *region* ← *armar_region*(*p*1, *p*2, t_i, t_i)
4. Mientras *Nupla*(*tiempo*) = t_i and not *fin*
5. Si *Nupla*(*pos*) ∩ *region*
6. *Q* ← *Q* ∪ *Nupla*
7. Si *Nupla*(*Psig*) <> Null
8. *Nupla* ← *Nupla*(*Psig*)
9. Sino
10. *fin* ← *true*
11. Retornar *Q*

4.4 Consulta Trayectoria

El algoritmo para el procesamiento de consultas de trayectoria, toma como entrada un intervalo de tiempo, y el identificador del objeto del cual se desea conocer su trayectoria durante ese intervalo de tiempo. Por medio de BuscarEnIndice_tray() se accede al índice para recuperar la posición actual del objeto de consulta, luego se recorre la lista que mantiene enlazadas las diferentes posiciones del objeto a lo largo del tiempo.

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 posiciones que responden a la consulta.
Nupla es un objeto de la forma <*oid*, *pos*, *tiempo*, *Psig*, *Pant, Ptray*> donde *oid* es el identificador del objeto, *pos* es la posición del objeto, *tiempo* es el instante en el cual el objeto llegó a *pos*, *Psig* es el puntero a la siguiente nupla ordenada por tiempo, *Pant* es el puntero a la nupla anterior ordenada por tiempo y *Ptray* es puntero a la pos anterior del objeto para mantener la trayectoria.

1. *Nupla* ← *BuscarEnIndice_tray(oid)*
2. *fin* ← *false*
3. Si *Nupla*(*tiempo*)>= t_i and *Nupla*(*tiempo*) <= t_k
4. *Q* ← *Q* ∪ *Nupla*(*pos*)
5. Mientras *Nupla*(*Ptray*) <> Null and not *fin*
6. *Nupla* ← *Nupla*(*Ptray*)
7. Si *Nupla*(*tiempo*)>= t_i and *Nupla*(*tiempo*) <= t_k
8. *Q* ← *Q* ∪ *Nupla*(*pos*)
9. Sino
10. Si *Nupla*(*tiempo*) < t_i
11. *fin* ← *true*
12. Retornar *Q*

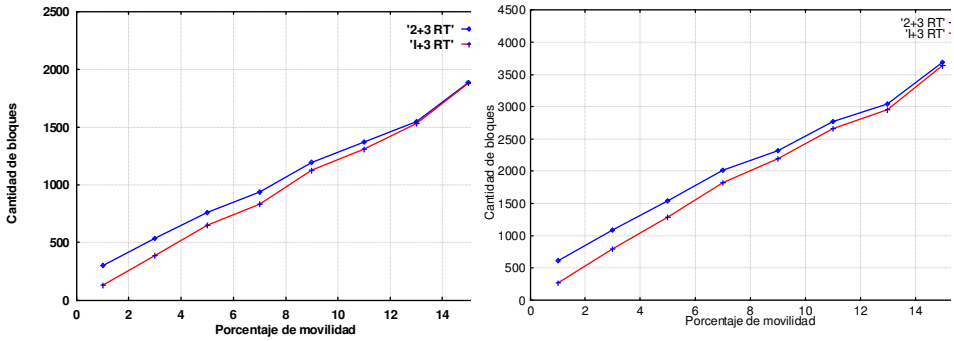
5 Evaluación Experimental

La evaluación experimental realizada consiste en comparar nuestra estructura con respecto a la estructura original en la cual nos basamos [2].

Para las evaluaciones se utilizan lotes de prueba con 1000, 3000 y 5000 objetos, moviéndose en el espacio durante 50 instantes de tiempo consecutivos, con un porcentaje de movilidad de 1, 3, 5, 7, 9, 11, 13 y 15 por ciento por instante.

5.1 Comparación del uso de espacio en disco.

Se midió la cantidad de bloques utilizados en disco para ambas estructuras.



Nº de bloques utilizados para 5000 obj. Nº de bloques utilizados para 10000 obj.

Figura 2

Como se puede observar en la figura 2, la cantidad de bloques utilizados por las estructuras aumenta a medida que crece el porcentaje de movilidad. También se puede apreciar la diferencia en la cantidad de bloques utilizados por ambas estructuras.
El I+3 R-Tree presenta un menor costo que el 2+3 R-Tree, esto se debe a que nuestra implementación solo maneja en disco la estructura que almacena la información histórica mientras que el índice se mantiene en memoria principal. 2+3 R-Tree maneja ambas estructuras en disco lo cual implica el costo adicional.

5.2 Comparación del número de bloques leídos.

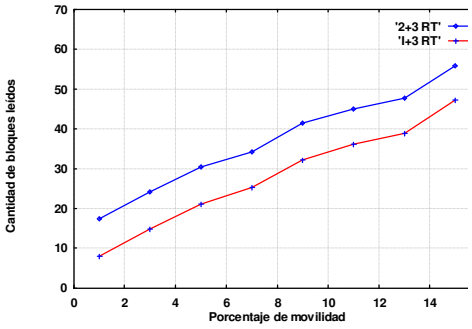
Para medir el desempeño de los algoritmos de consulta se calculó el promedio de bloques leídos tras realizar 100 consultas aleatorias.

5.2.1 Consulta Timeslice

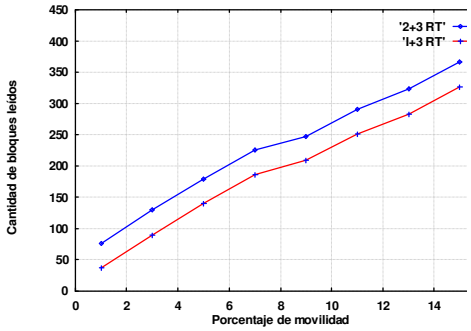
Podemos observar en la figura 3 que la cantidad de bloques leídos se incrementa a medida que se incrementa el porcentaje de movilidad.
Ambas estructuras se comportan en este caso de manera similar, siendo un poco mas alto el costo de consulta en el 2+3 R-Tree debido a los accesos a disco extras necesarios para obtener las posiciones actuales de los objetos.

5.2.2 Consulta Intervalo

El comportamiento de las estructuras es en este caso similar al de la consulta de timeslice visto en el punto anterior.



Nº de bloques leídos para Timeslice en un área de 15% para 5000 objetos



Nº de bloques leídos para Intervalo de tamaño 10 en un área de 10% para 10000 objetos.

Figura 3

5.2.3 Consulta Eventos

Es un subconjunto de Timeslice, el cual contiene solamente los objetos que ingresan o salen de la región de consulta en el instante de tiempo dado. Por esta razón, la evaluación experimental realizada para la consulta de Timeslice es valida también para la consulta de tipo Eventos.

5.2.4 Consultas de Trayectoria

Para poder comparar el desempeño de la consulta de trayectoria se realizó un análisis de costo de búsqueda para las dos estructuras, tanto el I+3 R-Tree como para el 2+3 R-Tree. Este análisis de costo se realizó debido a que el 2+3 R-Tree no posee un método para resolver la trayectoria.

Cantidad de accesos a disco

I+3 R-Tree: Debido a que en este caso, por la definición del tamaño de hoja que estamos utilizando se requiere un acceso a disco por hoja, el costo máximo va a ser igual a la cantidad de hojas del árbol y el costo medio se calcula como promedio de acceder a todas las hojas debido ya que existe un algoritmo que resuelve la consulta de trayectoria sin la necesidad de realizar una búsqueda exhaustiva. Esto es:

$$\textit{Costo Máximo (peor caso)} = |\,\text{hojas}\,| \qquad \textit{Costo medio} = \frac{1+2+3+\ldots+|\,\text{hojas}\,|}{|\,\text{hojas}\,|}$$

2+3 R-Tree: El costo máximo va a ser la cantidad de hojas en el árbol. El costo medio en este caso es igual al costo máximo, ya que, al no existir un algoritmo para la resolución de la trayectoria, se deberá siempre recorrer la totalidad de las hojas del árbol para resolver la consulta. Esto es:

$$\textit{Costo máximo (peor caso)} = |\,\textit{hojas}\,| \qquad \textit{Costo medio} = |\,\textit{hojas}\,|$$

Cantidad de comparaciones

TH = Tamaño de hoja, TO = Tamaño de objeto, COH = Cantidad de objetos por hoja, CH = Cantidad de hojas, COH (Cantidad de objetos por hoja) = TH/TO

Luego la cantidad de hojas se puede definir como: $CH = \left\lceil \frac{N}{COH} \right\rceil = \left\lceil \frac{N * TO}{TH} \right\rceil$

I+3 R-Tree: Para calcular el costo máximo, basta con obtener el peor caso hipotético de la búsqueda, el cual sería tener, que recorrer completamente todas las hojas del árbol.

$$Costo\ máximo\ (peor\ caso) = |\ hojas\ |\ x\ TH$$

Para el costo medio, tenemos que debido a que el algoritmo de búsqueda utilizado para resolver la trayectoria evita recorrer completamente cada hoja y también evita recorrer todas las hojas, podemos ver dicho costo, como el promedio de búsqueda en cada hoja y el promedio de búsqueda en todas las hojas.

$$Costo\ medio = \frac{TH\left|hojas\right| + \left|hojas\right| + 1}{4}$$

2+3 R-Tree: Al igual que n el I+3 R-Tree, el costo máximo va a ser la cantidad de hojas multiplicado por el tamaño de la hoja. El costo medio en este caso es igual al costo máximo, debido a que, al no existir un algoritmo para la resolución de la trayectoria, se deberá siempre recorrer la totalidad de las hojas del árbol para resolver la consulta. Esto es:

$$Costo\ máximo\ (peor\ caso) = |\ hojas\ |\ x\ TH \quad Costo\ medio = |\ hojas\ |\ x\ TH$$

6 Conclusiones

En este trabajo presentamos un nuevo índice espacio-temporal conjuntamente con su evaluación experimental, obteniendo resultados satisfactorios. Finalmente, concluimos que nuestra estructura resulta una contribución al área de investigación, ya que proponemos un nuevo método de acceso espacio-temporal que responda eficientemente los principales tipos de consulta superando de esta manera a los métodos existentes en cuanto a la variedad de consultas que se pueden resolver eficientemente.

Referencias

[1] Antonin Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In ACM SIGMOD Conference on Management of Data, pages 47-57, Boston, 1984.
[2] M. A. Nascimento, J. R. O. Silva, and Y. Theodoridis. Evaluation of Access Structures for Discretely Moving Points. In Proc. of the Intl. Workshop on Spatio-Temporal Database Management, STDBM, Sept. 1999.
[3] Yannis Theodoridis, Michalis Vazirgiannis, and Timos K. Sellis. Spatio-temporal indexing for large multimedia applications. In International Conference on Multimedia Computing and Systems, pages 441-448, 1996.