Resultados Experimentales sobre
Nuevas Propuestas Heurísticas para Consultas a R-Tree

Edilma O. Gagliardi, Juan E. Gomez
Departamento de Informática, Universidad Nacional de San Luis
San Luis, Argentina
y
Gilberto Gutiérrez Retamal
Departamento de Auditoría e Informática, Universidad del Bío-Bío
Bío-Bío, Chile

Abstract

In this work an algorithm is presented to process consultations whose predicates establish restrictions on attributes derived from geometry (area or perimeter) of the objects. Our algorithm is based on the space access method R-Tree to evaluate the consultations and on the properties between MBR (Minimum Bounding Rectangle) and its area or perimeter of an object. We made a series of experiments that allowed to analyze the performance of the algorithm to process consultations with restrictions on the area of the objects. The experiments show that the algorithm has a good capacity of estimation of the answer of the consultation, since only overestimates 25,87% of the objects that really fulfill the properties of the consultation. Therefore, before the absence of an index for the derived attribute, our algorithm is a competitive alternative to process this type of consultations.

Key words: spatial index, spatial query, processing of spatial query, access multidimensional, spatial data bases.

Resumen

En este trabajo se presenta un algoritmo para procesar consultas cuyos predicados establecen restricciones sobre atributos derivados de la geometría (área o perímetro) de los objetos. Nuestro algoritmo se basa en el método de acceso espacial R-Tree para evaluar las consultas y en las propiedades entre el MBR (*Minimum Bounding Rectangle*) y su área o perímetro de un objeto. Nosotros realizamos una serie de experimentos que permitieron analizar el desempeño del algoritmo para procesar consultas con restricciones sobre el área de los objetos. Los experimentos muestran que el algoritmo tiene una buena capacidad de estimación de la respuesta de la consulta, ya que solamente sobreestima un 25.87% de los objetos que realmente cumplen con las propiedades de la consulta. Por lo tanto, ante la ausencia de un índice para el atributo derivado, nuestro algoritmo es una alternativa competitiva para procesar este tipo de consultas.

Palabras claves: índices espaciales, consultas espaciales, procesamiento de consultas espaciales, acceso multidimensional, bases de datos espaciales.

1 INTRODUCCION

En estos últimos tiempos ha surgido una creciente necesidad por poder almacenar y recuperar datos espaciales debido a que hay muchas aplicaciones que hacen uso de éste tipo de datos. Algunas aplicaciones están relacionadas con la superficie de la tierra (como los Sistemas de Información Geográfica (SIG)), también se usan en forma importante en las áreas de Diseño Asistido por Computador (CAD), diseño VLSI, visión por computador y robótica [1, 2]. También constituye un dominio interesante de aplicación el área de medicina y el de aplicaciones multimediales.

En la comunidad científica se ha destinado mucho esfuerzo para el procesamiento de las consultas espaciales, es decir, consultas en las cuales existe un predicado espacial. Por ejemplo, si contamos con un conjunto de objetos espaciales (polígonos) que representan los límites administrativos de las diferentes ciudades de un país, una consulta espacial puede ser la siguiente: *"Recuperar todas las ciudades que se encuentran dentro de una región específica"*. Existen varios predicados espaciales (intersección, disjoint, inside, etc) sin embargo, el mas utilizado es la intersección. Por otro lado, también existen varios tipos de consultas espaciales, destacando las consultas por rango espacial (window query), la reunión espacial, el vecino mas cercano, entre otras. El procesamiento de la mayoría de las consultas espaciales mencionadas anteriormente se apoya en el método de acceso espacial R-Tree [8] o en alguna de sus variantes y que se basa en realizar un agrupamiento de manera espacial de los objetos dentro de los bloques de disco lo que mejora en forma importante los tiempos de respuestas de las consultas espaciales.

Si bien es las consultas espaciales juegan un rol muy importante en las aplicaciones espaciales, existen consultas que es necesario resolver las cuales tienen como predicado relaciones sobre atributos derivados de la geometría de los objetos almacenados en la base de datos. Por ejemplo, *"Recuperar todas las ciudades de una área menor que 500 metros cuadrados de superficie o área"*. Podemos notar que si bien, esta consulta involucra los atributos espaciales de los objetos, el predicado se concentra en el área de que forman los polígonos de los objetos y no sobre su posición en el espacio. Este tipo de consultas también está siendo de interés [6, 7, 9] y por lo tanto es necesario contar con algoritmos eficientes para su evaluación. Una forma simple de procesar la consulta anterior es verificando secuencialmente si los objetos cumplen el predicado sobre el área, lo que implica accesar todos los objetos del conjunto (base de datos espacial). La otra solución es crear un índice para el área (por medio de un B-Tree por ejemplo). Esta solución, sin embargo, es peor pues la creación del índice también implica accesar todos los objetos. Podemos notar que, a pesar de existir un R-Tree para el conjunto de objetos espaciales, las dos soluciones descritas no lo utilizan. En este trabajo presentamos una solución (algoritmo) para evaluar el tipo de consultas anterior y que se basa en el método de acceso espacial R-Tree y en las propiedades entre el área y el MBR de un objeto.

La organización de este artículo es la siguiente: en la sección 2 se hace una breve descripción del método de acceso espacial R-Tree el cual es utilizado para procesar consultas espaciales y que también utilizaremos en nuestro algoritmo. En la sección 3 se presenta el algoritmo y en la sección 4 se evalúa en base a experimentos. Finalmente, en la sección 5 se entregan las conclusiones y los trabajos futuros o extensiones de nuestra propuesta.

2 R-TREE

R-Tree[8] 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, que sirve para almacenar objetos espaciales, por ejemplo puntos y regiones en el espacio. 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 del 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 del R-Tree contienen entradas de la forma <*I* ; *oid* > donde *I* es el menor rectángulo *n*-dimensional que contiene al objeto apuntado por *oid*, es decir, I = (I_1, I_2 ,...,*In*). Aquí *n* es el número de dimensiones e *I* es un intervalo cerrado [*a*,*b*] que describe los límites del objeto en la dimensión *i*. En caso de que un objeto espacial se extiende mas allá de los límites del espacio definido, entonces *I* puede tener uno o ambos puntos extremos igual a infinito. Los nodos no hoja contienen entradas de la forma <*I*;*pchild*> donde *pchild* es un puntero a un hijo del nodo y *I* contiene a todos los MBR's del nodo apuntado por *pchild*. 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 (ver Algoritmo 1).

Algoritmo 1

Algoritmo *Búsqueda* (*T*, *S*)

- Entrada: *T*, raíz de un R-Tree; *S*, rectángulo.
- Salida: *R* = {*R*1, *R*2, ..., *Rp*}, *Ri* es una tupla (*I*, *id-tupla*), tal que $Ri.I \cap S \neq \emptyset$.

If (T no es una hoja)

for (cada tupla *Ri* :(*I*, *child-pt*) en *T*)

if ($Ri.I \cap S \neq \emptyset$) Búsqueda (*Ri.child-pt*, *S*) *Finsi*

else

for (cada tupla *Ri*: (*I*, *id-tupla*))

if ($Ri.I \cap S \neq \emptyset$) $R \leftarrow R \cup Ri$; *retornar R* *Finsi*

Finsi

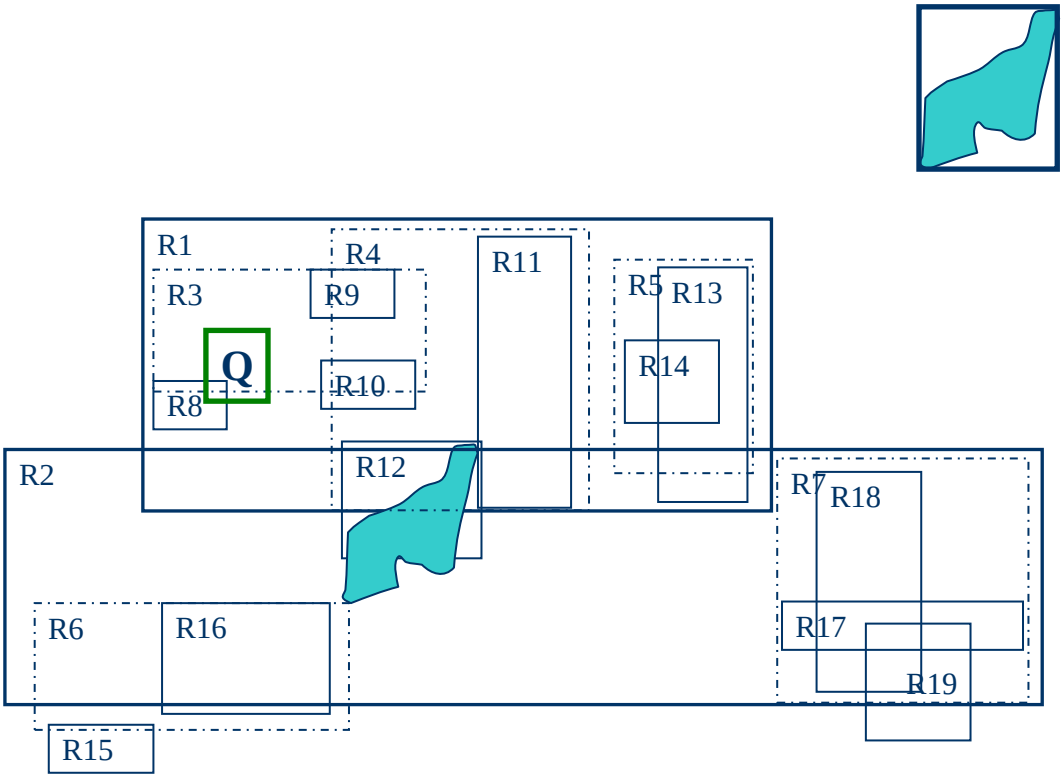

Figura 1: Disposición de los MBR's

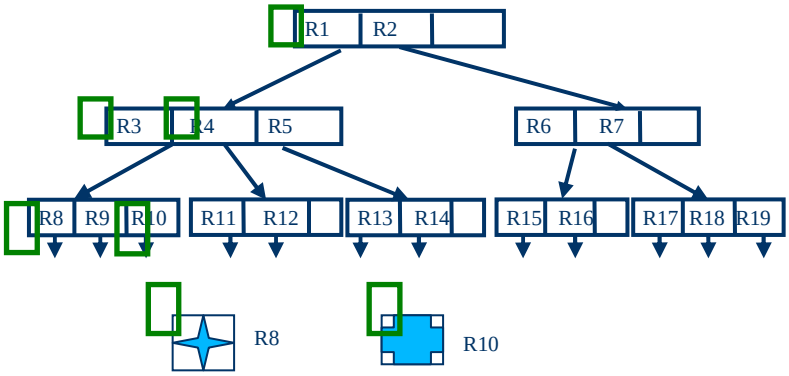

Figura 2: R-Tree

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[8]. Posteriores variaciones del 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.

3 NUESTRA PROPUESTA

La evaluación de una consulta espacial utilizando un índice espacial (típicamente creado con un R-Tree) se lleva a cabo en dos etapas [3, 4]: (1) filtrado y (2) refinamiento. En la etapa de filtrado se utiliza el índice para descartar los objetos que no cumplen con el predicado. Por ejemplo, si estamos interesados en una consulta sobre la intersección espacial de los objetos, si los MBRs (almacenados en el R-Tree) no se intersecan, entonces definitivamente los objetos tampoco lo harán y por lo tanto se descartan. Sin embargo, si los MBRs se intersecan, no sabemos si los objetos realmente se intersectarán. Para despejar esta incógnita es que se requiere la etapa de refinamiento la cual requiere contar con la geometría exacta de los objetos lo que implica recuperarla desde el almacenamiento secundario.

La idea detrás de nuestra propuesta, también sigue el paradigma de filtrado y refinamiento. Para la etapa de filtrado nosotros nos basamos en la propiedad que existe entre el MBR de un objeto y su área o superficie y que podemos ver en la figura 3. La propiedad es la siguiente: **"el área de un polígono es siempre menor que el área de su MBR"**. Esta propiedad nos permite descartar subárboles del R-Tree y de esta manera reducir el tamaño del conjunto de objetos que deben ser revisados en la etapa de refinamiento y por lo tanto reducir el costo para evaluar consultas sobre atributos derivados. Así si el área especificada en la consulta es mayor que el área de un MBR de un nodo interno del R-Tree, no se necesita seguir explorando el subárbol asociado al MBR, pues todos sus MBRs hijos tendrán también un área menor que la dada en la consulta y por lo tanto no existirán objetos que satisfagan el predicado de la consulta.

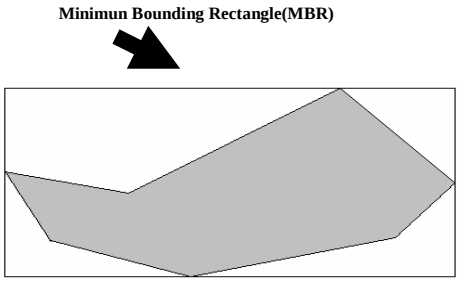

Figura 3: Relación entre área de un polígono y su MBR

Formalmente, entonces, nuestro problema es: dado un conjunto S de objetos espaciales del mismo tipo (polígonos) y Q (encontrar todas las comunas con un área mayor o igual a 500 Km^2) una consulta espacial cuyo predicado considera atributos derivados de la geometría de los objetos de S; y R el índice (R-Tree), evaluar Q sobre el conjunto S.

Tal como comentamos anteriormente, nuestra solución se apoya en el índice R y utilizar las aproximaciones MBR de los objetos espaciales para evaluar Q. Nuestro algoritmo es similar al descrito en [3], es decir, considera dos etapas. La primera etapa consiste en seleccionar el conjunto de objetos que puedan satisfacer Q. Esta selección se logra aplicando nuestra propiedad sobre los MBRs del R-Tree. Los objetos seleccionados en la etapa de filtrado forman un súper conjunto del

conjunto de objetos espaciales de la repuesta definitiva de Q. En la segunda etapa se accede a la geometría exacta de los objetos obtenidos en la primera etapa y se verifica si el objeto satisface el predicado de Q (ver Algoritmo 2).

Algoritmo 2

```
/* Dado un R-Tree con raíz T obtiene un conjunto de objetos que sobreestima el
 conjunto de la repuesta de una consulta espacial que contempla el atributo área */

ConjuntoOid  filtrar(R-tree T; float area) {
/*F conjunto de objetos espaciales cuyo MBR tiene área */
F = ∅ ;
if (T no es hoja) {
  for (cada entrada E € T)
    if (E.I.Area() ≥ area)
      F = F U Filtrar(E.pchild,area);
    else /* T no es una hoja */
      for (cada entrada E € T)
        if (E.I.Area() ≥ area)
          F = F U {E.Oid}
}
return F
}
```

4 EVALUACION EXPERIMENTAL

La implementación del algoritmo 2 sobre la estructura de índice R-Tree se evaluó experimentando con cuatro conjuntos de objetos espaciales de tipo polígonos y de tamaños 274, 1561, 4388 y 7060 respectivamente. Con cada uno de estos conjuntos se creó un R-Tree y luego ejecutamos consultas con restricciones sobre el área de los objetos. Se consideraron 11 áreas diferentes las que representan porcentajes del área total de los conjuntos que van desde 0,000001% hasta 0,1%. Por cada consulta, se midió la cantidad de objetos reales de la respuesta y la cantidad de objetos estimados por nuestro algoritmo. A partir de estos valores se obtuvo el porcentaje de sobreestimación que mide en cuanto se excede el conjunto estimado por el algoritmo con respecto del conjunto real de la respuesta. Por ejemplo, un porcentaje de un 5% de sobreestimación significa que el algoritmo devuelve un 5% más de objetos que la repuesta real. En la Tabla 1 se muestran algunos datos de los distintos conjuntos de objetos sobre los que se realizó la experimentación que

permiten visualizar el área o superficie de estos. Las áreas mínimas de los objetos de cada uno de los conjuntos son muy pequeñas en comparación con objetos con áreas excesivamente grandes, todo esto hace que la experimentación sea interesante debido a la diversidad de los objetos.

Tabla 1

Conjunto	Nro. Polígonos	Área Mínima	Área Máxima	Área Promedio	Área cubierta por todos los polígonos
1	274	16384	3.40209664E9	8.42186992E7	1.44421814E12
2	1561	32768	3.8535168E7	884573.32	1.81060633E12
3	4388	8192	3.071541248E9	1.07596148E7	1.6117878E12
4	7060	16384	4.126998528E9	1049940.22	2.37731316E12

En las tabla 2, tabla 3 y en la figura 4 podemos ver los resultados de nuestros experimentos. Es posible observar que en la medida en que disminuye el área de la consulta, la sobreestimación del algoritmo es mejor. Por ejemplo cuando el área es de un 0,00001% se llega a porcentajes de sobreestimación del 3.5% para el conjunto 1. En aquellas consultas con alta selectividad, el algoritmo presentó una sobreestimación del 23% aproximadamente. Sin embargo, cuando los porcentajes del área de las consultas son mayores a 0,00005% los valores de sobreestimación crecen. Sin embargo, y a pesar de este aumento, igual es conveniente utilizar nuestro algoritmo, pues evita una gran cantidad de accesos a memoria secundaria con lo que disminuye el tiempo de evaluación de la consulta. Por ejemplo, si nos concentramos en la tabla 1 y vemos los resultados para el conjunto 1 en las consultas con un 0,0005% del área, vemos que el algoritmo sobreestima alrededor del 34,92%, lo que significa que debemos revisar 22 objetos más que los que realmente forman la respuesta. Sin embargo, sin el algoritmo, se deberían procesar los 274 objetos, es decir, nuestro algoritmo sólo requiere revisar sólo un 34.92% más del conjunto de objetos que realmente cumplen con las propiedades de la consulta.

Tabla 2

Porcentaje del área total	Conjunto1= 274			Porcentaje del área total	Conjunto2= 1561		
	Real	Modelo	% Sobre estimación		Real	Modelo	% Sobre estimación
0,000001%	272	272	0,00	0,000001%	1473	1555	5,57
0,000005%	266	269	1,13	0,000005%	1207	1398	15,82
0,00001%	257	266	3,50	0,00001%	891	1107	24,24
0,00005%	170	207	21,76	0,00005%	245	428	74,69
0,0001%	122	166	36,07	0,0001%	147	260	76,87
0,0005%	63	85	34,92	0,0005%	25	79	216,00
0,001%	47	66	40,43	0,001%	8	38	375,00
0,005%	30	38	26,67	0,005%	0	1	-
0,01%	23	30	30,43	0,01%	0	0	-
0,05%	8	15	87,50	0,05%	0	0	-
0,1%	4	11	175,00	0,1%	0	0	-

Tabla 3

Porcentaje del área total	Conjunto 3 = 4388			Porcentaje del área total	Conjunto 4 = 7060		
	Real	Modelo	% Sobre estimación		Real	Modelo	% Sobre estimación
0,000001%	4299	4364	1,51	0,000001%	6178	7016	13,56
0,000005%	4137	4285	3,58	0,000005%	3890	4611	18,53
0,00001%	3947	4223	6,99	0,00001%	1951	2914	49,36
0,00005%	2080	2906	39,71	0,00005%	341	798	134,02
0,0001%	1320	2067	56,59	0,0001%	150	441	194,00
0,0005%	422	791	87,44	0,0005%	10	83	730,00
0,001%	277	493	77,98	0,001%	2	31	1450,00
0,005%	88	189	114,77	0,005%	1	2	100,00
0,01%	51	115	125,49	0,01%	1	1	0,00
0,05%	9	32	255,56	0,05%	1	1	0,00
0,1%	3	12	300,00	0,1%	1	1	0,00

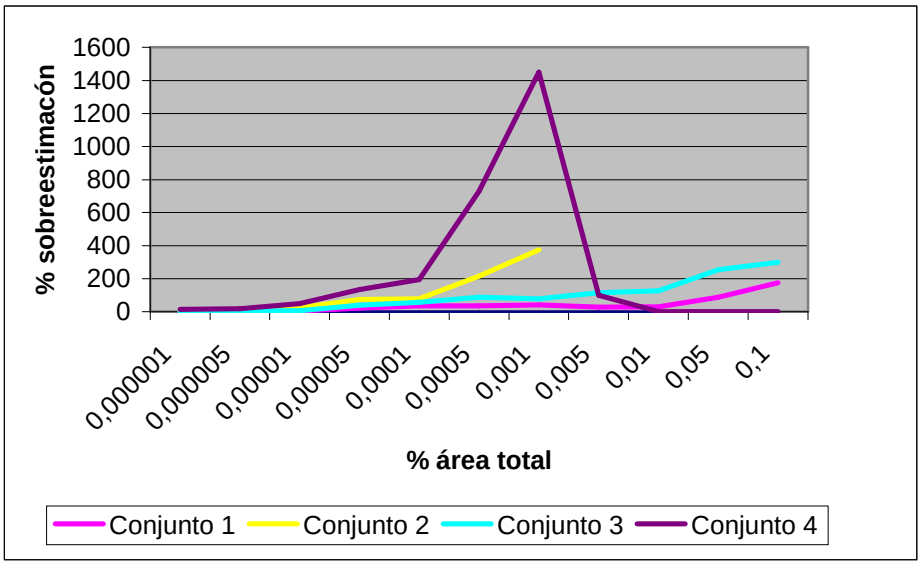

Figura 4: % del área total vs % estimación

Analizando los objetos de cada conjunto se pudo observar que cuando los objetos tenían una mayor cantidad de puntos que lo conformaban el porcentaje de estimación también mejoraba ya que el MBR se ajustaba mejor a la forma del objeto.

Los resultados mostrados en la experimentación arrojaron un porcentaje de estimación promedio de *25.87 %* para consultas que consideran porcentajes del área entre 0,000001% y 0,00005%, lo cual permite inferir que es conveniente procesar consultas espaciales sobre atributos derivados utilizando la aproximación MBR en la etapa de filtrado y por lo tanto, en ausencia de índice para el atributo derivado, el uso de un R-Tree para estimar el conjunto de objetos aparece como una buena alternativa.

5 Trabajo Futuro y Conclusiones

Nuestra propuestas presenta ventajas al procesar consultas sobre atributos derivados utilizando R-Tree, alcanzando solamente un 25.87% de sobreestimación, lo que la convierte en un enfoque interesante para procesar atributos derivados de los objetos espaciales. También es interesante constatar que, a pesar de que nuestro algoritmo presentó una sobreestimación alta para consultas con baja selectividad, es conveniente utilizarlo frente a la ausencia de un índice ad-hoc para el atributo derivado.

Como trabajo futuro pretendemos extender nuestra idea a otras propiedades entre la geometría exacta de los objetos y que puedan ser útil para evaluar para evaluar consultas sobre atributos derivados; por ejemplo, consultas con predicados de distancias sobre objetos de tipo polilíneas. También pretendemos diseñar algoritmos que combinen predicados sobres los atributos derivados de los objetos espaciales con atributos espaciales. Por ejemplo, *"los objetos que se intersecan con el objeto Q y cuya área es menor que 10 unidades"*. Finalmente, planteamos definir un modelo de costo que nos permita predecir el rendimiento de las consultas utilizando nuestro algoritmo.

Referencias

[1] Ralf Guting. *An Introduction to espatial database system.* In VLDB Journal, volume 3, pages 357-399, 1994.

[2] King-Ip Lin, H.V. Jagadish, an Christos Faloutsos. *The TV-Tree: An index structure for high-dimensional data*. VLDB Journal: Very Large Data Bases, 3(4):517-542, 1994.

[3] Thomas Brinkhoff, Hans-Peter Kriegel, Ralf Schneider and Bernhard Seeger. *Multi-step processing of spatial joins*. In ACM SIGMOD Conference on Management of Data, pages 197-208, Minesota, USA, 1994.

[4] Thomas Brinkhoff, Hans-Peter Kriegel and Bernhard Seeger. *Efficient processing of spatial jons using r-trees*. In ACM SIGMOD Conference on Management of Data, pages 237-246, Washington, DC, USA, 1993. ACM.

[5] Yun-Wu Huang, Ning Jing, and Elke Rundensteiner. *Spatial joins using r-tree: Breadth-first traversal with global optimizations*. IN 23rd Conference on Very Large Data Bases, pages 396-405, Athens, Greece, 1997.

[6] Ho-Hyun Park,Chan-Gun Lee, Yong-Ju Lee, and Chin-Wan Chung. *Early separation of filter and refinement steps in spatial query optimization*. In Database System for Advanced Applications, pages 161-168, 1999.

[7] Ho-Hyun Park, Yong-Ju Lee, and Chin-Wan Chung. *Spatial query optimization utilizing early separated filter and refinament strategy*. Information Systems, 25(1):1-22, 2000.

[8] Antonin Guttman. *R-Tree: A Dinamic Index Structure for Spatial Searching*. In ACM SIGMOD Conference on Management of Data, pages 47-57, Boston, 1984.

[9]Gilberto Gutiérrez. *Procesamiento de Consultas Espaciales con Restricciones sobre Atributos Derivados de la Geometría de Objetos Espaciales*. Bío-Bío Chile.