Índice Métrico-Temporal Event-FHQT

Anabella De Battista , Andrés Pascal
Dpto de Sistemas de Información
Universidad Tecnológica Nacional
Fac. Reg. Concepción del Uruguay
Entre Ríos, Argentina
{debattistaa, pascala}@frcu.utn.edu.ar

Norma Edith Herrera
Departamento de Informática
Universidad Nacional de San Luis
San Luis, Argentina
nherrera@unsl.edu.ar

Gilberto Gutierrez
Facultad de Ciencias Empresariales
Universidad del Bio-Bio
Chillán, Chile
ggutierr@ubiobio.cl

Resumen

Como respuesta a la necesidad de procesar consultas por similitud que además requieren tomar en cuenta el aspecto temporal, surgió el modelo de base de datos métrico-temporal. Dicho modelo combina los espacios métricos con las bases de datos temporales y permite la búsqueda de objetos similares a uno dado, en un intervalo o instante de tiempo. En este artículo presentamos el Event-FHQT, una nueva estructura basada en eventos que permite reducir los costos para el tipo de consulta antes mencionada. Además verificamos de manera experimental la eficiencia de esta estructura para un determinado conjunto de datos.

Palabras clave: consulta métrico-temporal, espacio métrico, base de datos temporal, búsqueda por similitud

1. Introducción

Las operaciones de búsquedas en una base de datos requieren de algún soporte y organización especial a nivel físico. En el caso de las bases de datos clásicas, la organización de la información se basa en el concepto de búsqueda exacta sobre datos estructurados. Esto significa que la información se organiza en registros con campos completamente comparables. Una búsqueda en la base retorna todos aquellos registros cuyos campos coinciden con los aportados en la consulta (búsqueda exacta). Otra característica importante de las bases de datos clásicas es que capturan sólo un estado de la realidad modelizada, usualmente el más reciente. Por medio de las transacciones, la base de datos evoluciona de un estado al siguiente descartando el estado previo.

Actualmente las bases de datos han incluido la capacidad de almacenar otros tipos de datos tales como imágenes, sonido, texto, video, datos geométricos, etc. La problemática de almacenamiento y búsqueda en estos tipos de base de datos difiere notablemente de las bases de datos clásicas en tres aspectos: primero los datos generalmente son no estructurados, esto significa que es imposible

organizarlos en registros y campos, segundo la búsqueda exacta carece de interés y tercero resulta de interés mantener todos los estados de la base de datos y no sólo el más reciente a fin de poder consultar el instante o intervalo de tiempo de vigencia de dichos objetos. Como solución a esta problemática surgen modelos que permiten procesar esta clase de datos. Entre estos nuevos modelos encontramos los siguientes:

Espacios métricos [2, 5, 7, 8, 9, 13], que permiten almacenar objetos no estructurados y realizar búsquedas por similitud sobre los mismos. Un espacio métrico es un par (U, d) donde U es un universo de objetos y $d : U \times U \rightarrow R^+$ es una función de distancia definida entre los elementos de U que mide la similitud entre ellos. Una de las consultas típicas en este nuevo modelo de bases de datos es la búsqueda por rango que consiste en recuperar los objetos de la base de datos que se encuentren como máximo a distancia r de un elemento q dado.

Bases de datos temporales [15, 12], que incorporan al tiempo como una dimensión, por lo que permiten asociar tiempos a los datos almacenados. Existen tres clases de bases de datos temporales en función de la forma en que manejan el tiempo: *de tiempo transaccional (transaction time)*, donde el tiempo se registra de acuerdo al orden en que se procesan las transacciones; *de tiempo vigente*, que almacenan el momento en que el hecho ocurrió en la realidad (puede no coincidir con el momento de su registro) y *bitemporales*, que integran la dimensión transaccional y la dimensión vigente a través del versionado de los estados, es decir, cada estado se modifica para actualizar el conocimiento de la realidad pasada, presente o futura, pero esas modificaciones se realizan generando nuevas versiones de los mismos estados.

Bases de datos espacio-temporales [11] que mantienen datos sobre el pasado y el presente y pueden, en algunos casos, realizar predicciones sobre el futuro. Las consultas típicas son de dos clases: *time slice queries* y *time interval (o windows) queries*. Las primeras consultas se realizan sobre un momento dado, como por ejemplo ”buscar todos los objetos que están en un área en un instante determinado”, mientras que las segundas consultan un intervalo de tiempo, ”buscar todos los objetos que crucen un área entre el momento t_1 y el momento t_2”. Algunas consultas sólo tienen sentido sobre el pasado, otras sólo sobre el presente, otras sobre el futuro, y otras sobre cualquiera de los tres.

Bases de datos métrico-temporales [3, 4, 14], que permiten almacenar objetos no estructurados con tiempos de vigencia asociados y realizar consultas por similitud y por tiempo en forma simultánea. Un ejemplo de aplicación de este último modelo podría ser el siguiente: dada una base de datos de huellas digitales de las personas que ingresan a un museo con la fecha y hora del ingreso, *determinar si una persona (en base a su huella digital) estuvo en el museo en una fecha dada*.

En este artículo se presenta un nuevo método de acceso métrico-temporal, el *Event-FHQT*, que permite resolver esta última clase de consultas con eficiencia. El mismo está organizado de la siguiente manera: en la Sección 2 se expone una breve reseña del trabajo relacionado y se define el modelo de bases de datos métrico-temporales. En la Sección 3 se presenta el *Event-FHQT*. En la Sección 4 se muestra la evaluación experimental de este índice. Por último, en la Sección 5 se plantean las conclusiones y el trabajo futuro.

2. Trabajo relacionado

Dado que el índice métrico Fixed-Height Queries Tree (FHQT) es la base de desarrollo de nuestra propuesta comenzaremos dando una breve reseña del marco teórico de este índice, para luego definir los conceptos fundamentales en el modelo métrico-temporal.

2.1. El Modelo de Espacios Métricos

La problemática de búsquedas por similitud en bases de datos no tradicionales puede formalizarse por medio del modelo de *espacios métricos*. Un espacio métrico es un par $(\mathcal{X}, d)$, donde $\mathcal{X}$ es un conjunto de objetos y $d : \mathcal{X} \times \mathcal{X} \rightarrow R^+$ es una función de distancia que modela la similitud entre los elementos de $\mathcal{X}$. La función d cumple con las propiedades características de una función de distancia: $\forall x, y \in \mathcal{X},\ d(x, y) \geq 0$ (positividad), $\forall x, y \in \mathcal{X},\ d(x, y) = d(y, x)$ (simetría), $\forall x, y, z \in \mathcal{X},\ d(x, y) \leq d(x, z) + d(z, y)$(desigualdad triangular). La base de datos es un conjunto finito $\mathcal{U} \subseteq \mathcal{X}$.

Una de las consultas típicas que implica recuperar objetos similares de una base de datos es la *búsqueda por rango*, que denotaremos con $(q, r)_d$. Dado un elemento de consulta q, al que llamaremos *query* y un radio de tolerancia r, una búsqueda por rango consiste en recuperar aquellos objetos de la base de datos cuya distancia a q no sea mayor que r, es decir, $(q, r)_d = \{u \in \mathcal{U} : d(q, u) \leq r\}$.

Una forma trivial de resolver este tipo de búsquedas es examinando exhaustivamente la base de datos, es decir, comparando cada elemento de la base datos con la query. En general, esto resulta demasiado costoso en aplicaciones reales y no es posible realizarlo. Se han logrado avances importantes sobre espacios métricos en torno al concepto de construir un *índice*, es decir, una estructura de datos que permita reducir el tiempo necesario para resolver una consulta. En este tiempo T influyen tres factores; por un lado tenemos la cantidad de evaluaciones de la función de distancia d que se realizaron durante el proceso de búsqueda; por otro lado tenemos una cierta cantidad de operaciones adicionales que implican un tiempo extra de CPU; finalmente, tenemos un tiempo de I/O determinado por la cantidad de accesos a memoria secundaria, si es que fuera necesario; en símbolos:

T= #evaluaciones de $d\times$ complejidad(d) + tiempo extra de CPU + tiempo de I/O

En [9] se presenta un desarrollo unificador de las soluciones existentes en la temática. En dicho trabajo se muestra que todos los enfoques para la construcción de índices en espacios métricos consisten en particionar el espacio en clases de equivalencia e indexar las clases de equivalencia. Luego, durante la búsqueda, usando el índice y la desigualdad triangular, se descartan algunas clases y se busca exhaustivamente en las restantes. La diferencia entre los distintos algoritmos de indexación radica en cómo construyen esta relación de equivalencia. Básicamente se pueden distinguir dos grupos:

Algoritmos basados en pivotes: estos algoritmos construyen la relación de equivalencia tomando la distancia de los elementos de la base a un conjunto preseleccionado de elementos denominados *pivotes*; sea $\{p_1, p_2, \ldots, p_k\}$ el conjunto de pivotes, dos elementos x e y son equivalentes si y solo si están a la misma distancia de todos los pivotes, es decir, $d(x, p_i) = d(y, p_i),\ \forall i = 1 \ldots k$. Durante la indexación, se seleccionan los k pivotes y se le asigna a cada elemento x de la base de datos el vector o firma $(d(x, p_1), d(x, p_2), \ldots, d(x, p_k))$. Ante una búsqueda $(q, r)_d$, se usa la desigualdad triangular junto con los pivotes para filtrar elementos de la base de datos sin medir su distancia a la query q. Para ello se computa la distancia de q a cada uno de los pivotes p_i, y luego se descartan todos aquellos elementos a, tales que para algún pivote p_i se cumple que $\mid d(q, p_i) - d(a, p_i) \mid > r$. Los elementos no descartados forman parte de una *lista de candidatos*,

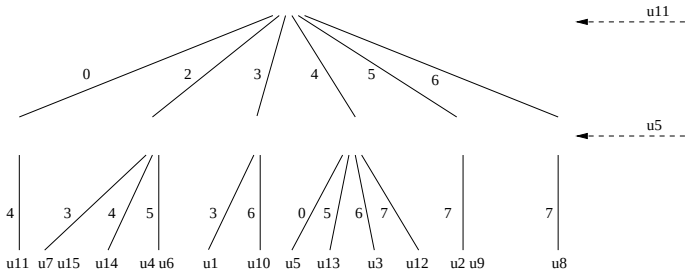

Figura 1: Un ejemplo de un FHQT sobre un conjunto de 15 elementos

que posteriormente se comparan directamente con la query q para decidir si forman o no parte de la respuesta.

Algoritmos basados en particiones compactas: en este caso la relación de equivalencia se construye teniendo en cuenta la cercanía de los elementos a un conjunto preseleccionado de elementos denominados *centros*; dos elementos son equivalentes si tienen al mismo centro c como su centro más cercano. Durante la indexación, seleccionan un conjunto de *centros* $\{c_1, c_2, \ldots, c_k\}$ y dividen el espacio asociando a cada centro c_i el conjunto de puntos que tiene a c_i como su centro más cercano. Existen muchos criterios posibles para descartar zonas durante una búsqueda. Los dos más populares son *criterio del hiperplano* y *criterio del radio de cobertura* [9].

El Fixed-Height FQT (FHQT), presentado en [2], es un índice basado en pivotes. Es una variante del Fixed Queries Tree (FQT) [1] en donde todas las hojas se encuentran a la misma altura. Originalmente estas estructuras fueron propuestas para funciones de distancias discretas, pero se pueden adaptar a distancias continuas discretizando los valores de las mismas [9, 10].

El árbol se construye a partir de un elemento p (pivote) que puede ser elegido arbitrariamente, o mediante algún procedimiento de selección de pivotes [6], del universo U. Para cada distancia i se crea el conjunto C_i formado por todos aquellos elementos de la base de datos que están a distancia i de p. Luego, para cada C_i no vacío se crea un hijo del nodo correspondiente a p, con rótulo i, y se construye recursivamente un FHQT teniendo en cuenta que todos los subárboles del mismo nivel usarán el mismo pivote como raíz. Este proceso recursivo, en el caso del FQT se repite hasta que queden menos de b elementos, los cuales se almacenan en una hoja. En el caso del FHQT se continúa hasta lograr que todas las hojas tengan menos de b elementos y estén en un mismo nivel.

La figura 1 muestra un ejemplo de un FHQT con dos pivotes sobre un conjunto de 15 elementos. Ante una consulta $(q, r)_d$, se comienza por la raíz y se descartan todas aquellas ramas con rótulo i tal que $i \notin [d(p, q) - r, d(p, q) + r]$ siendo p el pivote utilizado en la raíz. La búsqueda continúa recursivamente en todos aquellos subárboles no descartados, utilizando el mismo criterio.

2.2. Bases de Datos Métrico-Temporales

Este tipo de bases de datos permite realizar búsquedas sobre objetos no estructurados que tienen un intervalo de vigencia asociado y que no poseen un identificador que se pueda utilizar como clave de búsqueda, por lo cual tiene sentido realizar consultas por similitud y considerar también el aspecto temporal.

Formalmente, sea O el universo de objetos válidos, un *Espacio Métrico-Temporal* [3, 4, 14] es un par (U, d), donde $U = O \times N \times N$ y la función de distancia d, es de la forma $d : O \times O \to R^+$. Cada elemento $u \in U$ es una 3-upla (obj, t_i, t_f), donde obj es un objeto y $[t_i, t_f]$ es el intervalo de vigencia

de obj. La función d es una métrica que mide el grado de similitud entre dos objetos. Los problemas que se resuelven a través de consultas métrico-temporales se caracterizan de la siguiente manera:

- No tiene sentido realizar búsquedas exactas, ya que los objetos no poseen claves que los identifiquen, o si existen, no se conocen en el momento de la consulta.
- Los objetos tienen instantes o intervalos de vigencia que representan los períodos en los cuales son válidos para la realidad.
- En las búsquedas se requiere consultar por similitud y tiempo simultáneamente.
- La búsqueda secuencial no es una opción factible debido a su costo.

Una vez que definimos el modelo métrico-temporal estamos en condiciones de definir cómo se realizan las búsquedas en este modelo. Sea $X \in U$ el conjunto finito sobre el que se realizan las búsquedas, una consulta métrico temporal se denota mediante la 4-upla $(q, r, t_{iq}, t_{fq})_d$, y se define formalmente de la siguiente manera:

$$(q, r, t_{iq}, t_{fq})_d = o/(o, t_{io}, t_{fo}) \in X \wedge d(q, o) \leq r \wedge (t_{io} \leq t_{fq}) \wedge (t_{iq} \leq t_{fo})$$

La variable r es el radio de búsqueda y representa el grado de similitud requerido, y q es el objeto que se consulta. Respecto al tiempo, esta clase de consulta tiene éxito sólo para los objetos cuyo intervalo se superpone en algún punto con el intervalo consultado. En el caso de una consulta instantánea $t_{iq} = t_{fq}$.

Una forma trivial de resolver una consulta métrico-temporal es construir un índice métrico agregándole a cada objeto el intervalo de tiempo de vigencia del mismo. Luego, ante una consulta $(q, r, t_{iq}, t_{fq})_d$, primero se realiza la búsqueda $(q, r)_d$ sobre el índice métrico y luego se procede a buscar secuencialmente sobre el conjunto de objetos obtenido en el paso anterior, teniendo en cuenta la restricción temporal.

La desventaja que tiene esta solución es que no se usa la componente temporal para filtrar la búsqueda en el índice métrico. Una mejor estrategia sería que durante el proceso de búsqueda se utilice tanto la componente métrica como la componente temporal para descartar elementos.

Se han propuestos dos índices para procesar eficientemente consultas métrico-temporales:

FHQT-Temporal. Este índice, presentado en [14], es una adaptación del FHQT en la que se agrega un intervalo de tiempo en cada nodo del árbol. Este intervalo representa el período máximo de vigencia para todos los objetos del subárbol cuya raíz es dicho nodo. En cada nodo hoja, este intervalo es el período total de vigencia de los objetos que contiene. Para cada nodo interior, el intervalo se calcula tomando el tiempo inicial mínimo, y el tiempo final máximo de sus hijos. Cuando se realiza una consulta métrico-temporal se procede de la siguiente manera: en cada nivel del árbol se filtran los subárboles hijos por el intervalo de tiempo de la consulta y luego de acuerdo a la distancia entre la consulta y el pivote. Al llegar al último nivel, se realiza una búsqueda secuencial sobre las hojas que no fueron descartadas seleccionando los objetos que cumplen con las condiciones temporales y de similitud.

Historical-FHQT. Consiste en una lista de instantes válidos donde cada uno contiene un FHQT correspondiente a todos los objetos vigentes en dicho instante [4]. Esta estructura es eficiente en bases de datos métrico-temporales en las que los objetos tienen vigencia en un solo instante de tiempo. Los FHQT tienen distintas profundidades en función de la cantidad de elementos que deban indexar. La

cantidad de pivotes utilizada en un árbol se calcula como $\lceil log_2(|o_i|) \rceil$ donde $|o_i|$ es la cantidad de objetos vigentes en el instante i. De esta manera se evita que haya árboles con mayor profundidad de la necesaria, con el fin de que la estructura no tenga un costo excesivo en almacenamiento. Las consultas métrico-temporales se efectúan de la siguiente manera: en primer lugar se seleccionan los instantes incluidos en el intervalo de consulta. Luego se realizan consultas por similitud usando cada uno de los FHQT correspondientes, y finalmente se unen los conjuntos resultantes.

3. Un nuevo índice métrico-temporal: el Event-FHQT

El Event-FHQT combina ideas de la estructura de acceso espacio-temporal SEST L [11] y de la estructura métrica FHQT [2] para responder consultas métrico-temporales. Este índice, que está orientado a eventos, mantiene una lista de intervalos de tiempo donde cada intervalo contiene un snapshot (instantánea) que representa a todos los objetos vigentes en el primer instante de tiempo del intervalo, y listas de eventos que indican los cambios que se produjeron entre dos snapshots. Este diseño presenta una ventaja sobre el índice métrico-temporal Historical-FHQT [4] ya que, a diferencia de este último, no necesita duplicar todos los objetos que están vigentes en más de un instante de tiempo. El Event-FHQT en la mayoría de los casos sólo registra los cambios (entradas y salidas) de los objetos. También tiene la ventaja de ser más estable que el FHQT-Temporal [14], que disminuye su eficiencia cuando los intervalos de objetos similares (que se encuentran en las mismas hojas) son distantes, lo que produce que los intervalos de los nodos antecesores sea muy grande y por lo tanto, que no tenga buena capacidad de filtrado por tiempo.

3.1. Estructura

El Event-FHQT está compuesto por una Línea del Tiempo representada por una lista de intervalos consecutivos de tamaño fijo. Cada intervalo contiene un FHQT que representa las relaciones de similitud entre los objetos vigentes en el intervalo. El FHQT se construye a partir de los objetos del primer instante, y luego se actualiza con las entradas y salidas de objetos durante todo el intervalo. Las hojas del FHQT contienen listas de eventos que registran estas entradas y salidas. El Event-FHQT posee un parámetro de configuración que especifica el tamaño de los intervalos. Formalmente, un Event-FHQT es una 3-upla (l_p, tam, l_i) donde

- $l_p = (p_1, p_2, \ldots, p_k)$ es la lista de pivotes utilizados en la construcción de los FHQT, donde k es la cantidad de pivotes.
- tam es el tamaño de cada intervalo
- l_i es una lista de intervalos $(I_1, I_2, \ldots, I_n)$ donde cada intervalo I es de la forma $(t_i, t_f, FHQT)$, siendo t_i y t_f los tiempos inicial y final del intervalo respectivamente. El t_i de un intervalo es igual al $t_f + 1$ del intervalo anterior. El tercer elemento del intervalo es un FHQT modificado que contiene todos los objetos vigentes en el intervalo. Se define de la misma manera que un FHQT, a excepción de sus hojas, que contienen listas de eventos ordenadas en forma ascendente por instante de tiempo. Cada evento es de la forma *(instante, objeto, tipoDeEvento)*. El campo *tipoDeEvento* puede tomar los valores *in*, *out* o *stay* . El evento *in* significa que el objeto comenzó a estar vigente en el instante asociado al evento, *out* indica que dejó de estar vigente en dicho instante, y *stay* representa que el objeto ya existía en el intervalo anterior y permanece vigente al menos en el primer instante del intervalo actual. Este evento se utiliza únicamente por razones de eficiencia ya que de no hacerlo, para conocer los objetos vigentes en un intervalo

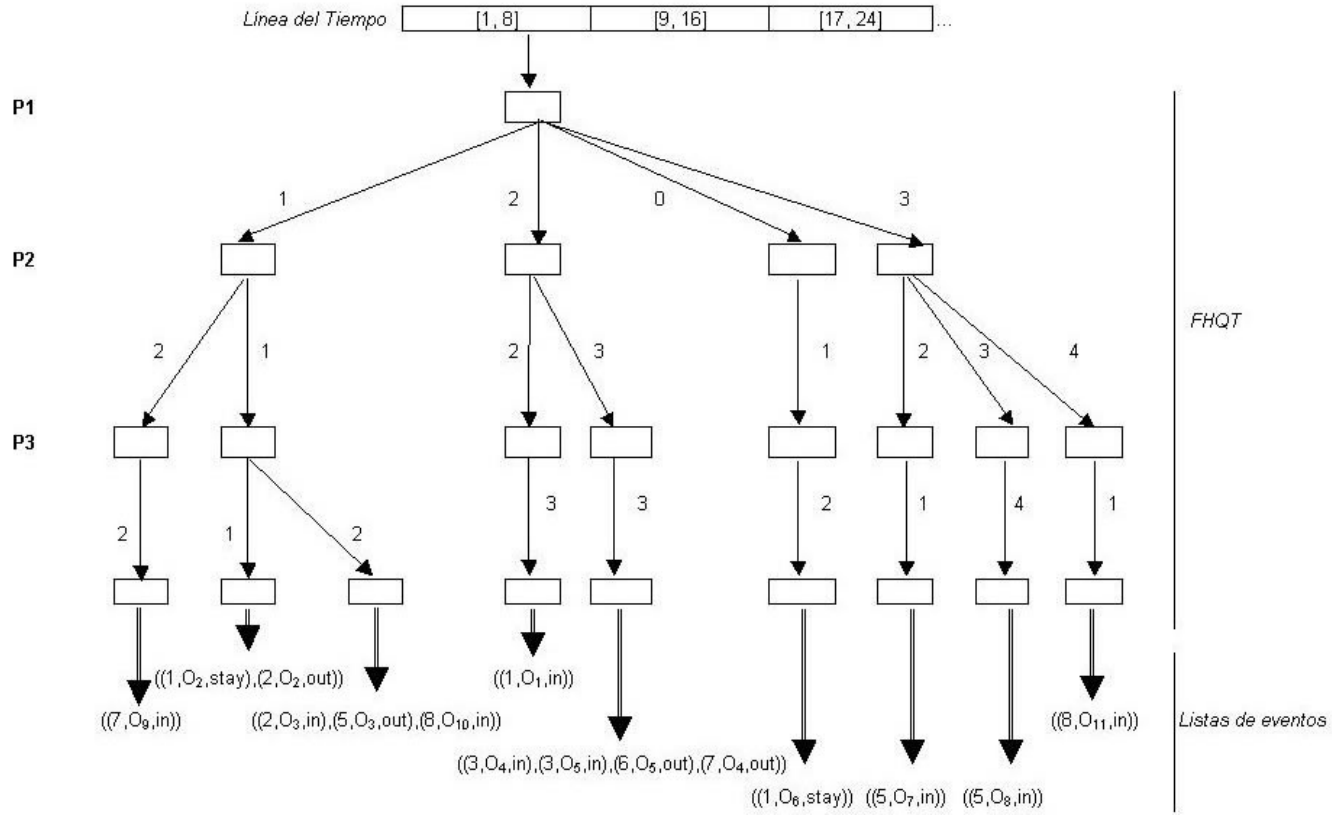

Figura 2: Un ejemplo de un Event-FHQT

habría que recorrer todos los FHQT de los intervalos anteriores. El instante correspondiente a un evento *stay* siempre es el primero del intervalo.

La estructura permite altas y bajas, tanto históricas como de objetos en nuevos intervalos. Las altas de objetos que poseen un intervalo de vigencia que abarca más de un intervalo de la línea del tiempo, se deben realizar dividiendo dicho intervalo en subintervalos que coincidan o estén incluidos dentro de los intervalos definidos en el Event-FHQT, para luego dar de alta el objeto en los FHQT correspondientes a cada uno de ellos. El costo del alta es entonces, el costo del alta a un FHQT multiplicado por la cantidad de intervalos afectados. En la Figura 2 se muestra un ejemplo de un Event-FHQT. Por razones de espacio sólo se muestra el FHQT correspondiente al primer intervalo.

3.2. Consultas

Una consulta a un Event-FHQT se define como $(q, r, t_{iq}, t_{fq})_d$, donde q es el objeto consultado, r el radio de similitud, $[t_{iq}, t_{fq}]$ el intervalo que se consulta, y d es la función de distancia. La consulta se realiza conceptualmente en cuatro pasos:

1. Determinar qué intervalos de la lista se superponen con el intervalo de consulta (primer filtro temporal).

2. Para cada intervalo, realizar la consulta por similitud a los FHQT correspondientes, hasta el nivel de las hojas (primer filtro por similitud).

3. Recorrer las listas de eventos contenidas en las hojas, determinando cuales objetos tienen superposición temporal con la consulta (segundo filtro temporal).

Rama	Lista de Eventos
$1-2-2$	$((7, O_9, in))$
$1-1-1$	$((1, O_2, stay), (2, O_2, out))$
$1-1-2$	$((2, O_3, in), (5, O_3, out), (8, O_10, in))$
$3-2-1$	$((5, O_7, in))$
$2-2-3$	$((1, O1, in))$

Figura 3: Listas de eventos correspondientes a las ramas seleccionadas

4. Unir los conjuntos resultantes -para eliminar objetos duplicados- y realizar la comparación final por similitud respecto a la consulta, obteniendo así el conjunto de objetos definitivo.

Por ejemplo, ante la consulta $(q, 1, 3, 10)_d$, siendo $(2, 1, 2)$ la firma de q, se procede de la siguiente manera: en primer lugar se seleccionan los intervalos $[1, 8]$ y $[9, 16]$, ya que ambos poseen superposición temporal con el intervalo de consulta $[3, 10]$. Luego para cada intervalo se consulta el FHQT asociado. En el primer intervalo, las hojas que cumplen con la restricción de similitud son las correspondientes a las distancias (nombradas desde la raíz hacia abajo) 1 2 2, 1 1 1, 1 1 2, 3 2 1 y 2 2 3. Por ejemplo, en el primer caso, la evaluación de 1 2 2 con la firma de la consulta satisface las condiciones de similitud $(2-1 \leq 1 \leq 2+1)$, $(1-1 \leq 2 \leq 1+1)$ y $(2-1 \leq 2 \leq 2+1)$. Aplicando el mismo razonamiento para las demás ramas, se obtiene la lista de eventos mostradas en la Figura 3.

Posteriormente se recorren estas listas para determinar cuales objetos estuvieron vigentes dentro del intervalo de consulta. Ante un evento *in* o *stay* correspondiente a un instante anterior al tiempo final de la consulta, el objeto asociado se ingresa al conjunto de resultados. Si el evento es *out* y el instante es menor al tiempo inicial de la consulta, el objeto se da de baja del conjunto. Como el intervalo de consulta es $[3, 10]$, los objetos O_9 (ingresa en 7), O_3 (ingresa en 2 y permanece hasta 4 inclusive), O_{10} (ingresa en 8), O_7 (ingresa en 5) y O_1 (ingresa en 1 y permanece durante todo el intervalo, ya que no tiene evento *out* en este intervalo) forman parte del conjunto de resultados candidatos. Note que el objeto O_2 tiene asociado un evento *stay*, lo que indica que estaba vigente en el intervalo anterior. En este caso no forma parte del resultado porque tiene un evento *out* anterior al tiempo inicial de la consulta. Seguidamente se realiza la unión entre este conjunto y el conjunto resultante de procesar el intervalo $[9, 16]$ de la misma manera. Así se evita evaluar la función de distancia más de una vez para un mismo objeto. Por último, se calcula la distancia entre cada objeto candidato y la consulta, y se descartan aquellos con distancia mayor al radio de búsqueda, obteniéndose el resultado final. Este diseño permite que se descarten en primer lugar intervalos -y por lo tanto FHQTs- completos y luego, dentro de cada intervalo, subárboles completos. En la Figura 4 se muestra el pseudocódigo correspondiente al algoritmo de consulta del Event-FHQT, que se descompone en un programa principal, y una rutina auxiliar que realiza las consultas a los FHQT.

4. Resultados experimentales

A fin de determinar experimentalmente la eficiencia de esta nueva estructura ante consultas métrico-temporales, se realizaron pruebas sobre una base de datos de 18.000 objetos. Cada objeto representaba una imagen a través de un vector de características de 760 dimensiones. Además los objetos tuvieron un intervalo de tiempo asociado indicando su período de vigencia, dentro del rango total $[1, 1000]$. La amplitud máxima de dichos intervalos fue 100 instantes, y la densidad promedio, 900 objetos por instante de tiempo.

```
ConsultarEventFHQT ( (q, r, t_iq, t_fq)_d )
 1. F = firmaDe(q, pivotes)
 2. C = ∅
 3. For all{ intervalo k del Event-EFHQT / (t_iq ≤ t_fk) and (t_fq ≥ t_ik) }
 4.    C = C ∪ ConsultarFHQT(FHQT_k, q, r, t_iq, t_fq, 1, F)
 5. end For
 6. resultado = ∅
 7. For all (o ∈ C )
 8.     If d(q,o) ≤ r then
 9.       resultado = resultado ∪ {o}
 10. end For
 11. return resultado

ConsultarFHQT (nodoActual, q, r, t_iq, t_fq, n, F)
 1. res = ∅
 2. If nodoActual es hoja then
 3.     For i=1 to cantidadDeEventos(nodoActual)
 4.       Sea e_i el iésimo evento de nodoActual
 5.         If instante(e_i) ≤ t_fq then
 6.           If tipoDeEvento(e_i) ∈ {in, stay} then
 7.             res = res ∪ {objeto(e_i)}
 8.           else If (tipoDeEvento(e_i) ∈ {out}) and (instante(e_i) < t_iq) then
 9.                 res = res - {objeto(e_i)}
 10.     end For
 11. else
 12.     For all (hijo h_i de nodoActual)
 13.       If |F(n) - d_i| ≤ r then
 14.         res = res ∪ ConsultarFHQT (h_i, q, r, t_iq, t_fq, n+1, F)
 15.     end For
 16. return res
```

Figura 4: Pseudocódigo del algoritmo de consulta al Event-FHQT

La función de distancia utilizada fue la distancia coseno discretizada [10]. En estas pruebas sólo se tomó en cuenta como variable de costo la función de distancia ya que la estructura se mantuvo en memoria principal, pero en próximas pruebas consideraremos como otro factor importante la cantidad de accesos a disco para considerar el caso de que resida en almacenamiento secundario. El tamaño de cada intervalo se fijó como un porcentaje del intervalo total, y los FHQT asociados se construyeron con 16 pivotes elegidos aleatoriamente.

Una vez construido el índice, se ejecutaron 100 consultas por cada radio de búsqueda probado (1, 3 y 10), variando también el tamaño de los intervalos de consulta (instantánea, 10 %, 25 % y 50 % del total). Para obtener curvas de evolución de los costos en función de la cantidad de elementos, se hicieron pruebas con subconjuntos de 500, 1000, 5000, 10000 y 18000 objetos de la base de datos. Se tomó como base de comparación el algoritmo trivial planteado en [14] que se describió anteriormente.

En la Figura 5 se muestra el costo -medido en cantidad de evaluaciones de la función de distancia- de las consultas al Event-FHQT en comparación con la solución trivial, para el radio de búsqueda 1. Notar que para la solución trivial existe una única curva. Esto se debe a que la cantidad de evaluaciones de la función de distancia hechas es la misma independientemente del intervalo de consulta. Recordemos que en esta solución la componente temporal no se utiliza para descartar elementos. La navegación en el árbol se realiza usando el radio de búsqueda hasta llegar a una hoja, y allí se hace una búsqueda exhaustiva usando la componente temporal. Como sólo se realizan evaluaciones de

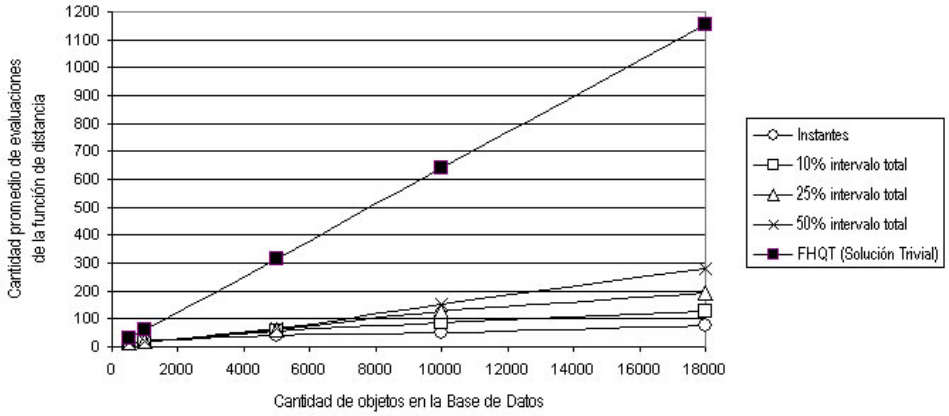

Figura 5: Resultados para radio $r = 1$.

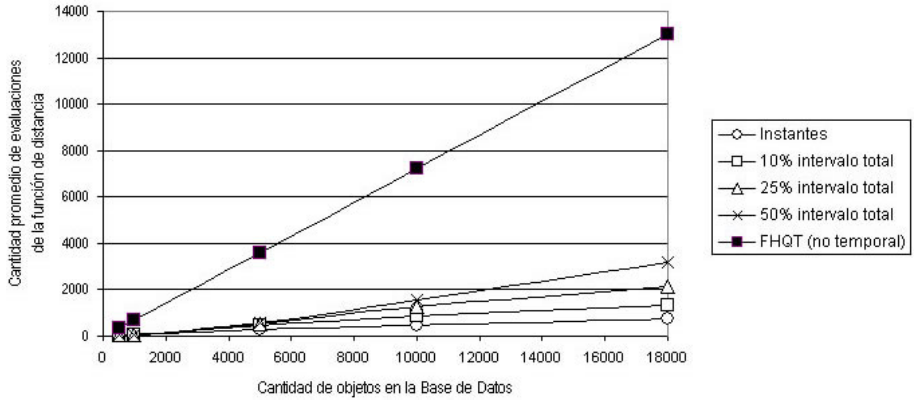

Figura 6: Resultados para radio $r = 3$.

distancias mientras se navega en el árbol, la componente temporal no afecta el costo de la búsqueda.

De la gráfica mostrada en la Figura 5 surge que los costos de las consultas métrico-temporales son significativamente menores en el Event-FHQT. En el peor de los casos -radio 1, 50 % del intervalo, 1000 elementos - el costo de consulta de este índice se reduce a un 30 % del costo de la solución trivial, y en el mejor de los casos las mejoras alcanzan un 94 %. Se evidencia un importante aumento de la eficiencia de esta estructura cuando se reduce el tamaño de los intervalos de las consultas debido a que en estos casos el filtro temporal prevalece sobre el métrico. También se puede observar que la solución trivial es mucho más sensible al aumento del tamaño de la base de datos que el Event-FHQT.

El mismo comportamiento se observa al aumentar el radio de búsqueda. Las Figuras 6 y 7 muestran los resultados para radios de búsqueda 3 y 10 respectivamente. Nuevamente el Event-FHQT es más competitivo que la solución trivial y menos sensible al aumento del tamaño de la base de datos.

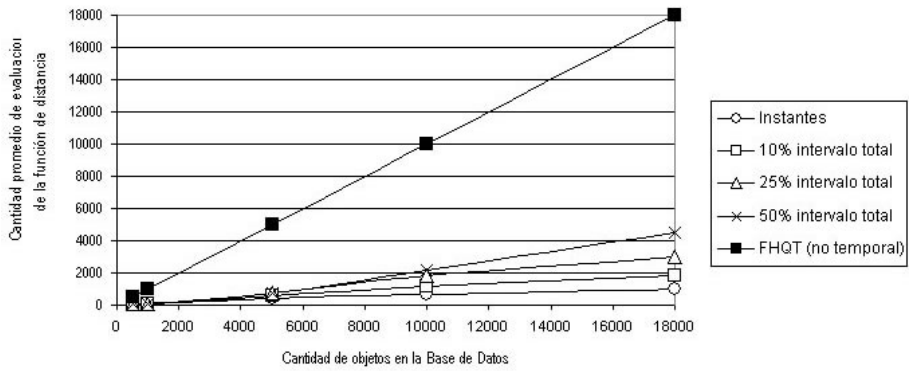

Figura 7: Resultados para radio $r = 10$.

5. Conclusiones y trabajo futuro

En este trabajo presentamos el Event-FHQT, un nuevo un método de acceso métrico-temporal, que permite resolver con eficiencia consultas métrico-temporales. Este índice se puede utilizar sobre bases de objetos tanto con instantes como con intervalos de tiempo asociados. Los experimentos han mostrado que el Event-FHQT en todos los casos tiene menor costo que la solución trivial. Esta eficiencia se logra debido a que en primer lugar se filtra por el tiempo, lo que reduce significativamente la cantidad necesaria de evaluaciones de la función de distancia para hallar la respuesta a la consulta.

Aún es necesario calcular los costos en forma analítica, tema en el cual estamos trabajando. También tenemos previsto realizar nuevas pruebas para verificar la influencia en el costo, del tamaño de los intervalos entre dos snapshots. Actualmente estamos analizando la incorporación de la cantidad de accesos a disco como otro factor en el cálculo del costo, ya que esta variable puede tener una influencia importante en la práctica. Se están considerando dos variantes: el índice completo en memoria con accesos a disco sólo cuando es necesario leer los objetos de la base de datos; y manteniendo el índice completo en disco.

Como trabajo futuro nos proponemos verificar la influencia de distintas distribuciones de datos de entrada sobre la eficiencia de las estructuras, y estudiar el problema de la selección de pivotes para la optimización simultánea de la dimensión métrica y temporal, ya que actualmente sólo se tiene en cuenta la primera.

Referencias

[1] R. Baeza-Yates. Searching: an algorithmic tour. In A. Kent and J. Williams, editors, *Encyclopedia of Computer Science and Technology*, volume 37, pages 331–359. Marcel Dekker Inc., 1997.

[2] R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixed-queries trees. In *Proc. 5th Combinatorial Pattern Matching (CPM'94)*, LNCS 807, pages 198–212, 1994.

[3] De Battista, A. Pascal, G. Gutierrez, and N. Herrera. Búsqueda en bases de datos métricas-temporales. In *Actas del VIII Workshop de Investigadores en Ciencias de la Computación*, Buenos Aires, Argentina, 2006.

[4] De Battista, A. Pascal, G. Gutierrez, and N. Herrera. Un nuevo índice métrico-temporal: el historical fhqt. In *Actas del XIII Congreso Argentino de Ciencias de la Computación*, Corrientes, Agentina, 2007.

[5] S. Brin. Near neighbor search in large metric spaces. In *Proc. 21st Conference on Very Large Databases (VLDB'95)*, pages 574–584, 1995.

[6] B. Bustos, G. Navarro, and E. Chávez. Pivot selection techniques for proximity searching in metric spaces. In *Proc. of the XXI Conference of the Chilean Computer Science Society (SCCC'01)*, pages 33–40. IEEE CS Press, 2001.

[7] E. Chávez and K. Figueroa. Faster proximity searching in metric data. In *Proceedings of MICAI 2004. LNCS 2972, Springer*, Cd. de México, México, 2004.

[8] E. Chávez, J. Marroquín, and G. Navarro. Fixed queries array: A fast and economical data structure for proximity searching. *Multimedia Tools and Applications (MTAP)*, 14(2):113–135, 2001.

[9] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Proximity searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[10] E. Chávez, N. Herrera, C. Ruano, and A. Villegas. Funciones de discretización basadas en histogramas de distancia. In *Actas de la Conferencia Latinoamericana de Informática (CLEI'06)*, Santiago, Chile, 2006.

[11] Gilberto A. Gutiérrez, Gonzalo Navarro, Andrea Rodríguez, Alejandro González, and José Orellana. A spatio-temporal access method based on snapshots and events. In *GIS '05: Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 115–124, New York, NY, USA, 2005. ACM.

[12] C. S. Jensen. A consensus glossary of temporal database concepts. *ACM SIGMOD Record*, 23(1):52–54, 1994.

[13] G. Navarro. Searching in metric spaces by spatial approximation. In *Proc. String Processing and Information Retrieval (SPIRE'99)*, pages 141–148. IEEE CS Press, 1999.

[14] A. Pascal, De Battista, G. Gutierrez, and N. Herrera. Procesamiento de consultas métrico-temporales. In *XXIII Conferencia Latinoamericana de Informática*, pages 133–144, San José de Costa Rica, 2007.

[15] B. Salzberg and V. J. Tsotras. A comparison of access methods for temporal data. *ACM Computing Surveys*, 31(2), 1999.