Un nuevo índice Métrico-Temporal: el Historical FHQT

Anabella De Battista, Andrés Pascal
Univ. Tec. Nacional, Fac. Reg. C. del Uruguay, Dpto. Sistemas de Información
Concepción del Uruguay, Argentina, 3260
{debattistaa,pascala}@frcu.utn.edu.ar

Gilberto Gutiérrez
Universidad del Bío-Bío, Facultad de Ciencias Empresariales
Chillán, Chile, 3810563
ggutierr@ubiobio.cl

Norma Herrera
Univ. Nac. de San Luis, Departamento de Informática
San Luis, Argentina, 5700
nherrera@unsl.edu.ar

Abstract

Recently a new database model, the metric-temporal databases, has been proposed. This model uses concepts of metric spaces to efficiently solve similarity queries, and concepts of temporal databases to allow store and efficiently retrieve data with a temporal component. This new model combines both aspects to solve problems where is necesary perform similarity searches, but having in account the temporal component. For it, we present the Historical FHQT, a new metric-temporal index that uses several instances of metric structure FHQT in order to represent the alive objects in each moment, and then we verified experimentally the efficiency of this method for a determined set of queries.

Keywords: metric-temporal query, metric space, temporal database, metric-temporal database, Historical-FHQT

Resumen

Recientemente ha sido propuesto un nuevo modelo de bases de datos, las bases de datos métrico-temporales, que utiliza conceptos de espacios métricos para realizar eficientemente consultas por similitud, y de bases de datos temporales para permitir almacenar y recuperar eficientemente datos que poseen una componente temporal. Este nuevo modelo combina ambos aspectos con el fin de resolver problemas donde resulta de interés realizar búsquedas por similitud pero teniendo en cuenta también la componente temporal. En este artículo se propone el Historical FHQT, un nuevo índice métrico-temporal que utiliza varias instancias de la estructura métrica FHQT para representar los objetos vigentes en cada instante de tiempo, y luego verificamos su eficiencia experimentalmente para un conjunto determinado de consultas.

Palabras clave: consulta métrico-temporal, espacio métrico, base de datos temporal, base de datos métrico-temporal, Historical FHQT

1 Introducción

En la actualidad las bases de datos no sólo almacenan datos estructurados, sino también objetos tales como imágenes, sonido, texto, video, datos geométricos y otros tipos de datos que no pueden ser consultados mediante búsquedas exactas. A su vez, en algunas situaciones es necesario tener en cuenta el aspecto temporal, es decir, registrar la evolución de los datos a través del tiempo.

Para poder representar estos nuevos tipos de bases de datos han surgido modelos que permiten almacenar eficientemente esta clase de datos y realizar búsquedas eficientemente sobre los mismos. Los modelos sobre los cuales trabajamos en este artículo son las Bases de Datos Temporales, los Espacios Métricos y las Bases de Datos Métrico-Temporales.

- Las ***bases de datos temporales*** [14, 11] permiten almacenar y recuperar datos que dependen del tiempo. Mientras que las bases de datos tradicionales tratan al tiempo como otro tipo de dato más, este tipo de base de datos incorpora al tiempo como una dimensión. Por ejemplo, una consulta de interés en una base de datos temporal de catastro podría ser: "*conocer un instante o período de tiempo en que una parcela fue propiedad de cierta persona*".

- Los ***espacios métricos*** [7, 4, 5] son un modelo de bases de datos que permiten tratar con búsquedas por similitud, es decir, búsquedas de objetos parecidos o similares a uno dado. Este tipo de búsqueda tiene una amplia gama de aplicaciones, por ejemplo: reconocimiento de imágenes y sonido, compresión de texto, biología computacional, inteligencia artificial y minería de datos, entre otras [9, 13].

- Las ***bases de datos métrico-temporales*** [15] permiten realizar consultas por similitud y teniendo en cuenta el aspecto temporal: por instante, o por intervalo de tiempo. Un ejemplo de aplicación de este modelo, es el siguiente: supongamos una base de datos de imágenes de la policía federal donde se registran rostros de un grupo de individuos. Sobre esa base de datos sería de interés, dada la especificación de un rostro, buscar las personas que tenían rasgos similares en un momento determinado. Esta consulta implica buscar teniendo en cuenta tanto la componente métrica como la componente temporal.

En este artículo presentamos un nuevo método de acceso métrico-temporal orientado a las consultas por similitud sobre una base de datos de objetos no estructurados instantáneos, es decir, que sólo tienen vigencia en un instante de tiempo.

Este artículo está organizado de la siguiente manera. En la Sección 2 presentamos una breve reseña del trabajo relacionado. En la Sección 3 presentamos el HFHQT, un índice para resolver consultas métrico-temporales. En la Sección 4 mostramos los resultados de la evaluación experimental de este nuevo método de acceso y en la Sección 6 exponemos las conclusiones y el trabajo futuro.

2 Trabajo relacionado

En esta sección se presenta un breve resumen de los modelos de bases de datos y métodos de acceso a los que se hace referencia en este trabajo: las Bases de Datos Temporales, los Espacios Métricos y las Bases de Datos Métrico-Temporales.

2.1 Bases de datos Temporales

Las bases de datos temporales mantienen información acerca del pasado, el presente y en algunos casos, pueden predecir el futuro más probable. La dimensión temporal es manejada internamente por el sistema administrador de la base de datos. Una verdadera base de datos temporal es aquella que soporta *tiempo válido* y *tiempo transaccional*. El *tiempo válido* expresa el tiempo durante el cual una proposición es cierta. El *tiempo transaccional* indica el tiempo en el que una proposición aparece reflejada en la base de datos como cierta, es decir, el momento en que se incorpora esa información en la base de datos.

Existen tres clases de bases de datos temporales, en función de lo anterior:

- **De tiempo transaccional** (transaction time): registran el tiempo de acuerdo al momento en que se almacena un hecho, es decir, de acuerdo al orden en que se procesan las transacciones. Hay que notar, que este registro no necesariamente coincide con el orden real en que se produjeron los eventos, más bien, es acorde al tiempo en que la base tomó conocimiento del evento. Debido a que se mantiene la historia de todos los estados consistentes de la base de datos, se puede realizar un "*rollback*" hacia cualquiera de estos estados anteriores. Este tipo de bases de datos no permite modificar el pasado.

- **De tiempo válido o vigente** (valid time): soportan el tiempo en que el hecho ocurrió en la realidad, que puede no coincidir con el momento de su registro. Este sistema permite realizar correcciones sobre los datos registrados. En dicho caso, sólo se mantiene la última versión de cada estado.

- **Bitemporales**: integran la dimensión transaccional y la dimensión vigente a través del versionado de los estados, es decir, cada estado se puede modificar para actualizar el conocimiento de la realidad pasada, presente o futura, pero esas modificaciones se realizan generando nuevas versiones de los mismos estados.

Los tipos básicos de consultas a sistemas de bases de datos temporales son tres:

(1) dado un intervalo continuo, encontrar todos los objetos "vigentes" en ese período
(2) dados un rango de claves y un intervalo continuo, encontrar todos los objetos cuyas claves forman parte del rango, y que estuvieron "vigentes" dentro de ese período
(3) dado un rango de claves, devolver la historia de todos los objetos cuyas claves están en ese rango.

Existen casos especiales de estos tipos de consultas como por ejemplo, que los intervalos se reduzcan a un instante (*range-time slice*), o que el rango de claves contenga una sola clave (*pure-key* query) [11, 12, 14, 17].

2.2 Espacios Métricos

Las aplicaciones en las que se requiere realizar consultas por similitud tienen un marco conceptual común que da soporte a este tipo de búsquedas bajo alguna función de distancia o similitud determinada. En esta sección introducimos el modelo formal correspondiente.

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; esto significa que a menor distancia más cercanos o similares son los objetos. Esta función *d* cumple con las propiedades características de una función de distancia:

(a) $\forall x, y \in U, d(x, y) \geq 0$ (positividad)
(b) $\forall x, y \in U, d(x, y) = d(y; x)$ (simetría)
(c) $\forall x \in U, d(x, x) = 0$ (reflexividad)
(d) $\forall x, y, z \in U, d(x, y) \leq d(x, z) + d(z, y)$ (desigualdad triangular)

La base de datos será un subconjunto finito $X \subseteq U$ de cardinalidad *n*. En este nuevo modelo de bases de datos, una de las consultas típicas que implica recuperar objetos similares es la búsqueda por rango, que denotaremos con $(q, r)_d$. Dado un elemento $q \in U$ al que llamaremos *query,* y un radio de tolerancia *r*, una búsqueda por rango consiste en recuperar los objetos de la base de datos que estén a distancia a lo sumo *r* de *q*, es decir:

$$(q, r)_d = \{x \in X / d(q, x) \leq r \}$$

Para resolver esta clase de consultas con mayor eficiencia que *O(n)* evaluaciones de distancias (correspondiente a recorrer secuencialmente la base de datos), se utilizan estructuras auxiliares (índices) que permiten ahorrar cálculos durante el proceso de búsqueda.

Existen distintos enfoques para la construcción de algoritmos de indexación en espacios métricos; en este trabajo utilizamos la estrategia basada en Pivotes.

Los Algoritmos Basados en Pivotes trabajan de la siguiente manera: durante la indexación seleccionan *k* pivotes $\{p_1, p_2, ..., p_k\}$ y le asignan a cada elemento *a* de la base de datos el vector o firma $\Phi(a)=(d(a, p_1), d(a, p_2), ..., d(a, p_k))$. Durante la búsqueda usan la desigualdad triangular junto con la firma de cada elemento para filtrar objetos de la base de datos sin medir su distancia a *q*. Dada $(q, r)_d$, se computa la firma de la query *q*, $\Phi(q)=(d(q, p_1), d(q, p_2), ..., d(q, p_k))$, y luego se descartan todos aquellos elementos *a* tales que para algún pivote p_i, $|d(q, p_i) - d(a, p_i)|>r$. Los elementos no descartados forman parte de una lista de candidatos, que posteriormente se comparan directamente con la query *q*.

De los métodos basados en pivotes, se seleccionó el FHQT (Fixed Height FQT) [1] como base para el diseño de un índice métrico-temporal. Este índice pertenece al grupo de algoritmos basados en pivotes y admite dinamismo, una característica que no todos los índices métricos poseen. A continuación se explica el funcionamiento de este índice.

Fixed-Height FQT

En [1, 2] se presenta el *Fixed-Height FQT* o *FHQT* (Figura 1), que 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 [16, 8].

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. 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.

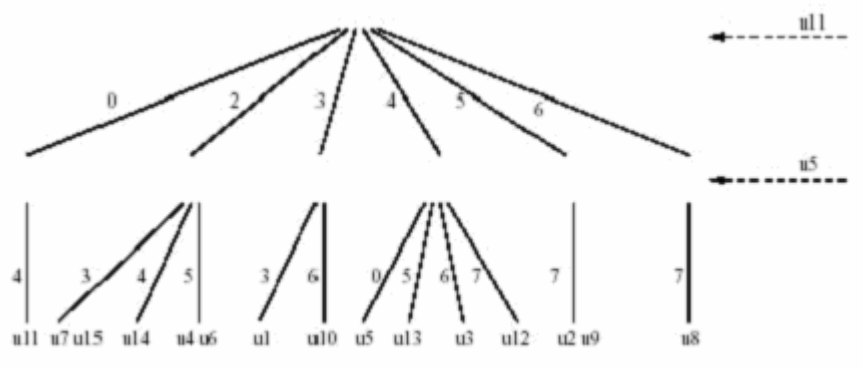

Figura 1: FHQT con pivotes u11 y u5

2.3 Bases de Datos Métrico-Temporales

Las Bases de Datos Métrico-Temporales permiten realizar búsquedas sobre objetos que tienen un intervalo de vigencia, y que no poseen un identificador que se pueda utilizar como clave de búsqueda, por lo cual tiene sentido consultar por similitud y considerando también el aspecto temporal.

Formalmente se define como sigue [15]: sea *O* el universo de objetos válidos, un **Espacio Métrico-Temporal** es un par *(U, d)*, donde el conjunto $U=O \times N \times N$ y la función *d*, es de la forma $d: O \times O \rightarrow R^+$. Cada elemento $u \in U$ es una 3-upla (obj, t_i, t_f), donde *obj* es un objeto (por ejemplo, una imagen, sonido, cadena, etc) y $[t_i, t_f]$ es el intervalo de vigencia de *obj*. La función de distancia *d*, que mide la similitud entre dos objetos, cumple con las propiedades de una métrica (positividad, simetría, reflexividad y desigualdad triangular).
Sea $X \subseteq U$ 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 con que se consulta, y *q* es el objeto de consulta. Respecto al tiempo, esta clase de consulta tiene éxito sólo para los objetos cuyo intervalo de vigencia se superpone en algún punto con el intervalo consultado. En el caso de una consulta instantánea (time-slice), $t_{iq} = t_{fq}$.

3 Método de Acceso Métrico-Temporal: el Historical FHQT

En este artículo se propone una nueva estructura de acceso métrico-temporal denominada Historical FHQT (HFHQT), que utiliza el índice métrico FHQT [1] para considerar el aspecto métrico, e ideas de índices temporales para tratar el aspecto temporal.

El HFHQT consiste en una lista de los instantes válidos de tiempo, donde cada celda contiene un índice FHQT de todos los objetos vigentes en dicho instante. Esta estructura, que puede considerarse trivial, es eficiente en bases de datos métrico-temporales en donde los objetos tienen vigencia en un sólo instante de tiempo.

3.1 Estructura

Formalmente, un HFHQT es un par (*li*,*lp*) donde:

- *li* es una lista (f_1, f_2, ..., f_n) en la cuál [1, *n*] es el intervalo válido de tiempo, y cada f_i es un FHQT correspondiente al instante *i*, o *nil* si no hay ningún objeto vigente en dicho momento.
- *lp*=(p_1, p_2, …, p_{max}) es la lista de pivotes utilizados en la construcción de todos los árboles. *max* es la cantidad de pivotes del árbol más profundo de la lista.

Los árboles pueden tener distintas profundidades de acuerdo a la cantidad de elementos que se 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 profundos cuando la cantidad de objetos es baja, con el fin de que la estructura no tenga un costo espacial excesivo.

El valor *max* se utiliza durante la consulta para determinar el tamaño de la firma del objeto consultado. La lista *lp* contiene los pivotes que se utilizan en todos los árboles; el pivote p_i es el pivote correspondiente al nivel *i* de los árboles que poseen al menos dicho nivel.

La estructura es dinámica, permitiendo altas de dos tipos: históricas o de nuevos instantes.

- Un alta es histórica cuando se incorpora un objeto a un instante ya existente. El costo de este tipo de alta es el costo de calcular la firma del nuevo objeto, siempre que no haya que reestructurar el árbol.
- Un alta de un nuevo instante, toma como entrada un conjunto de objetos, construye el FHQT correspondiente a dicho conjunto, y lo agrega al final de la lista *li* como nuevo instante. Los instantes en los cuales no haya objetos vigentes, deben ser agregados a *li* a través de conjuntos vacíos, y se representan dentro de la lista con el valor *nil*.

3.2 Consulta

Las consultas métrico temporales se efectúan de la siguiente manera: en primer lugar se seleccionan de la lista *li* los instantes incluidos en el intervalo de consulta. Posteriormente se realizan consultas por similitud usando cada uno de los FHQT correspondientes, y se unen los conjuntos resultantes.

En la Figura 2 se muestra un ejemplo del HFHQT. El intervalo total representado es [1, 12]. La estructura contiene FHQTs sólo en los instantes 2, 5, 7, 8 y 9, ya que en el resto no hay objetos

vigentes registrados. Como se ve, los árboles no tienen todos la misma profundidad: los correspondientes a los instantes 2 y 9 utilizan sólo un pivote; los correspondientes a los instantes 7 y 8 usan dos pivotes y el 5 contiene un FHQT de 3 pivotes. En el ejemplo $lp = (p_1, p_2, p_3)$ y max=3.

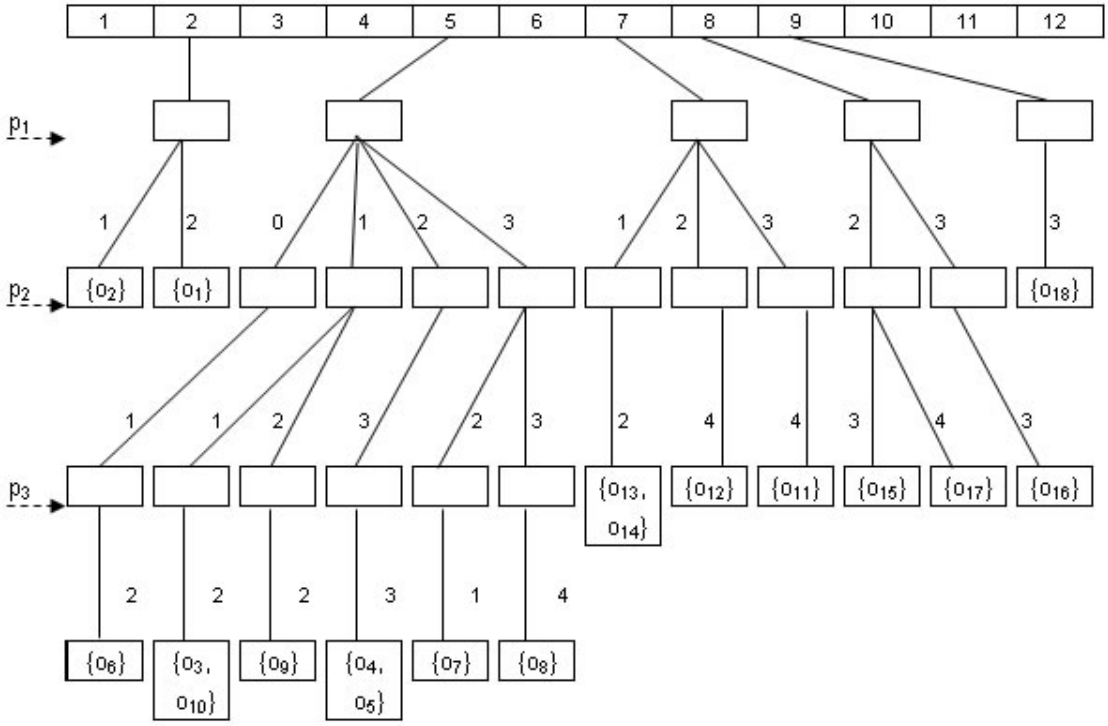

Figura 2: Historical FHQT

Para la consulta $(q, 1, 6, 9)_d$, siendo *(1, 2, 4)* la firma de *q*, el HFQT se comporta de la siguiente manera: se acceden en forma directa los instantes 6, 7, 8 y 9 y se busca por similitud en los FHQT correspodientes. El instante 6 se ignora ya que no posee un FHQT asociado. En el instante 7, como la firma de *q* contiene como primer elemento a 1, y el radio de la búsqueda es 1, se toman las ramas 1 *(1-1≤1≤1+1)* y 2 (*1-1≤2≤1+1*), y se descarta la 3. En el siguiente nivel, sólo cumple la restricción la rama 2, por lo cual los objetos candidatos son $\{o_{13}, o_{14}\}$. El mismo proceso se ejecuta para los instantes 8 y 9, dando como candidato a $\{o_{15}\}$. Luego estos conjuntos se unen en $\{o_{13}, o_{14}, o_{15}\}$ y los objetos contenidos se comparan con la consulta *q* para obtener el resultado final.

3.3 Algoritmo de consulta

En la Figura 3, se muestra el pseudocódigo del algoritmo de consulta al HFHQT. La función toma como entrada el objeto de consulta *q*, el radio de búsqueda *r*, y el intervalo que se consulta $[t_{iq}, t_{fq}]$. Como primer paso se calcula la firma de *q* para todos los pivotes utilizados en al menos un árbol del HFHQT; luego se consulta cada FHQT correspondiente al intervalo consultado y se unen los conjuntos resultantes para obtener el resultado final

HFHQT$(q, r, t_{iq}, t_{fq})_d$
begin
  calcular la firma f de q -- *distancias a cada uno de los pivotes de lp*
  candidatos:={}
  for all (fhqt h correspondiente a los instantes del intervalo $[t_{iq}, t_{fq}]$ del HFHQT)
    candidatos :=candidatos $\cup h(q, r, f)_d$
  resultado:={$x \in$ candidatos | $d(q, x) \leq r$ }
  return resultado
end.

Observación: $h(q, r, f)_d$ devuelve el conjunto de objetos candidatos a ser similares a q con radio r a través de una consulta al FHQT h, donde f es la firma de q.

Figura 3: Pseudocódigo de consulta del HFHQT

4 Resultados Experimentales

Para verificar su funcionamiento se realizó la implementación del HFHQT y se probó sobre una base de datos de 750 imágenes representadas a través de vectores de *762* dimensiones [9] a las cuales se les agregó un número natural que representa el instante de validez de la imagen. El intervalo total considerado fue *[1, 100]*. Se utilizó la distancia coseno discretizada [16, 8] como función de distancia, y los costos se expresaron en cantidad de evaluaciones de dicha función. No se evaluaron otras variables que podrían tener influencias importantes en los resultados, como por ejemplo, la cantidad de accesos a disco.

Los árboles correspondientes a cada instante de tiempo se construyeron tomando de una lista de pivotes elegidos al azar [6]. La cantidad de pivotes de cada árbol se calculó como $\lceil \log_2(|O_i|) \rceil$, donde $|O_i|$ es la cantidad de objetos vivos en el instante i. En cada prueba se ejecutaron y promediaron *100* consultas variando el radio de búsqueda (*1, 3 y 10*), el tamaño promedio del intervalo de consulta (instantánea; 10%, 25% y 50% del intervalo total), y la cantidad de elementos de la base de datos (*100, 250, 500 y 750*).

La comparación se realizó contra la solución trivial planteada en [15], que consiste en construir un FHQT para toda la base de datos y ante una consulta, realizar la búsqueda por similitud utilizando dicho índice, para luego eliminar del conjunto resultante los elementos que no cumplen con la restricción temporal. En esta solución, el costo está dado sólo por la cantidad de evaluaciones de la función de distancia en las consultas al FHQT, y es el mismo tanto para consultas instantáneas como para consultas por intervalos.

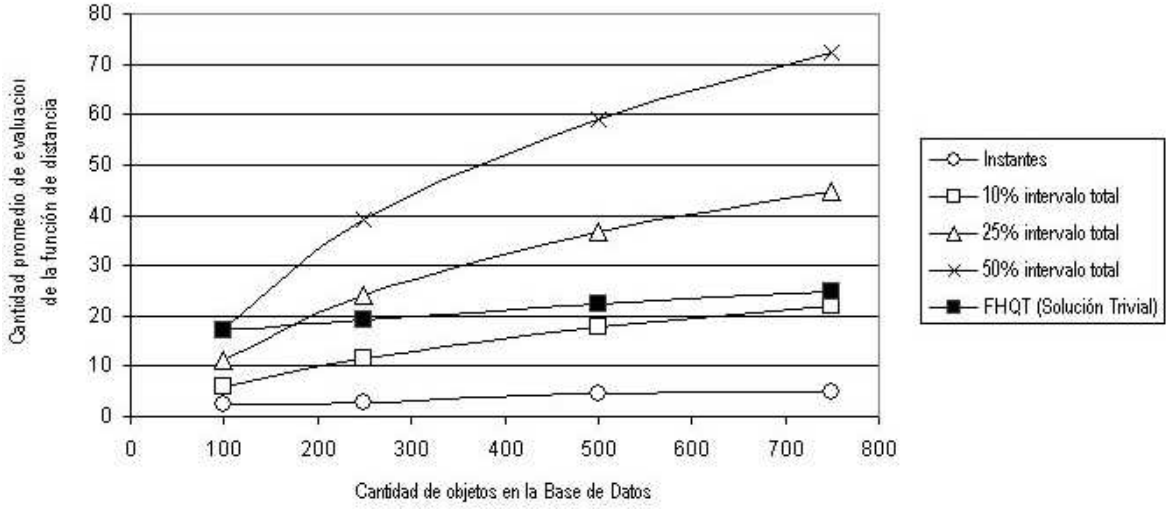

Figura 4: Historical FHQT, radio 1

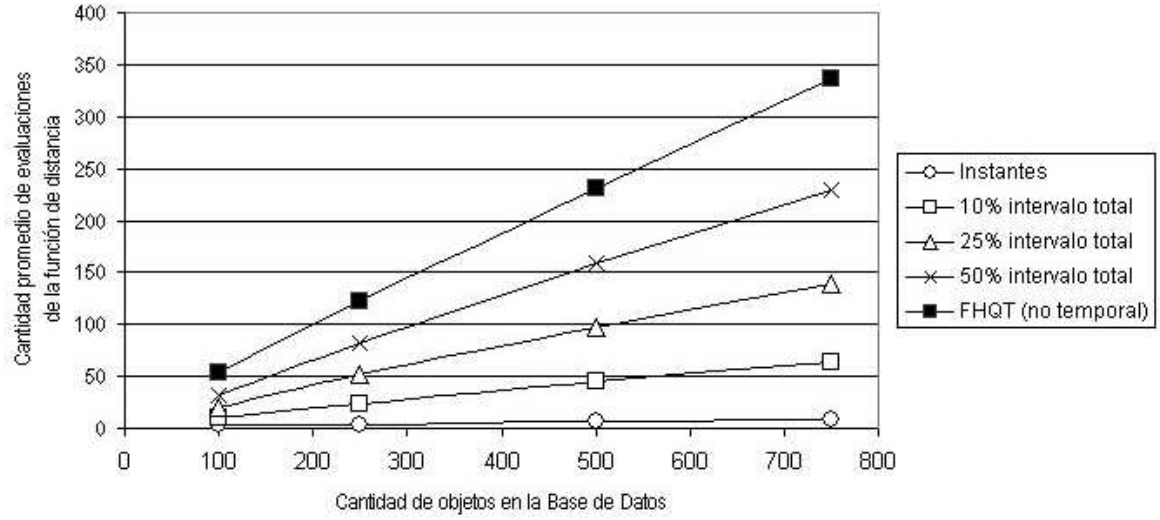

Figura 5: Historical FHQT, radio 3

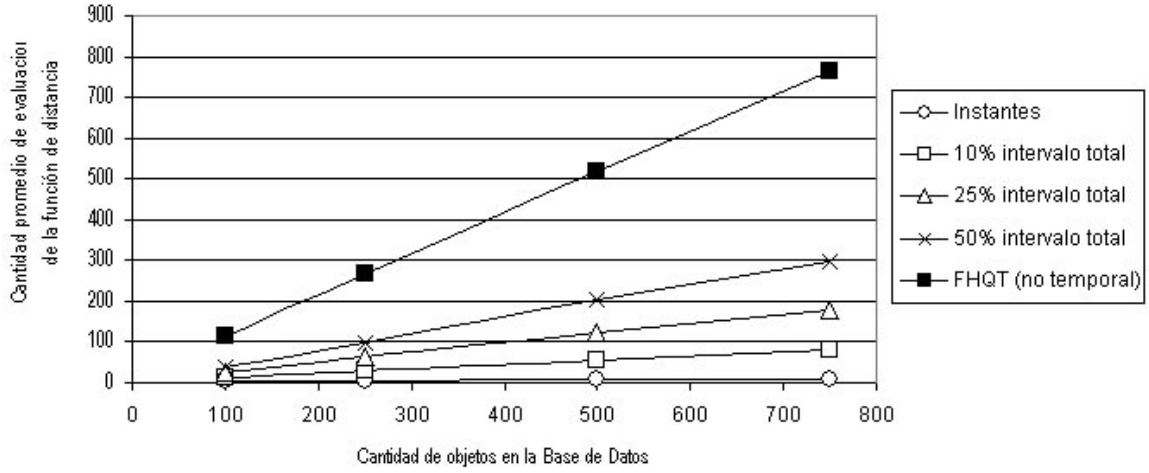

Figura 6: Historical FHQT, radio 10

En las Figuras 4, 5 y 6 se muestran las cantidades promedio de evaluaciones de la función de distancia ante consultas al HFHQT y de la solución trivial, para los radios de búsqueda *1, 3* y *10* respectivamente.

Cuando la consulta es instantánea (time-slice) el HFHQT obtiene sus mejores resultados, alcanzando entre un 524% y un 7628% de disminución del costo sobre la solución trivial. Esto es debido a que mientras en la solución trivial se debe consultar un árbol que contiene todos los objetos de la base de datos, en el HFHQT sólo es necesario acceder al árbol que contiene los objetos vigentes en el instante requerido en la consulta, que normalmente es mucho mas chico.

Como se ve, en las consultas por intervalo el HFHQT también se comporta mejor que la solución trivial, excepto para radio 1 con intervalos del 25 o 50% del tiempo total. En estos casos, la solución trivial es más eficiente ya que la capacidad de filtrado de un único FHQT que contiene todos los objetos de la base de datos y por lo tanto, que posee mayor cantidad de niveles (pivotes), es superior a la de varios FHQTs de menor profundidad. Se verificó experimentalmente que la eficiencia del HFHQT en estos casos puede ser mayor a la de la solución trivial, si se utiliza una cantidad fija, igual a la máxima, de pivotes para todos los árboles.

Costos

Un HFHQT es muy eficiente ante consultas por similitud instantáneas o por intervalos reducidos. Sea *dt* la densidad temporal (cantidad promedio de objetos por instante de tiempo), el costo de una consulta $(q, r, t_{iq}, t_{fq})_d$ al HFHQT es $c(t_{fq} - t_{iq}+1)$ donde *c* es el costo de la consulta $(q, r)_d$ a un FHQT con *dt* cantidad de objetos y $\lceil \log_2(|dt|) \rceil$ niveles [1, 2, 3]. En una consulta por similitud instantánea, *c* se reduce a 1, es decir que se consulta sólo un árbol, y de tamaño reducido en comparación con el FHQT de la solución trivial planteada. Si *n* es la cantidad de objetos de la base de datos, en este caso se descartan (*n-dt*) objetos en forma directa, sin realizar ningún cálculo de la función de distancia.

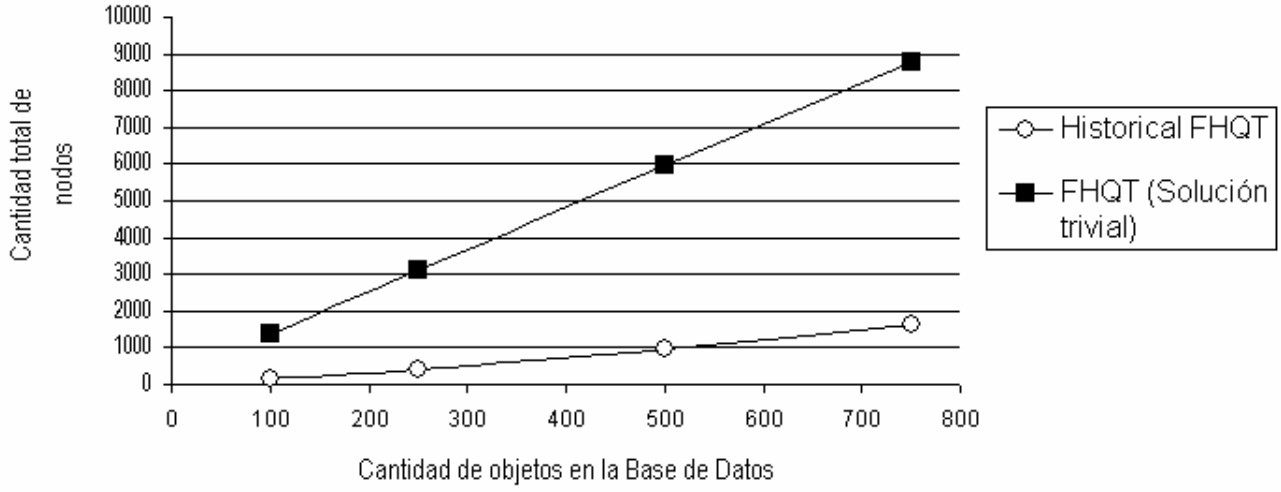

Figura 7: Costo Espacial del Historical FHQT

El HFHQT tiene menor costo espacial que el FHQT de la solución trivial, si se sigue la estrategia de utilizar cantidades logarítmicas de pivotes para los árboles correspondientes a cada instante. Los resultados se presentan en la Figura 7, donde se compara la cantidad total de nodos de los árboles del HFHQT, con la cantidad de nodos de un único FHQT para 100, 250, 500 y 750 objetos. Esto se debe a que varios árboles pequeños ocupan menos espacio que un solo árbol de mayor profundidad,

ya que normalmente el incremento en nodos de un nivel al siguiente es exponencial. En caso de utilizar una cantidad fija de pivotes para todos los árboles, igual a la de la solución trivial, el costo espacial del HFHQT alcanza a ser hasta el doble que el de un único FHQT.

5 Conclusiones y Trabajo Futuro

En este trabajo presentamos un nueva estructura -el Historical FHQT-, orientada a resolver consultas métrico-temporales instantáneas sobre una base de datos de objetos instantáneos. El método se comporta eficientemente en estos casos, aunque también puede ser utilizado para consultar intervalos de tiempo, pero con menor eficiencia.

Los resultados de los experimentos realizados muestran que los costos de consultas por similitud instantáneas o por intervalos pequeños son significativamente menores que los de la solución trivial planteada. Esta eficiencia se logra debido a que se consulta una cantidad reducida de árboles de pocos elementos porque se descartan en forma directa todos los objetos que no estuvieron vigentes en el instante o intervalo requerido.

Una desventaja de este método es que las consultas métricas puras tienen mayor costo que el mismo tipo de consultas a un FHQT. Por otro lado, es de notar que en una consulta temporal pura el acceso a los instantes es directo, y los objetos se recuperan a través de un recorrido a los árboles correspondientes.

Este nuevo índice se presenta como un primer avance de una solución más completa en la que estamos trabajando actualmente. En la misma, introducimos modificaciones al HFHQT tomando ideas del HR-Tree, para mejorar su eficiencia ante consultas por intervalos, y permitir representar objetos que tengan asociado un intervalo de vigencia en lugar de instantes.

También estamos organizando nuevas pruebas sobre una base de datos de 30000 imágenes con el fin de verificar el comportamiento del índice en situaciones más cercanas a la realidad. Sobre esta base haremos pruebas de eficiencia tomando en cuenta la variable “cantidad de accesos a disco” como un factor más en el establecimiento de los costos, ya que consideramos que puede tener una influencia importante en los resultados.

Referencias

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

[2] Baeza-Yates, R. 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.

[3] Baeza-Yates, R., Navarro G. Fast Approximate String Matching in a Dictionary. In Proceedings of SPIRE'98, pages 14-22. String Processing and Information Retrieval: a South American Symposium, IEEE Computer Society, Sept. 1998

[4] Bozkaya, T., Ozsoyoglu, M. Distance-based indexing for high-dimensional metric spaces. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 357–368, Sigmod Record 26(2), 1997.

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

[6] Bustos, B., Navarro, G. and Chávez, E. 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] Chávez, E., Navarro, G., Baeza-Yates, R. and Marroquín, J.L. Searching in metric spaces. In *ACM Computing Surveys*, 33(3):273.321, September 2001.

[8] Chávez, E., Herrera, N., Ruano, C., Villegas, A. Una implementación completa del FQTrie. *VII Workshop de Investigadores de Ciencias de la Computación*, pp 61-65. 2005.

[9] Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D. and Equitz, W. Efficient and effective querying by image content. *Journal of Intelligent Information Systems (JIIS)*, 3(3/4):231–262, 1994.

[10] Guttman, A. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 47-54. 1984

[11] Jensen, C.S. editor et al. A Consensus Glossary of Temporal Database Concepts. In *ACM SIGMOD Record.* 23(1):52-64. 1994.

[12] Kumar, A., Tsotras, V.J., Faloutsos, C. Designing Access Methods for Bitemporal Databases. IEEE Transactions on Knowledge and Data Engineering. 10(1):1-20. 1998.

[13] Navarro, G. A guided tour to approximate string matching. ACM Computing Surveys (CSUR), 33(1):31–88, 2001.

[14] Ozsoyoglu, G., Snodgrass, R.T. Temporal and Real-Time Databases: A Survey. IEEE Trans. on Knowledge and Data Engineering. 7(4):513-532, 1995.

[15] Pascal, A., De Battista, A., Gutiérrez, G., Herrera, N. Procesamiento de Consultas Métrico-Temporales. Artículo aceptado para su presentación en la XXXIII Conferencia Latinoamericana de Informática (CLEI), 2007.

[16] Ruano, C., Chávez, E., Herrera, N. Discretización binaria del FQtrie. In Actas del X Congreso Argentino de Ciencias de la Computación (CACIC'04), pages 100–111, Buenos Aires, Argentina, 2004.

[17] Salzberg, B., Tsotras, V.J. A Comparison of Access Methods for Temporal Data. ACM Computing Surveys. 1999.