Procesamiento de Consultas Métrico-Temporales

Andrés Pascal, Anabella De Battista
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

The temporal databases allow efficiently to store and recover data that have a temporal component. The metric spaces are a databases model that support similarity searches, that is to say, searches of objects similar to a given query element. There are a vast number of applications where it turns out from interest to also make searches by similarity but having in account the temporal component. This new type of query cannot be efficiently solved neither with temporal indexes, nor with metric indexes. In this article we undertook the study of these queries in order to formalize this concept and to propose efficient methods for its resolution. For it, we present the FHQT-Temporal, an adaptation of metric index FHQT with the aggregate of time intervals. Finally we verified experimentally the efficiency of this structure of access for a determined set of queries.

Keywords: metric-temporal query, metric space, temporal database, similarity search, FHQT-Temporal

Resumen

Las bases de datos temporales permiten almacenar y recuperar eficientemente datos que poseen una componente temporal. Los espacios métricos son un modelo de bases de datos que soportan búsquedas por similitud, es decir, búsquedas de objetos parecidos a uno dado. Existen aplicaciones donde resulta de interés realizar búsquedas por similitud pero teniendo en cuenta también la componente temporal. Este nuevo tipo de consultas no puede resolverse eficientemente ni con índices temporales, ni con índices métricos. En este artículo abordamos el estudio de estas consultas con el fin de formalizarlas y proponer métodos eficientes para su resolución. Para ello, presentamos el FHQT-Temporal, una adaptación del índice métrico FHQT con el agregado de intervalos de tiempo. Finalmente verificamos experimentalmente la eficiencia de esta estructura de acceso para un conjunto determinado de consultas.

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

Este trabajo ha sido parcialmente financiado por el proyecto “Algoritmos para Evaluación de Consultas Espacio-Temporales”, código 073218 A/R de la Universidad del Bío-Bío (Chile) y por el proyecto “Nuevas Tendencias y Aplicaciones en Bases de Datos” del Universidad Tecnológica Nacional, Facultad Regional de Concepción del Uruguay (Argentina).

1 Introducción

En 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 que contienen campos comparables. Una búsqueda en la base de datos retorna todos aquellos registros cuyos campos coinciden con los aportados en la consulta. Otra característica importante de las bases de datos clásicas es que capturan sólo un estado de la realidad modelada, 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 otro tipos de datos tales como imágenes, sonido, texto, video, datos geométricos, etc. La problemática de almacenamiento y búsqueda en éstas difiere notablemente de la correspondiente a las bases de datos clásicas. En primer lugar los datos generalmente son no estructurados, esto significa que difícilmente puedan ser organizados en registros y campos. En segundo lugar, aún cuando tal estructuración fuera posible, la búsqueda exacta carece de interés en este ámbito; no es de interés por ejemplo, buscar una imagen que sea exactamente igual a una dada. Y en tercer lugar, en muchas aplicaciones es de importancia mantener todos los estados de la base de datos y no sólo el más reciente; más aún, hasta puede surgir la necesidad de realizar proyecciones de estados futuros.

Es en este contexto donde surgen nuevos modelos de bases de datos capaces de cubrir eficaz y eficientemente las necesidades de almacenamiento y búsqueda de estas aplicaciones. Entre esos nuevos modelos se encuentran las bases de datos temporales y los espacios métricos:

- Las ***bases de datos temporales*** [12, 13, 14, 17] 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*** [8, 3, 4] 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 [10, 16].

Existen aplicaciones donde resulta de interés realizar búsquedas por similitud pero teniendo en cuenta también la componente temporal. Por ejemplo, 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 el año 1990. Esta consulta implica buscar teniendo en cuenta tanto la componente métrica como la componente temporal.

Un segundo ejemplo es el siguiente: supongamos la existencia de una serie de "puntos de control" en las distintas rutas de una región. Cada punto de control posee una cámara fotográfica automática que toma fotos de los vehículos que pasan por dicho punto. Cada imagen, aunque sea del mismo auto, se almacena como un objeto distinto. Estas imágenes se guardan en una base de datos registrando también el momento (fecha y hora) real del suceso. Si existen 15 puntos de control en dicha región, y hay una circulación media de 180 vehículos por hora por punto, por día se generarán alrededor de 64.800 imágenes. Una consulta típica en esta situación sería *"encontrar los Peugeot 307 color azul marino, que pasaron por algún punto de control, entre las 5:30 am y las 9:00 am"*. Las consultas deben ser por similitud, ya que las imágenes se registran sin identificar ni clasificar. Intentar consultar la base de datos completa, que en un año alcanzaría más de 2 millones de imágenes, es altamente ineficiente y en la práctica no serviría si hay urgencia. Lo más adecuado sería buscar por similitud en un período de tiempo determinado.

Este nuevo tipo de consultas, a las que llamaremos ***consultas métrico-temporales***, no pueden ser resueltas con eficiencia mediante índices temporales, dado que los mismos están organizados para realizar búsquedas exactas sobre los objetos y aquí necesitamos consultar por similitud. Tampoco pueden ser resueltas eficientemente por índices métricos [5, 1, 7, 9] porque si bien estos índices permiten realizar búsquedas por similitud, sólo capturan un instante determinado de tiempo y aquí necesitamos que la componente temporal sea considerada como un elemento de decisión al momento de realizar la consulta. En consecuencia, se hacen necesarios nuevos enfoques para atacar esta problemática.

En este artículo abordamos el estudio de este tipo de consultas con el fin de proponer métodos eficientes para su resolución. De acuerdo a nuestro conocimiento no existen trabajos que aborden el procesamiento de consultas métrico-temporales, por lo tanto este artículo representa un primer intento tanto en el planteamiento y formalización del problema, como en el diseño de métodos de acceso para procesar las consultas planteadas.

Este artículo está organizado de la siguiente manera. En la Sección 2 presentamos una breve reseña del trabajo relacionado. En las Secciones 3 y 4 planteamos el problema de consultas métrico-temporales y proponemos una solución para las mismas: el FHQT-Temporal. En la Sección 5 exponemos la evaluación experimental de este nuevo índice y por último, en la Sección 6 expresamos las conclusiones y el trabajo futuro.

2 Trabajo relacionado

Si bien ni los índices usados en bases de datos temporales ni los utilizados en espacios métricos están adaptados para resolver una consulta métrico-temporal, los mismos servirán como base para el diseño de un índice que resuelva las consultas métrico-temporales de manera eficiente. En esta sección presentamos una breve reseña de estos dos modelos de bases de datos.

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, solo 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) [17, 14, 15, 19].

2.2 Espacios Métricos

Como lo mencionáramos en la Sección 1, las búsquedas por similitud tienen una amplia gama de aplicaciones tales como reconocimiento de imágenes y sonido, compresión de texto, biología computacional, inteligencia artificial y minería de datos, entre otras. Todas estas aplicaciones comparten un marco de trabajo común, que es el de buscar

objetos que sean similares bajo alguna función de distancia o similitud adecuada. En esta sección introducimos el modelo formal que abarca todos estos casos.

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:

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

Una búsqueda por rango puede ser resuelta con *O(n)* evaluaciones de distancias examinando exhaustivamente la base de datos. Para evitar esta situación, se preprocesa la base de datos por medio de un algoritmo de indexación con el objetivo de construir una estructura de datos (índice) diseñada para ahorrar cálculos en el momento de resolver una búsqueda. Un algoritmo de indexación se considera eficiente si puede responder una búsqueda por similitud haciendo una cantidad pequeña de cálculos de distancia, sublineal en la cantidad de elementos de la base de datos.

Básicamente existen dos enfoques para la construcción de algoritmos de indexación en espacios métricos:

- **Algoritmos Basados en Pivotes:** estos algoritmos, 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*.
- **Algoritmos Basados en Particiones Compactas:** este tipo de algoritmo divide el espacio en *k* zonas tan compactas como sea posible. Para ello se selecciona un conjunto de *centros* $\{c_1, ..., c_k\}$ y se le asigna a cada centro c_i el conjunto $[c_i]$ formado por todos los elementos de *X* que tienen a c_i como su centro más cercano. Existen varios criterios posibles para descartar zonas durante la búsqueda. Los dos más populares son: criterio del hiperplano y criterio del radio de cobertura.
    a. **Criterio del hiperplano:** es el más básico y el que mejor expresa la idea de partición de Voronoi. Básicamente, si *c* es el centro de la clase *[q]* (es decir, el centro más cercano a *q*) entonces la bola con centro *q* no intersecta $[c_i]$ si $d(q, c) + r < d(q, c_i) - r$. Es decir, si la bola asociada a *q* no intersecta el hiperplano que divide su centro más cercano *c* y el centro c_i, entonces cae fuera de la clase de c_i.
    b. **Criterio del radio de cobertura:** en este caso se trata de limitar la clase $[c_i]$ considerando la bola centrada en c_i que contiene todos los elementos de *X* que caen en la clase. Definimos el radio de cobertura de *c* para *X* de la siguiente manera:

$$cr(c) = max_{(x \in [c] \cap X)}\, d(c, x)$$

    Ahora es claro que podemos descartar $[c_i]$ si $d(q, c_i) - r > cr(c_i)$.

De los métodos existentes, 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 (FHQT o FHFQT)

En [1, 2] se presenta el *Fixed-Height FQT* o *FHQT* (Figura 3), 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 [18, 9].

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 3 muestra un ejemplo de un FHQT con dos pivotes.

La Figura 1 muestra un ejemplo sobre un conjunto de puntos en un espacio bidimensional usando como función de distancia la distancia euclideana. En esta figura se puede apreciar la partición que se provoca sobre el espacio cuando se elige a u_{11} como pivote y se discretizan las distancias. Cada anillo se corresponde con un conjunto C_i asociado a u_{11}. Las Figuras 2 y 3 muestran un FQT y un FHQT respectivamente, creados sobre el conjunto de puntos dado en la figura 1.

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.

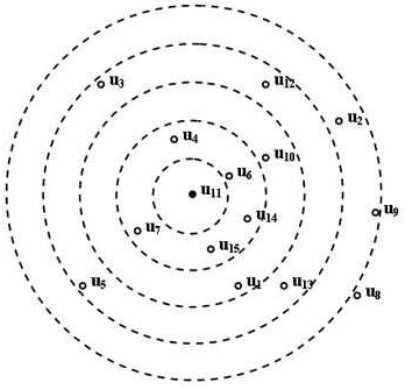

Figura 1: Partición provocada sobre un conjunto de puntos cuando se utiliza u_{11} como pivote y se discretizan los valores de la función de distancia *d*.

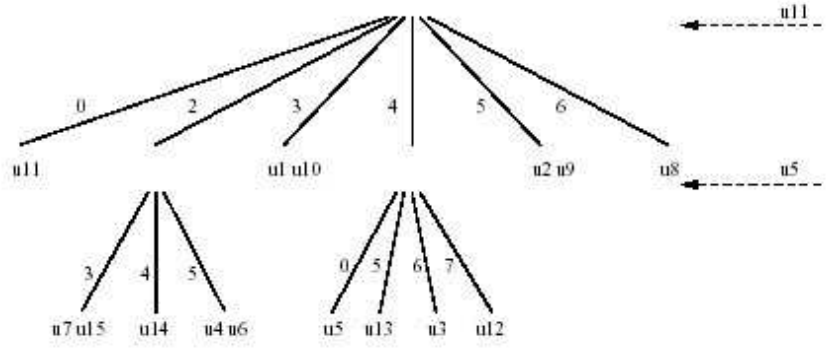

Figura 2: FQT con pivotes u11 y u5

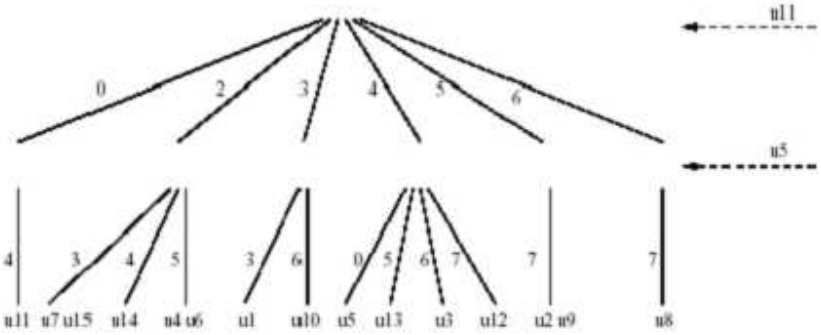

Figura 3: FHQT con pivotes u11 y u5

3 Bases de Datos Métrico-Temporales

En la Sección 1 vimos ejemplos donde resulta de interés realizar consultas por similitud pero teniendo en cuenta también la componente temporal. En general las aplicaciones donde tienen sentido las consultas métrico-temporales tienen las siguientes características:

- No se pueden realizar búsquedas exactas sobre los objetos: los elementos de la base de datos no tienen un identificador (o un grupo de atributos) que se pueda utilizar como clave de búsqueda, o si existe, no se conoce en el momento de la consulta. Esto sucede por ejemplo, cuando los objetos se incorporan a la base de datos en tiempo real, y no hay tiempo suficiente para clasificarlos en el momento.
- Los objetos tienen un intervalo de vigencia. Este intervalo representa el período en el cual el objeto tiene significado para la realidad, por ejemplo el intervalo de tiempo en el que un auto permanece en un estacionamiento. En algunos casos, dicho intervalo se puede reducir a un instante de tiempo.
- En las consultas se necesita combinar la similitud con el aspecto temporal.
- La base de datos contiene una cantidad suficientemente grande de objetos o el tiempo de respuesta ante una consulta debe ser suficientemente reducido como para que no tenga sentido realizar una búsqueda secuencial.

Sea O el universo de objetos válidos, un **Espacio Métrico-Temporal** es un par (U, d), donde el conjunto $U=O\text{x}N\text{x}N$ y la función d, es de la forma $d: O\text{x}O \rightarrow R^+$. 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 solo 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), $t_{iq} = t_{fq}$.

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 $(q, r, t_{iq}, t_{fq})_d$, se procede de la siguiente manera:

1. Realizar la búsqueda $(q, r)_d$ sobre el índice métrico,
2. Realizar una búsqueda secuencial sobre el conjunto de objetos resultantes del punto 1, para obtener los que cumplen con la restricción temporal, es decir, los objetos cuyo intervalo de vigencia se superpone en algún punto con el intervalo de consulta $[t_{iq}, t_{fq}]$.

La desventaja que tiene esta solución trivial es que no se usa la componente temporal para filtrar la búsqueda en el índice; en este proceso sólo se aprovecha la componente métrica.

Una mejor estrategia es que, durante el proceso de búsqueda, se utilice tanto la componente métrica como la componente temporal para descartar elementos. Básicamente nuestra propuesta se basa en la idea de adaptar un índice métrico

agregándole el intervalo de vigencia de cada objeto, y en cada paso de la búsqueda filtrar usando toda la información disponible, tanto temporal como métrica.

4 Método de Acceso Métrico-Temporal: el FHQT-Temporal

En este trabajo se presenta un índice que adapta el índice métrico FHQT [1] mediante la incorporación de la dimensión temporal en base a ideas del índice espacio-temporal RT-tree [20, 11].

Este nuevo índice se denomina FHQT-Temporal. Para su construcción, al FHQT se le agregó un intervalo de tiempo en cada nodo del árbol. Este intervalo representa el máximo período de tiempo 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 un nodo interior, el intervalo se calcula tomando el tiempo inicial mínimo, y el tiempo final máximo de sus hijos.

Estructura
Formalmente, un nodo interior N del FHQT-Temporal es una 3-upla $(t_{incial}, t_{final}, \{(d_1, h_1), (d_2, h_2),\ldots, (d_m, h_m)\})$ donde:

- $h_1..h_m$ son los m hijos del nodo N,
- las d_i , para $i=1..m$, son las distancias entre el pivote correspondiente al nivel de N y los objetos contenidos en las hojas de los subárboles de h_i.
- los dos primeros componentes de la 3-upla, t_{incial} y t_{final}, se definen de la siguiente manera: $t_{inicial} = min_{j=1...m}\ (t_{inicial}\ (h_j))$, y $t_{final} = max_{j=1...m}\ (t_{final}\ (h_j))$.

Las hojas tienen una estructura similar: están representadas por una 3-upla $(t_{incial}, t_{final}, \{e_1, e_2,\ldots, e_f\})$, donde

- los e_i para $i=1..f$ son los f elementos que contiene la hoja, que a su vez están formados por tres componentes: el objeto (o un puntero al mismo), el tiempo inicial de vigencia del objeto, y su tiempo final.
- los valores t_{incial}, t_{final} poseen el mismo significado que para los nodos interiores, pero aplicados a los elementos e_i.

Consulta
Cuando se realiza una consulta métrico-temporal, ya sea instantánea (time-slice) o de intervalo (time-interval), se procede de la siguiente manera: en cada nivel del árbol, se filtran los subárboles hijos en primer lugar por el intervalo de tiempo y si cumple con esta condición, 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.
En símbolos: sea la consulta $(q, r, t_{iq}, t_{fq})_d$, y el nodo interior que se visita $(t_{incial}, t_{final}, \{(d_1, h_1), (d_2, h_2),\ldots, (d_m, h_m)\})$, primero se comprueba que $(\ t_{inicial} \leq t_{fq}\ ) \wedge (\ t_{iq} \leq t_{final}\ )$, es decir, que haya superposición temporal. Si este predicado devuelve verdadero, entonces se recorren los m nodos hijos, ingresando solo a los subárboles h_j que cumplan con la condición de similitud: $|\ d(\ q, p_{niv}\ ) - d_j\ )\ | \leq r$, donde p_{niv} es el pivote correspondiente al nivel, y descartando los demás. Finalmente, cuando se alcanzan las hojas, se recorren los conjuntos de elementos contenidos en cada una de ellas, y se seleccionan los objetos que cumplen con la condición temporal, y con la condición de similitud $d(q, o(e_i)) \leq r$, donde $o(e_i)$ es el objeto correspondiente al elemento e_i.

Construcción del índice
La construcción del árbol se realiza utilizando el mismo procedimiento que para el FHQT, pero además actualizando los intervalos de tiempo de la rama en la cual se ubica el nuevo objeto. Si para algún nodo de la rama, el tiempo inicial del nuevo objeto es menor que el inicial del nodo, es necesario cambiar este último, asignándole el correspondiente al nuevo objeto. En forma similar, si el tiempo final del objeto es mayor que el tiempo final del nodo, este último también deberá actualizarse.

En la Figura 4 se ve un ejemplo del FHQT-Temporal para el mismo conjunto de objetos de las Figuras 1, 2 y 3. El nodo raíz, cuyo pivote es "u_{11}", contiene el intervalo de tiempo 0-98, donde el 0 es el mínimo tiempo de inicio (correspondiente a "u_1") y 98, es el tiempo final mayor (correspondiente a "u_3"). Como se ve, cada nodo posee un intervalo formado por el mínimo tiempo inicial de sus hijos, y el máximo tiempo final de los mismos.

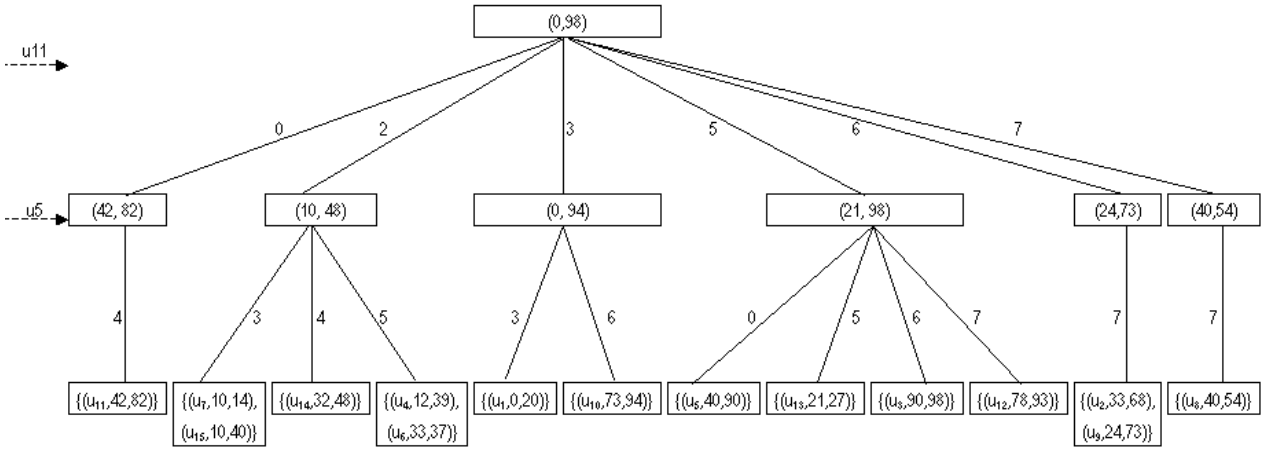

Figura 4: FHQT-Temporal

Ante la consulta $(q, 2, 5, 20)_d$, y siendo $(5, 4)$ la firma de q, los subárboles de la raíz con distancias 3, 5, 6 y 7 cumplen con el requisito de similitud, ya que $(5\text{-}2 \leq 3 \leq 5+2)$, $(5\text{-}2 \leq 5 \leq 5+2)$, $(5\text{-}2 \leq 6 \leq 5+2)$ y $(5\text{-}2 \leq 7 \leq 5+2)$. Sin embargo, de estas ramas solo la que contiene al intervalo $[0,94]$ cumple con la superposición temporal. De esta manera, las demás ramas se descartan y así se ahorran las evaluaciones de la función de distancia correspondientes a los objetos contenidos en estas últimas.

Algoritmo de consulta

En la Figura 5, se muestra el pseudocódigo del algoritmo de consulta. El algoritmo utiliza recursividad, y está organizado en dos funciones. En la primera se calcula la firma del objeto que se consulta, y luego se invoca a la función *Consultar*, que realiza la búsqueda partiendo del nodo raíz. En la segunda función se recorre el árbol descartando las ramas que no cumplen con las restricciones temporales o métricas, y tratando las demás. Cuando se alcanzan las hojas, se realiza una búsqueda secuencial sobre los objetos contenidos en las mismas, devolviendo aquellos que cumplen ambas restricciones.

```
FHQTT (q, r, t_iq, t_fq)_d
begin
   Calcular la firma F de q              --lista de distancias a cada uno de los pivotes
   Return Consultar (q, r, t_iq, t_fq, 1, F, raíz)
end.

Consultar(q, r, t_iq, t_fq, n, F, nodoActual)        --n es el nivel de nodoActual dentro del árbol
  resultado:={}
  If  (intervalo del nodoActual se superpone en algún punto a [t_iq, t_fq]) then
  begin
     If (nodoActual es hoja) then
        For all (objeto x contenido en el nodo hoja)
           If  (intervalo de x se superpone en algún punto a [t_iq, t_fq]) and (d(q, x)<= r) then
              resultado :=resultado ∪ {x}
     else                                        --el nodoActual no es hoja
        For all (hijo h_i de nodoActual)
           If (abs(F(n) – d_i) <= r) then        -- d_i es la distancia asociada a hijo i de nodoActual
              resultado:=resultado ∪ Consultar(q, r, t_iq, t_fq, n+1, F, h_i)
  end
  Return resultado
end.
```

Figura 5: Pseudocódigo de consulta del FHQT-Temporal

5 Resultados Experimentales

Como parte de este trabajo se implementó el FHQT-Temporal y se realizaron pruebas sobre una base de datos de 750 imágenes representadas a través de vectores de *762* dimensiones [10], con un intervalo de tiempo asociado a cada una. El intervalo total fue *[1,100]*. Se utilizó la distancia coseno discretizada [18, 9] como función de distancia, y los costos se expresaron en cantidad de evaluaciones de dicha función. En muchas situaciones reales, otro de los factores relevantes es el costo de acceso a disco, que en este caso no se evaluó.

El árbol se construyó tomando *16* pivotes elegidos al azar [6] desde la base de datos. 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 tomando como base la solución trivial explicada en la Sección 3, en la cual el costo total de una consulta está dado únicamente por el costo de búsqueda por similitud en el FHQT.

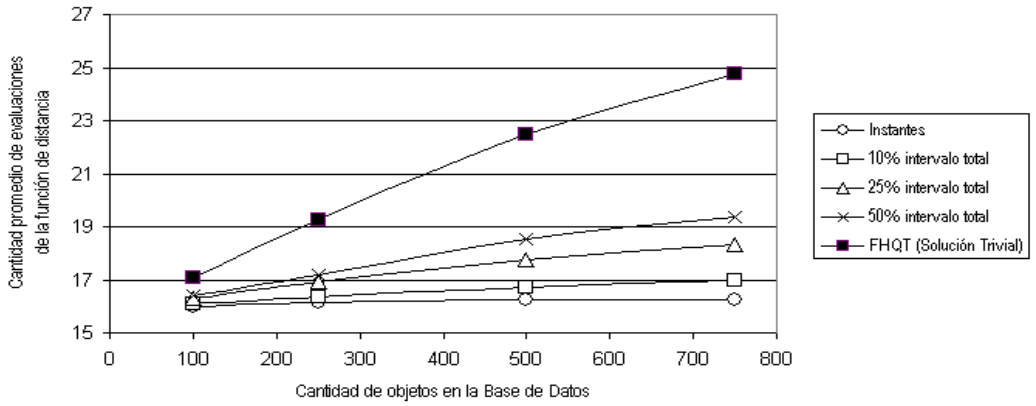

Figura 6: FHQT-Temporal, radio 1

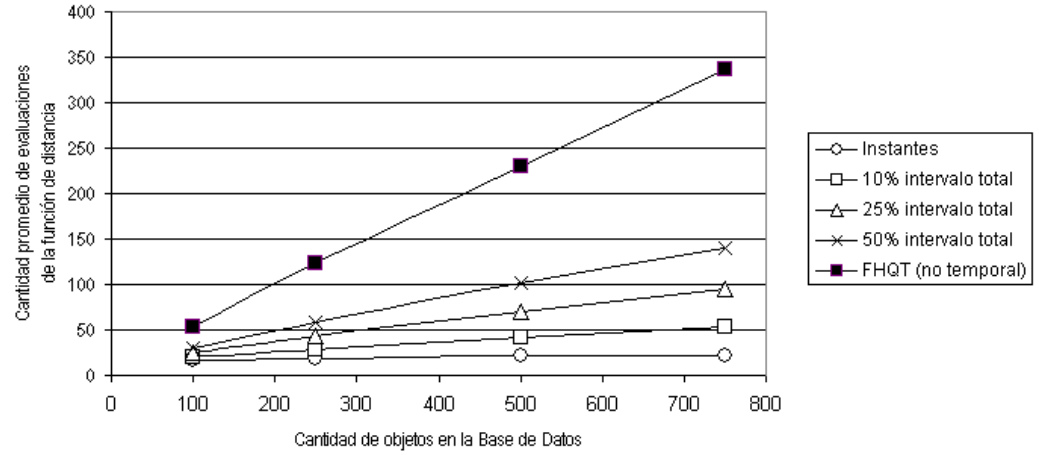

Figura 7: FHQT-Temporal, radio 3

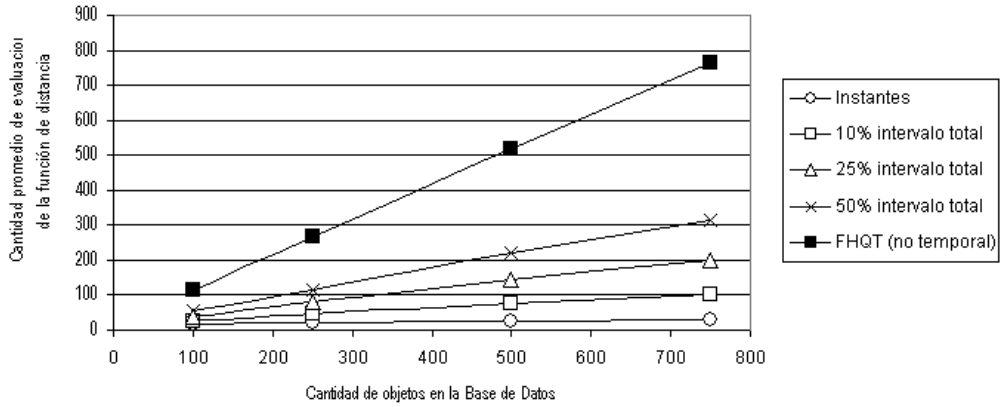

Figura 8: FHQT-Temporal, radio 10

En las Figuras 6, 7 y 8 se muestran los resultados de las pruebas de FHQT-Temporal y de la solución trivial, para los radios de búsqueda *1, 3* y *10* respectivamente. Como se ve, el FHQT-Temporal resuelve las consultas métrico-temporales en la mayoría de los casos en la mitad de tiempo o menos que la solución trivial. Su eficiencia aumenta claramente cuando el tamaño de los intervalos de consultas es menor, obteniendo su mejor rendimiento en el caso de las consultas por instantes de tiempo (time slice). Esto es consecuencia de la mayor capacidad de filtrado de ramas del árbol que poseen los intervalos pequeños.

En la Figura 9, se muestran las tablas de resultados correspondientes a los gráficos anteriores. En estas pruebas, la solución trivial (FHQT) tuvo que evaluar la función de distancia desde un 6% (para radio 1, 100 objetos) hasta un 2641% (para radio 10, 750 objetos, consulta de instantes de tiempo) más veces que las necesarias para el FHQT-Temporal. La causa de esta diferencia es que a medida que aumenta el radio de búsqueda y disminuye el intervalo de consulta, la dimensión temporal de la consulta se hace más importante. En consecuencia, la diferencia de eficiencia entre ambas soluciones aumenta considerablemente. Esto se ve claramente comparando las columnas *Instantes* y *FHQT* para radio 10.

	Radio 1				
Cant Objetos	**Instantes**	**10%**	**25%**	**50%**	**FHQT**
100	16	16	16	16	17
250	16	16	17	17	19
500	16	17	18	19	22
750	16	17	18	19	25

	Radio 3				
Cant Objetos	**Instantes**	**10%**	**25%**	**50%**	**FHQT**
100	17	20	25	31	54
250	19	29	43	58	123
500	21	42	71	101	230
750	22	53	95	140	337

	Radio 10				
Cant Objetos	**Instantes**	**10%**	**25%**	**50%**	**FHQT**
100	17	27	40	55	116
250	21	45	79	116	266
500	27	76	143	220	516
750	29	102	200	316	766

Figura 9: Cantidad de evaluaciones de la función de distancia para el FHQT-Temporal y para la solución trivial (FHQT)

En esta misma figura es de notar que los valores resultantes en *Radio 1* son iguales en varios casos a 16. Este número es la cantidad de evaluaciones necesarias para calcular la firma de la consulta (ya que se trabajó con 16 pivotes) y es el mínimo costo para cualquier consulta a un FHQT-Temporal con dicha cantidad de pivotes.

Costos
El costo de las consultas con un FHQT-Temporal es menor, o igual en el peor de los casos, al del FHQT. Este nuevo índice mejora la capacidad de descarte de ramas del árbol a través del filtrado mediante la condición temporal, que se traduce en una reducción de la cantidad de evaluaciones de la función de distancia. El peor de los casos se da cuando cada objeto posee un intervalo de vigencia igual al intervalo total. En esta situación –irreal en la mayoría de los casos-, se pierde toda capacidad de filtrado por tiempo.

Una característica importante del FHQT-T es que es dinámico, es decir que puede recibir inserciones en cualquier momento sin necesidad de reestructurar el árbol, siempre que no se modifique la cantidad de pivotes utilizados. Otra ventaja de esta estructura es que permite realizar consultas métricas puras con el mismo costo que las consultas mediante un FHQT. También permite realizar consultas temporales puras, aunque en el peor de los casos éstas tendrán un costo lineal. Esto se debe a que la dimensión temporal está subordinada a la dimensión métrica ya que en la construcción del FHQT-T, los criterios de selección de pivotes utilizados no incluyen aún consideraciones para la optimización del aspecto temporal.

El costo extra en espacio, en comparación con el FHQT, proviene de agregar a cada nodo del árbol dos valores: el tiempo inicial y tiempo final de vigencia del objeto. Sea *e* el espacio requerido para almacenar un instante de tiempo, y *n* la cantidad de nodos del FHQT-T, el espacio total adicional necesario para almacenar el árbol es *(2.e.n)*. En cada nodo interior, el FHQT necesita espacio para almacenar apuntadores y distancias a sus hijos. Sea *a* el espacio necesario para un apuntador, *d* el necesario para una distancia, y *h* la cantidad promedio de hijos para los nodos interiores, entonces el espacio requerido por un nodo es *(h.(a+d))*. Ya que el orden de magnitud de *(a+d)* es similar a *e*, podríamos usar la aproximación $e \cong (a+d)$. Para este caso, el factor de costo adicional de espacio del FHQT-T por nodo interior, es $(2.e)/(h.(a+d)) = (2.e)/(h.e) = 2/h$. La variable *h* depende de la cantidad máxima de valores distintos que puede tomar la función de distancia, cantidad que puede ser manejada a través de funciones de discretización [18, 9] y que normalmente es varias veces superior a la constante 2. En el experimento presentado para 750 elementos, $h = 7$, por lo tanto el incremento en almacenamiento es del 29%.

6 Conclusiones y Trabajo Futuro

En este trabajo presentamos un nuevo tipo de consultas -las métrico-temporales-, y el FHQT-Temporal, un método de acceso que permite resolver dichas consultas con eficiencia. Un aspecto novedoso es la combinación de la dimensión temporal con la métrica y su aplicación a la solución de determinados problemas. Otra contribución es la creación de este nuevo índice mediante la incorporación de intervalos de tiempos al FHQT. El FHQT-Temporal se utiliza en problemas donde se requiera consultar por similitud e intervalo de tiempo en forma simultánea, a bases de datos con gran cantidad de objetos no estructurados y con un período de validez asociado.

Un análisis preliminar muestra que el FHQT-Temporal tiene menor costo para las consultas métrico-temporales, y que además permite resolver consultas métricas puras con el mismo costo que el FHQT. También permite consultas temporales puras. Los experimentos realizados verifican la reducción de los costos de consultas utilizando esta estructura. 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 el costo en tiempo en forma analítica, tema en el cual estamos trabajando actualmente. Además estamos analizando la incorporación de la cantidad de accesos a disco como otro factor en el cálculo del costo. Se sabe que esta variable puede tener una influencia importante en la práctica. Se están considerando dos variantes: el índice completo en memoria, y accesos a disco solo cuando es necesario leer los objetos de la base de datos; y manteniendo el índice en disco. Estamos preparando nuevos experimentos con una base de datos de 30000 imágenes para verificar el comportamiento de dicha variable. Pensamos extender esta cantidad a valores más elevados, similares a los planteados en las situaciones descriptas, y medir el costo en tiempo real.

Además, queremos estudiar el problema de la selección de pivotes para la optimización simultánea de la dimensión métrica y temporal, ya que actualmente solo se tiene en cuenta la primera.

Por otro lado, tenemos pensado crear -tomando como base índices métricos, temporales, espaciales, o espacio-temporales- otras estructuras de acceso orientadas a las consultas métrico-temporales, que posean mayor eficiencia que el FHQT-Temporal, o que lo complementen en situaciones en las que este índice no tenga un buen comportamiento.

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] Bozkaya, T. and 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.

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

[5] Burkhard, W. and Keller, R. Some approaches to best-match file searching. In *Comm. of the ACM*, 16(4):230–236, 1973.

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

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

[8] 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.

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

[10] 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.

[11] 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.

[12] Gutiérrez, G , Navarro, G, Rodríguez, A, González, A and Orellana, J. A spatio-temporal access method based on snapshots and events. In *Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems (GIS'05)*. ACM Press, 2005.

[13] Gilberto Gutiérrez, Gonzalo Navarro, and Andrea Rodríguez. $Sest_L$: An event-oriented spatio-temporal access method. *Technical Report TR/DCC-2006-5, Department of Computer Science*, Universidad de Chile (Chile), 2006.

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

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

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

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

[18] Ruano, C., Villegas A.,Chávez, E. and Herrera, N. Funciones de Discretización Basadas en Histogramas de Distancias. *In Actas del XXXII Conferencia Latinoamericana de Informática*, CLEI 2006, Santiago, Chile, 2006.

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

[20] Xu, X., Han, J. and Lu, W. RT-Tree: An Improved R-Tree Indexing Structure for Temporal Spatial Databases. *In Proc. of the Intl. Symp. on Spatial Data Handling, SDH*, pages 1040–1049, July 1990.