Consultas Métrico-Temporales

Andrés Pascal, Anabella De Battista
Universidad Tecnológica Nacional, Facultad Regional Concepción del Uruguay

Norma Herrera
Universidad Nacional de San Luis, Departamento de Informática

Gilberto Gutierrez
Universidad del Bio Bio, Facultad de Ciencias Empresariales

Abstract
El modelo de bases de datos métrico-temporal permite abordar aquellas situaciones en las que resulta necesario realizar búsquedas por similitud teniendo en cuenta también un componente temporal. En este artículo presentamos los tipos de consultas métrico-temporales más relevantes sobre este modelo, y en particular se propone un algoritmo para resolver las búsquedas por similitud de los k vecinos más cercanos restringidas a un intervalo o instante de tiempo. Además se presentan resultados experimentales que comparan la eficiencia de este algoritmo versus la solución trivial.

Palabras Claves: Consultas métrico-temporales, Bases de Datos Métrico-Temporales, Espacios Métricos, Bases de Datos Temporales, Índices Métricos, Búsquedas por Similitud.

1 Introducción

El hecho de que las bases de datos actualmente requieran almacenar tipos de datos no estructurados, también conocidos como datos multimedia (imágenes, audio, video, texto), y el hecho de que para este tipo de información no tenga sentido la búsqueda exacta , ha derivado en un nuevo paradigma de consulta denominado *búsquedas por similitud*, que se basa en recuperar los elementos de la base de datos que tengan cierto parecido con el objeto que se consulta. Por otro lado, en algunas aplicaciones interesa, además de consultar por similitud, tener en cuenta el instante o intervalo de vigencia de los objetos. Como solución a esta problemática surgen tres modelos que permiten almacenar esta clase de datos y realizar los tipos de búsquedas requeridos:

Espacios Métricos [6, 5, 4, 8] es un modelo de bases de datos que permite la manipulación de objetos multimedia y la realización de búsquedas por similitud sobre los mismos. Este tipo de búsqueda tiene una amplia gama de aplicaciones, como reconocimiento de imágenes y sonido, compresión de texto, biología computacional, inteligencia artificial y minería de datos, entre otras [9, 13].

Bases de Datos Temporales [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 modelo incorpora al tiempo como una dimensión. Por ejemplo, una consulta de interés en una base de datos temporal podría ser: 'conocer el período de tiempo en que una persona fue empleado de cierto departamento de una empresa'. Existen tres clases de bases de datos temporales de acuerdo a la forma en que gestionan el tiempo: *de tiempo transaccional*, en el cual el tiempo se registra en función al orden en que se procesan las transacciones; *de tiempo válido*, que registra el instante en que el hecho ocurrió en la realidad (puede no coincidir con el momento de su registro)

y *bitemporales*, que integran las dimensiones transaccional y válido 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 Métrico-Temporales [2, 3, 10] integran a los dos modelos anteriores. Permiten buscar objetos similares a uno dado cuyo intervalo de vigencia se superpone con el intervalo aportado en la consulta. Un ejemplo de aplicación de este modelo es el siguiente: supongamos que se tiene un registro fotográfico de las personas que ingresan a un museo y el período de tiempo que permanecieron en el mismo. Sobre dicha base se podría buscar, dada la especificación de un rostro y un intervalo temporal, cuáles fueron las personas que tenían rasgos similares al ingresado y permanecieron en el museo en dicho período de tiempo.

En este artículo se realiza un resumen de los distintos tipos de consultas métrico-temporales y se propone un algoritmo para resolver las consultas temporales por intervalo o instante, en combinación con los k vecinos más cercanos respecto al grado de similitud. También se presentan resultados experimentales que muestran la eficiencia de este algoritmo en comparación con la solución trivial.

2 Marco Teórico

En este apartado se define y explica el modelo *Métrico-Temporal*, que combina consultas por similitud basadas en el Modelo de Espacios Métricos, con búsquedas que incluyen alguna variable tiempo como uno de sus parámetros principales.

2.1 Espacio Métrico-Temporal

Sea U un universo de objetos válidos, en su forma más general se define un Espacio Métrico-Temporal mediante el par (X, d), donde $X = U \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ y la métrica $d : U \times U \rightarrow \mathbb{R}^+$. Cada elemento $x \in X$ es una 5-upla $(o, t_{vi}, t_{vf}, t_{ti}, t_{tf})$, donde o es un objeto (una huella digital, un rostro, un sonido, etc), $[t_{vi}, t_{vf}]$ es el intervalo de validez de o en la realidad y $[t_{ti}, t_{tf}]$ el intervalo de tiempo transaccional asociado. Por simplicidad, se definen todos los tiempos como valores pertenecientes al conjunto $\mathbb{N}$. Estos valores pueden ser fechas, horas, etc., pero en cualquier caso se pueden representar mediante números naturales. La función de distancia d mide la disimilitud entre dos objetos y cumple con las propiedades de toda métrica, es decir, positividad, simetría, reflexividad y desigualdad triangular.

Las situaciones problemáticas que este nuevo modelo de bases de datos permite resolver, se caracterizan a través de los siguientes puntos:

- No tiene sentido realizar búsquedas exactas sobre los objetos, es decir, los elementos de la base de datos no tienen una clave que se pueda utilizar en la búsqueda, o la clave existe pero no está almacenada en la base de datos al momento de consultar, o existe y se encuentra registrada, pero la consulta en si no contiene la clave.

- Los objetos poseen uno o dos instantes o intervalos de validez asociado. Uno de estos intervalos representa el período en el cual el objeto se encuentra vigente en la realidad, por ejemplo, el instante en que se registra una huella digital en el acceso a un edificio. El segundo intervalo representa el periodo de tiempo en el cual el objeto se dió de alta en la base de datos hasta su baja, ya sea por modificación o eliminación.

- Existen consultas en las cuales se requiere combinar las búsquedas por similitud con el aspecto temporal. Se puede requerir también, realizar consultas por similitud puras o consultas temporales puras.

- La base de datos contiene una cantidad suficientemente grande de objetos, o bien,

el tiempo de respuesta ante una consulta debe ser suficientemente reducido como para que no tenga sentido realizar una búsqueda secuencial.

2.2 Métodos de Acceso Métrico-Temporales

En los últimos años se han desarrollado índices métrico-temporales que permiten realizar consultas por similitud y temporal con eficiencia. Estos son: el *FHQT-Temporal* [10] y su variante para funciones de distancia continua *FHQT$^+$-Temporal* , el *Event-FHQT* [9] y su variante *Event-FHQT$^+$*, el *Historical-FHQT* [3] y el *Pivot-FHQT* [1]. En esta sección se describe el *FHQT$^+$-Temporal*, sobre el cual se plantean las consultas métrico-temporales en este artículo, y se propone un algoritmo para resolver eficientemente la búsqueda NN_k*/rank/*- para este índice.

2.3 FHQT$^+$-Temporal

El *FHQT-Temporal* es un FHQT al cual se le agrega un intervalo de tiempo en cada nodo del árbol. Este intervalo representa el período de tiempo de vigencia de 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.

La estructura es dinámica, permitiendo tanto altas como bajas de objetos, ya sea en instantes o intervalos contenidos en el intervalo que el índice posee hasta el momento, como alta de objetos con tiempos fuera de éste. En ambos casos el costo de la inserción, medido en cantidad de evaluaciones de la función de distancia, es el costo de calcular la firma del nuevo objeto.

El *FHQT$^+$-Temporal* es una variante del *FHQT-Temporal* generalizada que soporta valores continuos de la función de distancia. Para ello, en lugar de asociar un número natural a cada hijo de un nodo, se asocian dos intervalos de valores de distancias. El primer intervalo representa el rango máximo correspondiente a la rama, mientras que el segundo (incluido en el anterior) constituye el rango actual de valores, es decir, el intervalo formado por el mínino y el máximo valor de distancia del pivote a los objetos contenidos en las hojas del subárbol. Los intervalos máximos son constantes y se calculan cuando se construye el árbol en base al histograma de distancias, de tal manera de que el árbol tenga una alta probabilidad de quedar balanceado. Los intervalos actuales son variables y se van actualizando de acuerdo a los objetos que se insertan en la estructura. Para determinar en qué rama se agrega un nuevo elemento, se utilizan los intervalos máximos, y ante una consulta sólo se usan los actuales. Al utilizar estos últimos intervalos en las búsquedas, se incrementa la capacidad de filtrado por similitud, ya que los intervalos son más pequeños y quedan espacios vacíos entre intervalos consecutivos.

2.3.1 Estructura

Un *FHQT$^+$-Temporal* es un árbol r-ario donde el valor de r es un parámetro que se define en forma previa a su construcción, normalmente en base a la distribución del histograma de distancias.

Formalmente, un *FHQT$^+$-Temporal* es un árbol donde:

- todo nodo interior V es una 3-upla (t_{ini}, t_{fin}, $\{(int_{x1}, int_{a1}, h_1), (int_{x2}, int_{a2}, h_2),..., (int_{xm}, int_{am}, h_m)\}$) donde:
  - $h_1..h_m$ son los m hijos del nodo V,
  - los int_{xi} , para $i = 1..m$, son los intervalos *máximos* de distancias entre el pivote correspondiente al nivel de V y los objetos que pueden pertenecer a las hojas de los subárboles de h_i.
  - los int_{ai} , para $i = 1..m$, son los intervalos *actuales* de distancias entre el pivote correspondiente al nivel de V y los objetos contenidos actual-

mente en las hojas de los subárboles de h_i.

  - los dos primeros componentes de la 3-upla, t_{ini} y t_{fin}, se definen de la siguiente manera: $t_{ini} = min_{j=1..m}(t_{ini}(h_j))$, y $t_{fin} = max_{j=1..m}(t_{fin}(h_j))$.

- Las hojas se definen de la misma forma que para el *FHQT-Temporal*

Para calcular los intervalos máximos correspondientes al nodo raíz del árbol, se toma una muestra de la base de datos, se calcula el histograma de distancias y se divide el espacio en r intervalos, de tal manera de que cada uno de ellos posea la misma cantidad (± 1) de elementos. Luego para cada nodo hijo se procede de la misma manera, pero considerando solo los elementos de la muestra que fueron asignados a dicho nodo. De esta manera todos los nodos interiores tendrán exáctamente r hijos. Una vez definidos los rangos, se realiza la inserción de los elementos, actualizando los intervalos actuales. Sea o el objeto a insertar, v el nodo donde se quiere insertar el objeto, $[d_{xi}, d_{xf})$ y $[d_{ai}, d_{af}]$ los intervalos máximo y actual asociados al nodo, y p el pivote del nivel, primero se verifica que $d(p, o) \in [d_{xi}, d_{xf})$ y si esto se cumple, se actualiza el intervalo actual haciendo $d_{ai} := min(d_{ai}, d(p, o))$ y $d_{af} := max(d_{af}, d(p, o))$. El aspecto temporal se procesa de la misma manera que en el *FHQT-Temporal*. En la Figura 1 se muestra un ejemplo del *FHQT$^+$-Temporal*.

Este índice permite resolver consultas por similitud puras, temporales puras y métrico-temporales con funciones de distancia tanto continuas como discretas.

3 Consultas Métrico- Temporales

Los tipos de consultas métrico-temporales resultan de combinar los tipos de consultas por similitud (por rango y de los k vecinos más cercanos) con los tipos de consultas temporales (rango e instantánea correspondientes a las dimensiones transaccional y válida). Para ello se introduce una notación similar a la de tres entradas definida para el modelo temporal en [12]. La terna *similarity/valid/transaction* indica a través de su primer entrada el tipo de consulta por similitud, $range$ o NN_k, la segunda entrada el tipo de consulta de tiempo válido, $rank$ o $point$, y la tercera el tipo de consulta de tiempo transaccional.

Sea $X \subseteq U \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ el conjunto finito de objetos contenidos en la base de datos con sus tiempos asociados, los tipos de consultas métrico-temporales más importantes son:

1. **range/rank/-** y **range/-/rank** Devuelve todos los objetos similares a uno dado dentro de un radio de tolerancia, que poseen un tiempo de validez (primer tipo) o de transacción (segundo tipo) que se intersecta con un intervalo dado. Formalmente se denota mediante la 4-upla $(q, r, t_{vi}, t_{vf})_d$ y se define como:

$$(q, r, t_{vi}, t_{vf})_d = \{\, o/(o, t_{vio}, t_{vfo}, t_{tio}, t_{tfo}) \in X \wedge d(q, o) \leq r \ \wedge (t_{vio} \leq t_{vf}) \ \wedge \ (t_{vi} \leq t_{vfo}) \,\}$$

   La variable q es el objeto que se consulta y r es el radio de búsqueda que representa el grado de similitud requerido. Las variables t_{vi} y t_{vf} son el tiempo inicial y final respectivamente, del intervalo consultado. El intervalo puede hacer referencia al tiempo válido o transaccional, según se requiera.

2. **range/point/-** y **range/-/point** Consultas similares a las anteriores, pero con $t_{vi} = t_{vf}$, es decir, en lugar de un intervalo se consulta sólo un instante.

3. NN_k**/rank/-** y NN_k**/-/rank** Devuelve los k objetos con mayor similitud a uno dado, que poseen un tiempo de validez (primer caso) o transacción (segundo caso) que se intersecta con un intervalo dado. Sea X una base de datos métrico-temporal, una

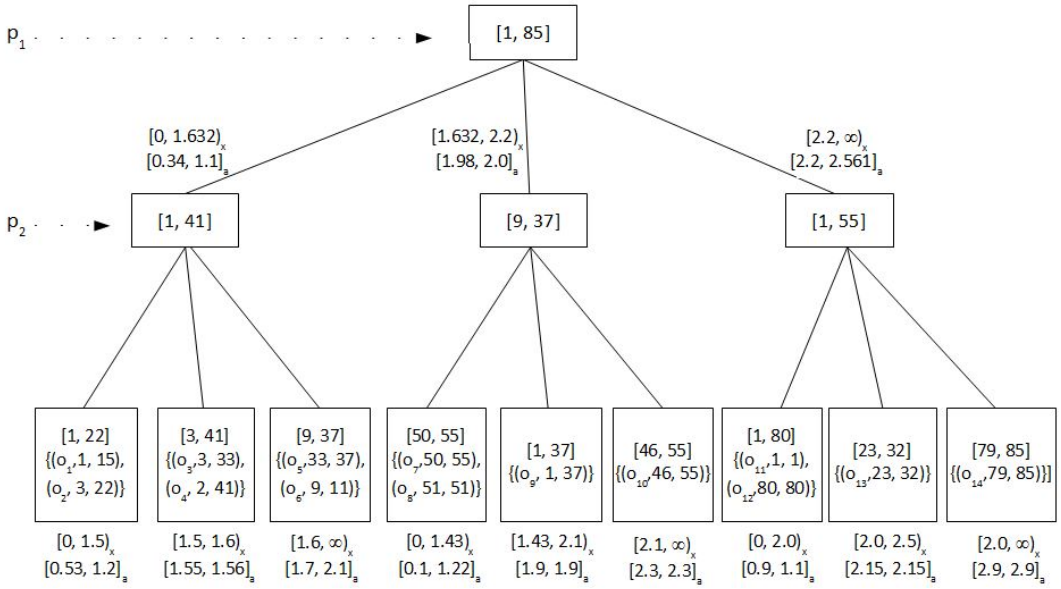

Figura 1: *Ejemplo de un FHQT$^+$-Temporal*

consulta NN_k/rank/- se denota mediante la 4-upla $(q, k, t_{vi}, t_{vf})_d$ y devuelve el conjunto A, que cumple:

$$\begin{aligned}\exists\, T = \{(-, t_{vio}, t_{vfo}, -, -) \in X | \\ (t_{vio} \leq t_{vf}) \wedge (t_{vi} \leq t_{vfo})\} \wedge \\ A \subseteq T \quad \wedge \\ |A| = k \quad \wedge \\ \forall (o, -, -, -, -) \in A, \\ \forall (p, -, -, -, -) \in (T - A) : \\ d(q, o) \leq d(q, p)\end{aligned}$$

La variable q es el objeto que se consulta; k es la cantidad de elementos más cercanos a q que se quiere obtener y las variables t_{vi} y t_{vf} son el tiempo inicial y final respectivamente, del intervalo válido consultado. Los $-$ indican que dicho valor es irrelevante en su contexto. De manera similar, se puede plantear la consulta NN_k/-/rank utilizando el tiempo transaccional en lugar del tiempo válido.

4. NN_k**/point/-** y NN_k**/-/point** Consultas similares a las anteriores, pero con $t_{vi} = t_{vf}$.

5. Las consultas anteriores se pueden extender a las dos dimensiones temporales, generando consultas tales como **range/rank/rank**, NN_k**/rank/rank**, **range/point/rank** y las demás combinaciones. En la práctica para estos tipos se pueden utilizar algoritmos similares a los definidos para una sola dimensión temporal.

Una forma trivial de resolver las consultas métrico-temporales que incluyen búsqueda por rango (por ejemplo, los tipos 1 y 2 enumerados anteriormente) sin recorrer la base de datos completa es utilizando un índice métrico de la siguiente manera:

1. Realizar una búsqueda por similtud sobre el índice métrico

2. Realizar una búsqueda secuencial sobre el conjunto de objetos resultantes del paso anterior, para obtener los que cumplen con la restricción temporal

En el caso de las consultas tipo NN_k, no existen soluciones simples (que no sea la búsqueda secuencial, por supuesto). El problema es que, a diferencia de las búsquedas por rango, encontrar los k elementos requeridos depende de la interrelación del aspecto métrico con el aspecto temporal, por lo cual usualmente no es suficiente consultar el índice métrico una sóla vez y luego filtrar los elementos

temporalmente, ya que esto podría reducir la cantidad k de vecinos más cercanos buscados. En tal caso, es necesario realizar una nueva búsqueda sobre el índice métrico con un valor mayor para k y repetir el proceso de filtrado temporal. En el peor de los casos, este proceso se repite hasta que k alcanza a ser mayor o igual a la cantidad de elementos de la base de datos:

1. $r = 0$
2. $m = k$
3. Mientras $r \leq k$
   (a) Realizar una búsqueda por similtud $NN(m)$ sobre el índice métrico
   (b) Realizar una búsqueda secuencial sobre el conjunto de objetos resultantes del paso anterior, para obtener los que cumplen con la restricción temporal (la cantidad de elementos resultantes es r)
   (c) incrementar m

Otra posible solución, sería consultar primero el índice temporal para obtener los objetos que se encuentran dentro del intervalo consultado, pero luego habría que realizar una búsqueda por similitud en forma secuencial sobre estos elementos.

La desventaja que tienen estas soluciones es que no utilizan ambos aspectos a la vez para filtrar los objetos que no pueden formar parte del resultado. Por esta razón, tiene sentido definir métodos de acceso especialmente diseñados que aprovechen tanto la componente métrica como la temporal para descartar elementos.

3.1 Consultas range/rank/-

Este algoritmo está definido en [10] para funciones de distancia discretas, y aquí se presenta su adaptaciíon para distancias continuas. Cuando se realiza una consulta métrico-temporal, ya sea de tipo *range/rank/-* (rango de similitud / intervalo válido) o *range/point/-* (rango de similitud / instante válido), se procede de la siguiente manera: en cada nivel del árbol, se seleccionan los subárboles hijos del nodo que se está procesando, cuyos intervalos temporales se intersectan con el intervalo o instante de la consulta. De éstos, posteriormente se eligen los que cumplen con la restricción de similitud tomando en cuenta la firma de la consulta y el radio de búsqueda. Este procedimiento se repite hasta llegar al último nivel. Para cada hoja no descartada, luego de verificar la superposición temporal, se realiza una búsqueda secuencial sobre todos los elementos contenidos en las mismas, comparando tanto el aspecto temporal como la distancia de cada elemento a la consulta.

Formalmente, sea $(q, r, t_{iq}, t_{fq})_d$ una consulta métrico-temporal por rango de similitud y tiempo válido, $[t_{ini}, t_{fin}]$ el intervalo correspondiente al nodo que se está procesando, y $\{(d_1, h_1), (d_2, h_2), \ldots, (d_m, h_m)\}$) los hijos del nodo y sus distancias al pivote del nivel, primero se comprueba si $(t_{ini} \leq t_{fq}) \wedge (t_{iq} \leq t_{fin})$, es decir, si el intervalo de la consulta contiene al menos un instante en común con el intervalo del nodo. Si satisface esta condición, se recorren sus m nodos hijos, ingresando solo a los subárboles h_j que cumplan con la restricción de similitud. Los demás hijos se descartan.

Finalmente, cuando se alcanzan las hojas, se recorren los objetos contenidos en cada una de ellas y se seleccionan los que cumplen tanto con la condición temporal como con la condición de similitud $d(q, o_i) \leq r$.

El algoritmo de consulta por rango de similitud y tiempo válido (Figura 2) se define recursivamente y está dividido en dos funciones. En la primera se calcula la firma del objeto que se consulta –para que la firma se compute una sola vez– y luego se invoca a la función *Consultar*, que es la que realiza la búsqueda en sí. Esta segunda función recorre el árbol descartando las ramas que no cumplen con las restricciones temporales o métricas y procesando las demás. Cuando se alcanzan las hojas, se realiza una búsqueda secuencial sobre los objetos contenidos en las mismas, devolviendo aquellos que cumplen ambas condiciones.

Range/Rank/- $(q, r, t_{iq}, t_{fq})_d$
1. calcular f de q – f es la *firma de q*
2. **return** Consultar(q, r, t_{iq}, t_{fq}, 1, f, $raiz$)

donde **Consultar** se define recursivamente como:

Consultar$(q, r, t_{iq}, t_{fq}, n, f, x)$ – n es el nivel del nodo actual x
1. Sea $[t_{ix}, t_{fx}]$ el intervalo asociado al nodo x
2. resultado:=$\emptyset$
3. **if** $([t_{ix}, t_{fx}] \cap [t_{iq}, t_{fq}]) \neq \emptyset$ **then**
4. **if** esHoja(x) **then**
5. **for all** objeto ($o \in x$)
6. **if** $([t_{io}, t_{fo}] \cap [t_{iq}, t_{fq}] \neq \emptyset) \wedge (d(q, o) <= r)$ **then**
7. resultado :=resultado $\cup$ $\{o\}$
8. **else**
9. **for all** hijo ($h_i \in x$)
10. **if** $[f_n - r, f_n + r] \cap int_{ai} \neq \emptyset$ **then**
11. resultado :=resultado $\cup$ **Consultar**$(q, r, t_{iq}, t_{fq}, n + 1, f, h_i)$
12. **return** resultado

Figura 2: *FHQT$^+$-Temporal: pseudocódigo de consulta por rango e intervalo de tiempo*

3.2 Consultas NN_k/rank/-

En este trabajo se propone el siguiente algoritmo para resolver este tipo de consultas sobre el *FHQT$^+$-Temporal*. La estrategia es la siguiente. En primer lugar se calculan los nodos hoja que cumplen la restricción temporal, ordenados de menor a mayor de acuerdo al mínimo radio necesario para ser incluidos en una búsqueda por rango. El mínimo valor de radio para que una hoja sea visitada, es la diferencia máxima entre los valores de la firma y los intervalos correspondientes. Por ejemplo, en el *FHQT$^+$-Temporal* de la Figura 1, ante una consulta cuya firma es $(2, 1)$, la cuarta hoja (de izquierda a derecha) tendrá a 0 como radio mínimo para poder ser accedida, ya que 2 está incluido en el intervalo de distancias al primer pivote y 1 en el intervalo correspondiente al segundo pivote, mientras que para la quinta hoja el radio mínimo será $0{,}9$ (correspondiente al segundo pivote: $1{,}9 - 1 = 0{,}9$) y para la sexta hoja será $1{,}3$ ($2{,}3 - 1$), correspondiente también a la diferencia del segundo valor de la firma. Esta estrategia define un orden para la búsqueda de los más cercanos, que hace que sólo sea necesario recorrer una vez el árbol (normalmente en forma parcial), a diferencia de otras soluciones en las cuales se debe realizar una serie de búsquedas por rango incrementando el radio en un valor arbritario hasta encontrar los k elementos más cercanos. Luego de calcular las hojas con los candidatos, se realiza un recorrido secuencial de las mismas, teniendo en cuenta el orden anterior, hasta encontrar los k elementos buscados.

En la Figura 3 se muestra el algoritmo de consulta del vecino más cercano e intervalo temporal. El mismo consta de dos funciones. La función $CalcularHojas$ calcula y devuelve una lista de pares $(hoja, distancia)$ ordenada de menor a mayor por el segundo elemento que representa, como se expresó anteriormente, el radio mínimo para ser tomada en cuenta. Esta función realiza el primer filtro temporal. La función principal invoca a $CalcularHojas$ y recorre la lista de elementos resultantes controlando el aspecto temporal y construyendo una nueva lista, ordenada de menor a mayor por distancia a la consulta. Al final, se devuelven los k primeros elementos de esta lista, o un número menor si es que no existen k elementos que cumplan con la restricción temporal. Nótese que en la solución trivial, para este último caso habría que recorrer la base de da-

NN_k**/Rank/-** $(q, k, t_{iq}, t_{fq})_d$
1. calcular f de q – f es la *firma de q*
2. hojas:= CalcularHojas(t_{iq}, t_{fq}, 1, f, $raiz$, 0)
3. resultado:= () – lista vacia, de pares $(objeto, distancia)$
4. c:=1
5. **while** (c $\leq$ $length$(hojas)) $\wedge$
   (($length$(resultado) < k) $\vee$ (resultado$_{length(resultado)}$ .distancia > hojas$_c$ .distancia)) **do**
6. **for all** ($o \in hojas_c.hijos$)
7. **if** ($[t_{io}, t_{fo}] \cap [t_{iq}, t_{fq}]) \neq \emptyset$ **then**
8. agregar($(o, d(q, h)), resultado$)
9. c:=c+1.
10. **return** $primeros(k, resultado)$

donde **CalcularHojas** se define recursivamente como:

CalcularHojas$(t_{iq}, t_{fq}, n, f, x, distancia)$ – n es el nivel del nodo actual x
1. Sea $[t_{ix}, t_{fx}]$ el intervalo asociado al nodo x
2. resultado:=() – lista vacia, de pares $(hoja, distancia)$
3. **if** ($[t_{ix}, t_{fx}] \cap [t_{iq}, t_{fq}]) \neq \emptyset$ **then**
4. **if** esHoja(x) **then**
5. agregar($(x, distancia), resultado$)
6. **else**
7. **for all** hijo ($h \in x$)
8. **if** $f_n < int_{ah}.desde$ **then**
9. max:=$int_{ah}.desde - f_n$
10. **elsif** $f_n \geq int_{ah}.hasta$ **then**
11. max:=$f_n - int_{ah}.hasta$
12. **if** $max > distancia$ **then**
13. $distancia$:=max
14. **for all** $par \in$ CalcularHojas($t_{iq}, t_{fq}, n + 1, f, h, distancia$)
15. agregar($par, resultado$)
16. **return** resultado

Figura 3: *FHQT$^+$-Temporal: pseudocódigo de consulta NN_k e intervalo de tiempo*

tos completa e inclusive comparar algunos elementos más de una vez, mediante el uso reiterado del índice métrico con radios de búsqueda cada vez mayores.

4 Resultados experimentales

Para la evaluación experimental del algoritmo $NN_k/rank/-$ propuesto, se utilizó la base de datos $NASA$ [7] (conjunto de 40.150 vectores 20-dimensionales de números reales, que representan características de imágenes), agregándole a cada elemento un intervalo o instante temporal dentro del rango [1, 1000]. Mediante un proceso aleatorio se generaron lotes de 1.000, 5.000, 10.000 y 20.000 elementos como bases de datos de prueba, y cuatro lotes de 100 elementos que se utilizaron como consultas (instantáneas, intervalos del 10 % del tiempo en promedio, del 25 % y del 50 %). Las cantidades de vecinos más cercanos consideradas fueron k=1, k=5 y k=10.

Como función de distancia se utilizó la distancia euclidiana que es la métrica usual sobre esta base de datos para realizar pruebas por similitud. En estas pruebas sólo se tomó en cuenta como variable de costo la cantidad de evaluaciones de la función de distancia, ya que la estructura se mantuvo en memoria principal.

4.1 Comparación con la solución trivial

En la Figura 4 se grafican los resultados de la evaluación del costo del algoritmo propuesto versus la solución trivial planteada anteriormente, en función del tamaño de la base de datos, para k=1 y k=5, considerando en promedio el 10 % del intervalo temporal total como intervalo de consulta. En todos los casos el algoritmo $NN_k/rank/-$ propuesto supera ampliamente en eficiencia a la solución trivial. Por ejemplo, para 1.000 elementos y k=1, nuestra propuesta requiere comparar el 7,7 % de los objetos, mientras que la solución trivial necesita más del 24 %. Esta diferencia es mucho mayor aún cuando la base de datos crece en tamaño: para 10.000 elementos la performance de la solución trivial se degrada hasta requerir en promedio, evaluar la función de distancia para casi la mitad de los elementos del conjunto. Este mismo caso es resuelto por el algoritmo propuesto evaluando solamente el 2,24 % de los elementos. El factor más importante en la disminución de la eficiencia de la solución trivial, es que en algunas consultas la cantidad de elementos existentes en el intervalo ingresado es menor que k, por lo cual es necesario realizar repetidas búsquedas sobre el índice métrico incrementando k hasta alcanzar la cantidad total de elementos de la base de datos, para asegurar que no haya más resultados.

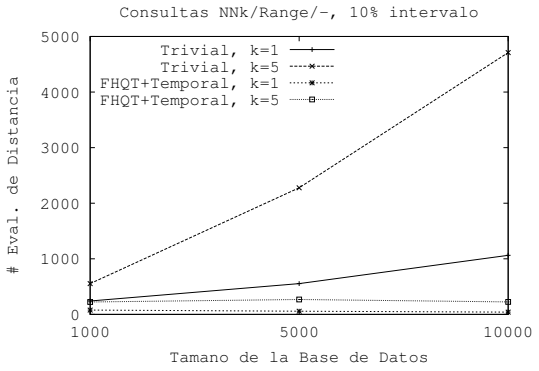

Figura 4: *Comparación de costos entre el algoritmo NN_k/point/- propuesto y la solución trivial, en función del tamaño de la base de datos*

4.2 Variación del Costo en Función de k

En las Figuras 5 y 6 se muestran las curvas de costos correspondientes al algoritmo propuesto, en función de la cantidad de vecinos más cercanos buscada. La primer figura representa las consultas NN_k/point/-, es decir, instantáneas, mientras que la segunda es de tipo NN_k/rank/- con un intervalo de tiempo promedio de consulta igual al 50 % del total.

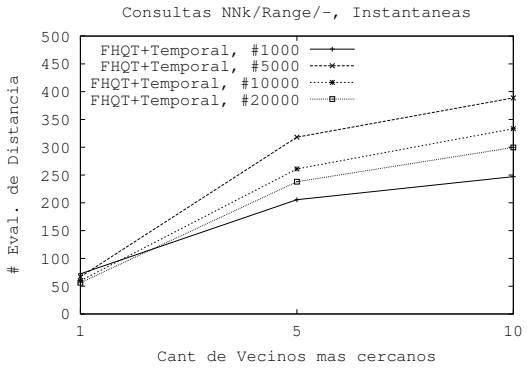

Figura 5: *Costo del algoritmo propuesto, consultas NN_k/point/- en función de k*

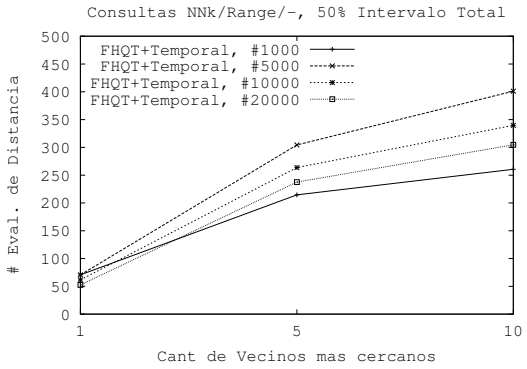

Figura 6: *Costo del algoritmo propuesto, consultas NN_k/rank/- en función de k*

Para ambos casos el comportamiento del algoritmo en función de k, es similar. El costo crece cuando la cantidad de vecinos más cercanos es mayor, pero este crecimiento es comparativamente menor. Es decir, cuando k aumenta de 1 a 5, el costo es solamente 4,2 veces mayor, y para un aumento de k de 1 a 10, la cantidad de evaluaciones sólo aumenta 5,33

veces. Por lo tanto el algoritmo es robusto ante el crecimiento de k.

4.3 Variación del Costo en Función de Tamaño del Intervalo

En las Figuras 7 y 8 se muestran las curvas de costos, en función del tamaño del intervalo de tiempo consultado, correspondientes a k=1 y k=10. Como se ve en los gráficos, el costo inicialmente tiene un pequeño incremento y luego disminuye a medida que aumenta el intervalo de consulta, aunque en general las diferencias no son significativas. Este comportamiento se debe principalmente a la distribución temporal de los elementos

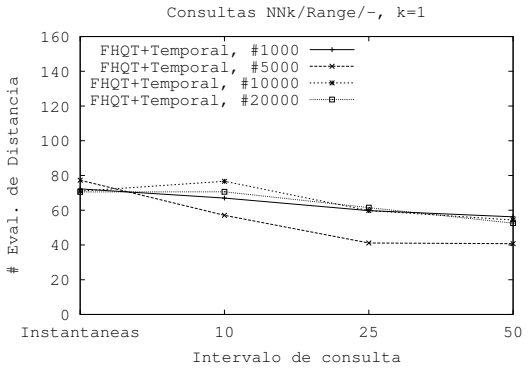

Figura 7: *Costo del algoritmo propuesto, consultas NN_k/rank/- en función del intervalo de tiempo consultado, k=1*

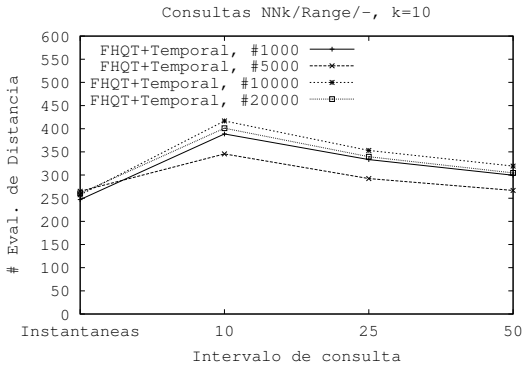

Figura 8: *Costo del algoritmo propuesto, consultas NN_k/rank/- en función del intervalo de tiempo consultado, k=10*

Se concluye también, que el algoritmo NN_k/rank/- muestra un comportamiento estable ante las variaciones del intervalo de tiempo de la consulta.

5 Discusión

Como consecuencia de este estudio, han surgido cuestiones a resolver en futuras investigaciones debido a limitaciones de los métodos propuestos y a necesidades de expansión de los mismos. A continuación se describen brevemente las principales:

- En todos los casos las consultas se plantearon para una sola dimensión temporal. Es necesario extender los algoritmos para que contemplen las dimensiones de tiempo válido y transaccional a la vez.

- El algoritmo para calcular los vecinos más cercanos y los resultados experimentales de este tipo de consulta fueron diseñados para el índice *FHQT-Temporal* y su variante para distancias continuas. Se planea adaptar este algoritmo a los índices *Event-FHQT*, *Historical-FHQT* y *Pivot-FHQT*.

- El aspecto temporal de las consultas planteadas implica en todos los casos superposición temporal entre los intervalos/instantes involucrados. Existen otros tipos de consultas temporales tales como “before” o “after” (es decir, que un intervalo o instante se encuentre “antes” o “después” de otro) que no se tuvieron en cuenta. La notación *similarity/valid/transaction* introducida en este estudio se podría extender para admitir éstos y otros tipos de consultas temporales.

- Todos los algoritmos de consultas desarrollados hasta el momento suponen que los índices residen en memoria principal. Se requiere investigar su adaptación a memoria secundaria ya que para bases de datos muy grandes, la primer opción sería muy costosa o imposible de realizar.

- Aún no se ha estudiado la posibilidad de utilizar paralelismo para resolver las consultas métrico-temporales.

6 Conclusiones

Las resolución de consultas en grandes bases de datos constituye un problema largamente estudiado y con muy buenas soluciones en algunos casos. En este trabajo se estudiaron las consultas métrico temporales, que son consultas por similitud sobre grandes bases de datos de objetos no estructurados, que involucran al menos una dimensión temporal. Para ello se presentaron los tipos más importantes de consultas sobre este modelo, se adaptó a distancia continua el algoritmo de búsqueda por rango y tiempo definido para el índice *FHQT-Temporal*, se propuso un algoritmo para resolver las consultas de tipo NN_k/rank/- y se presentaron resultados experimentales que muestran que es significativamente más eficiente que la solución trivial, y es estable ante la modificación de la cantidad de vecinos más cercanos a devolver y ante las variaciones de los intervalos temporales de consulta.

Referencias

[1] A. De Battista, A. Pascal, N. Herrera, and G. Gutierrez. Metric-temporal access methods. *Journal of Computer Science & Technology,*, 10(2):54–60, 2010.

[2] 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.

[3] 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.

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

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

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

[7] K. Figueroa, G. Navarro, and E. Chávez. Metric spaces library, 2007. Available at `http://www.sisap.org/Metric_Space_Library.html`.

[8] 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.

[9] A. Pascal, A. De Battista, G. Gutierrez, and N. Herrera. Indice metrico-temporal event-fhqt. In *Actas del XIIII Congreso Argentino de Ciencias de la Computación*, La Rioja, Argentina, 2008.

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

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

[12] R. Snodgrass and C.S. Jensen. Private communication. 1996.

Datos de Contacto

Andrés Pascal. Universidad Tecnológica Nacional, Facultad Regional Concepción del Uruguay, Dpto. Sistemas de Información. Ing. Pereira 676, Concepción del Uruguay, Entre Ríos, Argentina. e-mail: pascala@frcu.utn.edu.ar.

Anabella De Battista. Universidad Tecnológica Nacional, Facultad Regional Concepción del Uruguay, Dpto. Sistemas de Información. Ing. Pereira 676, Concepción del Uruguay, Entre Ríos, Argentina. e-mail: debattistaa@frcu.utn.edu.ar.
Norma Herrera. Universidad Nacional de San Luis, Departamento de Informática. Ejército de los Andes 950, San Luis, Argentina. e-mail: nherrera@unsl.edu.ar.