Métodos de Acceso y Procesamiento de Consultas Métrico-Temporales

Andrés Jorge Pascal

Concepción del Uruguay, Entre Ríos, Argentina
Diciembre de 2012

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL CONCEPCIÓN DEL URUGUAY
DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN

Métodos de Acceso y Procesamiento de Consultas Métrico-Temporales

ANDRÉS JORGE PASCAL

DIRECTORA: MG. CS. NORMA EDITH HERRERA
ASESOR: DR. GILBERTO GUTIÉRREZ

Tesis para optar al grado de
Magister en Ciencias de la Computación
con orientación Bases de Datos

A mi Familia más cercana, Dana, Aixa, Andrea.
A mi Gran Familia.
A mis Amigos GFU, SOA, TMC, ARC.

AGRADECIMIENTOS

En primer lugar quiero agradecer a mi directora Norma Herrera por su guía, aliento, paciencia, dedicación y por sus observaciones siempre acertadas, justas y precisas. Y además por su balance entre la motivación personal y la exigencia que requiere un trabajo de esta clase.

Agradezco profundamente al Dr. Gilberto Gutiérrez por su orientación en los temas específicos de su incumbecia, por el nivel de detalle con el cual realiza sus observaciones y por su confianza en mi trabajo.

Un agradecimiento especial a mi amiga, colega y compañera en el apasionante mundo de la investigación, la Mg Anabella De Battista, por su apoyo y colaboración.

Por otro lado, agradezco a la Universidad Tecnológica Nacional, a la Facultad Regional Concepción del Uruguay y al Departamento de Ingeniería en Sistemas de Información, por brindarme la posibilidad de concretar esta etapa superior de mi formación académica.

Agradezco a mi Familia y a mis Amigos, por su tolerancia, apoyo y aliento, en especial durante las largas horas de escritura de esta Tesis.

Y a todos los que de una u otra manera colaboraron y me brindaron su apoyo para llevar a cabo este proyecto.

Índice general

Índice de figuras

Apéndice A

Índice de cuadros

Capítulo 1

Introducción

1.1. Introducción

Con el avance de las ciencias y tecnologías de la información, surge la necesidad cada vez más importante de almacenar grandes cantidades de datos y de realizar búsquedas para obtener información para la toma de decisiones. Hasta hace una década atrás, los tipos de datos mayormente utilizados eran datos relativamente simples, de tamaño fijo y sobre los cuales se realizaban búsqueda exactas, por rango o por patrón o prefijo en el caso de cadenas de caracteres. Estos datos se almacenaban como registros organizados en campos que contenían valores completamente comparables (*datos estructurados*). Las bases de datos estructurados se conocen en la actualidad con el nombre de *bases de datos clásicas*.

Si bien las bases de datos clásicas aun son de interés, resultan insuficientes para afrontar todas las necesidades actuales de almacenamiento y recuperación de información. Hoy en día existen requerimientos importantes de almacenamiento y recuperación de datos semi-estructurados o no estructurados, tales como imágenes (huellas digitales, rostros, paisajes, pinturas, logos, etc), sonido (música, voces, etc), video (filmaciones de cámaras de seguridad, películas, etc), texto (secuencias de ADN, secuencias de proteínas, textos en lenguaje natural, etc), documentos XML, entre otros. Estos datos tienen la característica de que no pueden ser organizados como datos estructurados, y aun cuando pudiera hacerse, las búsquedas exactas normalmente no tienen sentido, por lo que se requiere poder resolver otros tipos de consultas. Debido al tamaño de estas bases de datos, no es práctico realizar un recorrido exhaustivo de las mismas para resolver una consulta, por lo cual es necesario el uso de estructuras de datos (índices) que agilicen las búsquedas.

Otro aspecto importante a considerar es que las bases de datos clásicas modelan solo un instante dado de la realidad. En estos últimos quince años se han realizado avances importantes para superar esta limitación, dando origen a las bases de datos temporales, que permiten gestionar con cierto grado de automatización más de un instante a la vez y realizar consultas sobre uno o varios de dichos instantes en forma eficiente.

Estas necesidades actuales han dado origen a nuevos modelos de bases de datos, que se integran a los modelos existentes para ampliar el campo de aplicación de los almacenes de datos, pero estos nuevos modelos representan partes de la realidad y deben ser integrados para obtener un modelo más fiel y abarcativo de la misma. En esta Tesis se presenta un modelo que integra los modelos Métrico y Temporal.

El *Modelo Métrico* se basa en los Espacios Métricos, que facilitan la resolución eficiente de consultas por similitud sobre datos no estructurados o semi-estructurados. Este modelo caracteriza a la realidad como un universo de objetos U y una función métrica de distancia o similitud d que mide el grado de similitud (estrictamente, de diferencia) entre los objetos del universo. Las consultas por similitud usuales en espacios métricos son la *búsqueda por rango*, que toma como entrada un objeto de consulta y un radio y recupera todos aquellos objetos de la base de datos con distancia al objeto menor al radio, y la *búsqueda de los k vecinos más cercanos*, que devuelve los k elementos con menor distancia al objeto de consulta. Por ejemplo, algunas consultas por similitud son: *devolver todas las huellas digitales similares a una dada, con una radio de tolerancia r* o *devolver las k huellas digitales con mayor parecido a una dada*.

El Modelo Temporal permite 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 otra dimensión. Este modelo soporta dos tipos temporales: tiempo válido (el momento o rango de tiempo en el cuál la información es cierta) y tiempo transaccional (el instante de tiempo en el cuál la base de datos toma conocimiento de dicha información). Ejemplos de consultas temporales son: *devolver los valores los parámetros con los cuales se calcularon los sueldos de los empleados en marzo de 2007* o *devolver la evolución de los sueldos de los empleados desde marzo de 2007 a marzo de 2012*.

Si bien estos modelos permiten abordar situaciones que las bases de datos clásicas no pueden manejar, existen problemas en los cuales resulta de interés combinar el aspecto métrico con el temporal, por ejemplo:

- En un museo donde se registran las huellas digitales de los individuos que ingresan y egresan junto con su fecha y hora, una consulta de interés es *devolver todas las huellas digitales similares a una dada (dentro de un radio de tolerancia), de las personas que estuvieron en el museo el día 2 de marzo de 2012 entre las 15 y las 20 hs*
- En un punto de control en una ruta, donde se toman automáticamente fotos de los vehículos que pasan, junto con su fecha y hora, una consulta de interés es *devolver los 10 vehículos con mayor similitud a uno dado, que pasaron por el punto de control los días 5 o 6 de mayo de 2012*
- En un video de seguridad de un edificio, una consulta de interés es *devolver los fragmentos de video que contienen imágenes de personas con rostros similares a uno dado (dentro de un cierto radio de tolerancia), que circularon por el lugar durante el primer fin de semana de enero de 2012*. Obviamente en este caso la consulta está planteada informalmente y para que sea una consulta válida se debe especificar con mayor detalle.

En todos estos ejemplos es necesario buscar objetos que sean similares a uno dado, pero

dentro de un intervalo de tiempo. Tanto para las consultas métricas como para las temporales existen índices que hacen más eficiente la recuperación de la información, pero ante situaciones como las anteriormente planteadas, la única solución hasta hace un tiempo era realizar la búsqueda de los objetos en un índice métrico y otro temporal por separado para luego calcular la intersección de los conjuntos resultantes. Este método es mejor que realizar un recorrido secuencial, pero dista de ser una buena solución. En consecuencia, se considera conveniente el desarrollo de índices que permitan resolver con eficiencia este tipo de consultas combinadas.

En esta Tesis se estudia el modelo Métrico-Temporal, que permite la representación de situaciones en las cuales se requiere responder a consultas que consideran tanto el aspecto métrico como el temporal, y se presenta el diseño de estructuras que hacen más eficiente la resolución de estos tipos de búsquedas y que además permiten realizar consultas métricas puras y consultas temporales puras.

1.2. Aportes de la tesis

Podemos resumir los aportes de esta Tesis en los siguientes puntos:

- Definición del *Modelo Métrico-Temporal*. Se define formalmente este nuevo modelo y se caracterizan los problemas que resuelve. También se describen los tipos de consultas asociados al modelo y se analizan los costos de las búsquedas.
- Diseño del índice métrico-temporal *FHQT-Temporal*. Se presenta una estructura de datos basada en el índice métrico FHQT al que se le añade el aspecto temporal para realizar consultas por similitud y temporales combinadas. Se muestran los algoritmos de construcción y búsqueda y se realiza su evaluación experimental. Este índice está orientado a objetos instantáneos, es decir, que solo poseen un instante de tiempo asociado.
- Diseño del índice métrico-temporal *Event-FHQT* . Se presenta un índice métrico-temporal orientado a eventos, que permite trabajar con objetos que tienen asociado un intervalo de tiempo Se muestran los algoritmos de construcción y búsqueda y se realiza su evaluación experimental.

Los resultados logrados han sido publicados en actas de congresos y en revistas durante el proceso de elaboración de esta Tesis, quedando aún algunos puntos menores sin publicar. El detalle de estas publicaciones es el siguiente:

- A. De Battista, A. Pascal, G. Gutierrez, N. Herrera. Búsqueda en bases de datos métricas-temporales. En *Actas del VIII Workshop de Investigadores en Ciencias de la Computación (WICC)*, Buenos Aires, Argentina, 2006 [BPGH06].
- A. Pascal, A. De Battista, G. Gutierrez, N. Herrera. Procesamiento de consultas Métrico-Temporales. En *Actas de la XXXIII Conferencia Latinoamericana de Informática (CLEI 2007)*, San José, Costa Rica, 2007 [PBGH07].

- A. De Battista, A. Pascal, G. Gutierrez, N. Herrera. Índices para Bases de Datos Métrico-Temporales. En *Actas del X Workshop de Investigadores en Ciencias de la Computación (WICC)*, La Pampa, Argentina, 2008 [BPGH08].
- A. Pascal, A. De Battista, G. Gutierrez, N. Herrera. Índice Métrico-Temporal Event-FHQT. En *Actas del XIV Congreso Argentino de Ciencias de la Computación (CACIC)*, Chilecito, La Rioja, Argentina, 2008 [PBGH08].
- A. De Battista, A. Pascal, G. Gutierrez, N. Herrera. Bases de Datos Métrico-Temporales. En *Actas del XI Workshop de Investigadores en Ciencias de la Computación (WICC)*, San Juan, Argentina, 2009 [BPGH09].
- A. Pascal, A. De Battista, G. Gutierrez, N. Herrera. Metric-Temporal Access Methods. En el *Journal of Computer Science and Technology*, Vol. 10 - No. 2, pag 54-60, 2010 [BPHG10].

1.3. Organización del Informe

Este informe está organizado en séis capítulos: en los primeros tres capítulos se desarrolla el marco teórico y en los tres restantes se presentan los aportes de esta tesis, la evaluación experimental y las conclusiones.

En el Capítulo 2 se introduce el Modelo Métrico, incluyendo la definición, características y usos y limitaciones de los Espacios Métricos como base para resolver consultas por similitud de manera eficiente. También se presentan los métodos de acceso más importantes para las consultas por similitud y en especial se describe con mayor detalle el *FHQT*, que fue utilizado como base para el diseño de los dos nuevos índices métrico-temporales presentados en esta tesis.

En el Capítulo 3 se brinda una introducción al Modelo Temporal, su definción, clases de dimensiones temporales, tipos de bases de datos temporales y tipos de consulta. Se describen métodos de acceso temporales y en particular se decribe el índice espacio-temporal $SEST_L$, del cual se tomaron ideas para desarrollar el índice *Event-FHQT*.

En el Capítulo 4 se realizan los aportes principales de esta Tesis. Se define el Modelo Métrico-Temporal, se caracterizan los problemas que resuelve, se definen los tipos de consultas básicos y se presentan las nuevas estructuras de acceso: el *FHQT-Temporal* y el *Event-FHQT*, que permiten realizar búsqueda por similitud y temporal con eficiencia.

Posteriormente, en el Capítulo 5 se realiza la evaluación experimental de ambos índices y en el Capítulo 6 se presentan las conclusiones del informe y se proponen trabajos futuros.

Capítulo 2

El Modelo Métrico

ï»¿

2.1. Introducción

Las bases de datos tradicionales están orientadas a trabajar con datos simples, usualmente números o cadenas, que se estructuran para formar construcciones de mayor complejidad. En el caso del modelo relacional, sin dudas el de mayor uso en la actualidad, estos datos se componen en conjuntos de $n - uplas$. Estas $n - uplas$ mantienen una estructura bien definida y permiten el uso de índices eficientes para realizar las consultas requeridas. Para ello, se debe poder establecer un orden total entre los valores que intervienen en los índices; orden que es utilizado para descartar elementos sin necesidad de compararlos con el objeto de búsqueda. Por otro lado, las bases de datos tradicionales están orientadas al concepto de búsqueda exacta, por rango o a lo sumo, en el caso de las cadenas, por prefijo. Estos son los tipos de consulta en los cuales se pueden utilizar índices tales como alguno de la familia de los $B - Tree$, para disminuir considerablemente el costo (en tiempo) de la búsqueda. Otros tipos de consulta como la búsqueda por subcadena o patrón, no pueden ser indexadas por lo cual es necesario recorrer el conjunto de datos completo.

En las últimas dos décadas han surgido otras clases de objetos que no se pueden modelar de esta manera, como los nombrados anteriormente: imágenes (con todas sus variantes), sonido, video, texto, documentos XML, datos geométricos, secuencias de ADN, series temporales, etc. Hasta hace unos años, estos objetos solo se podían almacenar, pero no existían formas de realizar consultas que los involucren, ya que además de no tener una estructura fija, no tiene sentido compararlos por igualdad y no se puede establecer un orden total entre ellos. Por ejemplo, en el caso de fotos de pinturas, no sirve comparar pixel a pixel dos de estas imágenes para saber si corresponden al mismo objeto. Sus resoluciones pueden ser distintas, el ángulo con que fueron tomadas las fotos seguramente es diferente y tampoco tendrían el mismo nivel de luminosidad. En las aplicaciones que contienen estos tipos de objetos, algunas consultas usuales de interés son:

- Encontrar todos los objetos que sean similares (con cierto nivel de tolerancia) a uno dado.
- Encontrar los k objetos con mayor similitud a uno dado.

Esta clase de búsqueda se denomina *Búsqueda por Similitud* y posee un campo de aplicación cada vez más amplio. Algunas aplicaciones son:

Consultas por contenido Cuando se requiere mantener datos compuestos no estructurados, la solución trivial para poder consultarlos es asociarles un nombre, descripción o identificador y realizar la consulta sobre estos campos, sin embargo esta solución no es aplicable cuando la consulta es el objeto mismo, como por ejemplo, una huella digital. En estos casos surge la necesidad de recuperar la información *por contenido*. Ejemplos de consultas por contenido son:

- Búsqueda de personas a través de una fotografía de sus rostros.
- Búsqueda de huellas digitales para identificar sospechosos de un delito.
- Búsqueda de pinturas para identificar y recuperar información sobre las mismas.
- Búsqueda de logotipos, para asegurar que no haya otro similar ya registrado.
- Búsqueda de canciones o fragmentos de canciones similares a una dada.
- Búsqueda de videos, en base a un fragmento o imagen.

Recuperación de texto La recuperación de texto constituye un problema de características similares a los planteados anteriormente. Esto es debido a que los objetos textuales no están estructurados como para facilitar la ejecución de consultas que son de interés, como por ejemplo, búsquedas por conceptos semánticos. Este problema ha sido estudiado desde hace ya largo tiempo y uno de los métodos propuestos que resuelve este problema consiste en la definición de una función de distancia específica para comparar texto que es utilizada para realizar consultas por similitud.

Reconocimiento de patrones y aproximación de funciones En ciertas situaciones es necesario contar con un método automático de clasificación de nuevos elementos en base a un conjunto ya clasificado. Esta es una forma posible de reconocimiento de patrones. Para ello se cuenta con una base de datos de objetos asociados a sus clases y cuando se presenta un nuevo objeto se utiliza un *aproximador* que determina a cual clase pertenece. Existen distintas estrategias para construir este aproximador, algunas basadas en redes neuronales y funciones difusas, pero otra opción válida y que además no requiere entrenamiento, es determinar el vecino más cercano del nuevo elemento (con el que posee mayor similitud) y asignarle la misma clase.

Biología computacional Las secuencias de proteínas y el ADN constituyen el objeto de estudio de la biología molecular. Estas secuencias usualmente se modelan como texto y se presenta el problema de encontrar una subsecuencia que sea similar a una secuencia dada, ya que casi siempre existen diferencias menores en los fragmentos que describen funciones similares correspondientes a distintas especies o inclusive a distintos individuos de la misma especie. Como en los casos anteriores, una solución posible a este problema es el empleo de espacios métricos como base para realizar consultas por similitud.

Todas estas situaciones se pueden representar a través de un formalismo matemático denominado *Espacio Métrico*, que tiene como característica la existencia de un universo de objetos y de una función, llamada *función de distancia*, que calcula el grado de similitud (o diferencia) entre los objetos del universo.

Formalmente, un *Espacio Métrico* es un par (U, d) donde U es el universo de objetos válidos del espacio y $d : U \times U \longrightarrow R^+$ es una métrica definida entre los elementos de U que mide la similitud (estrictamente hablando, su diferencia) entre los mismos. Cuanto menor es el valor de la función para dos objetos dados, más cercanos o similares son éstos.

Para que el espacio sea métrico, la función de distancia d debe poseer las propiedades características de una métrica:

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

Bajo esta definición, se considera que dos elementos son iguales si su distancia es 0, por lo tanto es natural que las distancias sean positivas ya que una distancia negativa indicaría que hay dos elementos con mayor similitud que dos elementos iguales. La reflexividad indica que el mayor nivel de similitud posible se alcanza entre dos elementos iguales. La simetría expresa que la diferencia o similitud es única para todo par de elementos, y por último, la desigualdad triangular determina que la diferencia directa entre dos elementos siempre será menor o igual que la diferencia indirecta entre los mismos, es decir, pasando por un tercero. Es esta última propiedad una de las más importantes durante la búsqueda en espacios métricos dado que permite descartar elementos sin necesidad de compararlos con la consulta, como veremos más adelante.

Si d no satisface la positividad estricta, se habla de un espacio *pseudo métrico* y se puede adaptar para que mantenga las propiedades relevantes para este tipo de aplicaciones. En caso de que la función de distancia no cumpla con la simetría (*espacio cuasi-métrico*), se puede definir una nueva función d', que sea simétrica, haciendo: $d'(x, y) = d(x, y) + d(y, x)$. También se permite relajar la desigualdad triangular haciendo que solo cumpla $d(x, y) \leq \alpha d(x, z) + \beta d(z, y) + \gamma$. En estos casos es posible seguir usando los mismos algoritmos de espacios métricos previo escalamiento.

2.2. Consultas por similitud

Existen tres tipos genéricos de consultas que resultan de interés en espacios métricos [CNBYM01]. Sea $X \subseteq U$ un subconjunto del universo al cual llamaremos *base de datos* y $n = |X|$ su cardinalidad, se define:

- *Búsqueda por rango* $(q, r)_d$: donde $q \in U$ es el objeto de consulta o *query* y r es un número real que representa el radio de tolerancia, devuelve el conjunto de elementos que se encuentran a distancia de q menor o a lo sumo igual a r. Note que esta consulta podría retornar un conjunto vacío. Formalmente:

$$(q, r)_d = \{x \in X : d(q, x) \leq r\} \tag{2.1}$$

- *Búsqueda del vecino más cercano* $NN(q)$: donde $q \in U$ es el objeto de consulta, devuelve el elemento más cercano a q. En caso de que haya más de un elemento a la misma distancia, podría retornar un conjunto de elementos. Si la base de datos no es vacía, esta consulta siempre tiene al menos un objeto resultante. Formalmente:

$$NN(q) = \{x \in X : \forall y \in X, d(q, x) \leq d(q, y)\} \tag{2.2}$$

- *Búsqueda de los* k *vecinos más cercanos* $NN_k(q)$: esta consulta es similar a la anterior, pero generalizada a los primeros k elementos más cercanos. Formalmente, la consulta devuelve el conjunto $A \subseteq X$ tal que:

$$|A| = k \wedge \forall x \in A, y \in (X - A) : d(q, x) \leq d(q, y) \tag{2.3}$$

Para medir la eficiencia en tiempo de las consultas, hay tres factores a tener en cuenta. El primero en importancia suele ser el costo de cálculo de la función de distancia, ya que las funciones que calculan similitud entre objetos requieren muchos recursos para su procesamiento. El segundo es la cantidad de accesos a almacenamiento secundario (tiempo de E/S), que puede ser importante debido a que estos accesos son significativamente más lentos que el acceso a memoria principal. Por último, hay que considerar también el tiempo extra de CPU, es decir, el resto de las instrucciones del algoritmo de búsqueda. Por lo tanto, podemos calcular el costo en tiempo T de la siguiente manera:

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

El tiempo extra de CPU se puede descartar cuando se mide la eficiencia de la búsqueda porque normalmente es de un orden computacional mucho menor que los otros dos factores. Cuando los algoritmos de búsqueda realizan su procesamiento en memoria principal, el tiempo de E/S es nulo, por lo cual en estas situaciones para analizar la eficiencia de los algoritmos de consulta, solo se tiene en cuenta la cantidad de evaluaciones de la función de distancia. Este es el caso de los índices propuestos en este trabajo, por lo cual todas las mediciones sobre los índices propuestos están calculadas en cantidad de evaluaciones de d.

Dado el tamaño creciente de las bases de datos actuales y teniendo en cuenta el alto costo de cálculo de la función de distancia, no es operativo realizar una búsqueda exhaustiva sobre todos los elementos del conjunto de datos. Para ello se deben utilizar estructuras auxiliares (*índices o métodos de acceso*) que descarten elementos sin necesidad de compararlos con el objeto de consulta.

2.3. Funciones de Distancia

Las medidas de distancia en los espacios métricos representan el grado de similitud entre dos elementos en un dominio dado. Cuando se utilizan funciones de distancia como medida de similitud, los objetos a comparar no son datos simples, sino estructuras con cierta complejidad. Por ejemplo, secuencias de caracteres, vectores n-dimensionales, árboles o conjuntos. Los vectores n-dimensionales o vectores de características, contienen valores de propiedades, atributos o características de los objetos que representan. Una imagen, por ejemplo, se puede modelar a través de un vector que constituye el histograma de colores de la imagen; si la imagen contiene 16 colores, se puede obtener un vector con 16 elementos cuyos valores definen la cantidad de pixels de cada color contenidos en la imagen. En este caso el espacio métrico es también un *espacio vectorial* y existen varias funciones de distancia disponibles aplicables a este modelo. Si bien las medidas de distancia se definen específicamente para resolver cada problema particular, existen algunas funciones y familias de funciones conocidas que se pueden utilizar en varios dominios. Dependiendo del conjunto de valores que retornan, las funciones de distancia se clasifican en *discretas*, que retornan un conjunto predefinido y normalmente pequeño de valores, y *continuas*, que devuelven un conjunto potencialmente infinito o muy grande de valores. A continuación se presentan algunas de las funciones más importantes. Una lista más completa se puede encontrar en [Cha07].

2.3.1. Distancias de Minkowski

La familia de funciones más conocida y utilizada sobre espacios vectoriales, es la familia L_p de *Distancias de Minkowski*, definida sobre vectores de números reales como:

$$L_p[(x_1, ..., x_n), (y_1, ..., y_n)] = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$$

Algunos casos particulares de suma utilidad de esta familia son:

- L_1, conocida como *Distancia de Manhattan*.

$$L_1[(x_1, ..., x_n), (y_1, ..., y_n)] = \sum_{i=1}^{n} |x_i - y_i|$$

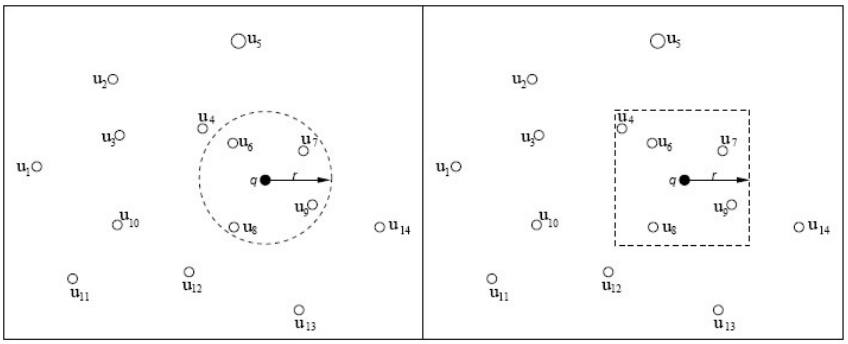

Figura 2.1: *Ejemplos de búsquedas por rango $(q, r)_d$, con d=L_2 (izquierda) y d = L_∞ (derecha)*

- L_2, conocida como *Distancia Euclidiana*

$$L_2[(x_1, ..., x_n), (y_1, ..., y_n)] = \sqrt{(\sum_{i=1}^{n} |x_i - y_i|^2)}$$

- L_∞, conocida como *Distancia Máxima*, que corresponde al límite de p tendiendo a infinito

$$L_\infty[(x_1, ..., x_n), (y_1, ..., y_n)] = \max_{i=1}^{n}(|x_i - y_i|)$$

La Figura 2.1 muestra un ejemplo de búsqueda por rango $(q, r)_d$ sobre un conjunto de puntos en R^2, utilizando como función de distancia L_2 y L_∞. La línea punteada identifica aquellos puntos que están a distancia r de q. Como consecuencia, todos los puntos que caen dentro de estas líneas conforman la respuesta a la búsqueda. La distancia L_2, se corresponde con nuestra noción de distancia espacial (en un espacio euclidiano). Las búsquedas que utilizan L_∞ se corresponden con la búsqueda por rango clásica, donde el rango es un hiper-rectángulo k-dimensional.

2.3.2. Distancia de Forma Cuadrática

En muchas aplicaciones que utilizan vectores de datos, los valores de las distintas dimensiones no son totalmente independientes. En estos casos las distancias de Minskowski no son aplicables ya que no tienen en cuenta la existencia de relaciones entre dimensiones. La *Distancia de Forma Cuadrática* utiliza una matríz de pesos que representa el grado de relación entre dos componentes x a y de los vectores X e Y respectivamente. Sean x e y dos vectores de n dimensiones, la siguiente expresión representa la *Distancia de Forma Cuadrática* generalizada d_M.

$$d_M(x, y) = \sqrt{((x - y)^T.M.(x - y))}$$

donde el superíndice T denota la transposición de vectores y

$$M = [m_{i,j}]$$

es la matríz de pesos. Si M es la matríz identidad, esta distancia se transforma en la distancia euclidiana.

2.3.3. Distancia de Edición

El grado de similitud entre dos cadenas de símbolos w y v, se puede medir efectivamente mediante la *Distancia de Edición* o *Distancia de Levenshtein*, definida como la menor cantidad de operaciones de edición necesarias para transformar la cadena w en v (o viceversa). Las operaciones consideradas son:

- Inserción(w, s, i): agrega el símbolo s en la posición i de la cadena w.
- Eliminación(w, i): borra el elemento que se encuentra en la posición i de la cadena w.
- Reemplazo(w, s, i): reemplaza el elemento que se encuentra en la posición i de la cadena w, por el símbolo s

Esta función se puede generalizar asignando pesos (números reales positivos) a cada tipo de operación. Por ejemplo, es común que el costo de la operación de reemplazo sea mayor que el de las operaciones de inserción y eliminación, ya que el reemplazo se puede simular mediante dos de estas. Otra variante es la asignación de pesos a operaciones sobre símbolos específicos, por ejemplo, la eliminación de la letra *h*, puede tener menor peso que la eliminación de la letra *a* cuando se trata del lenguaje natural español o castellano. Lo mismo sucede con el reemplazo de la letra *c* por la letra *s*, o *v* por *b*, ya que se pronuncian con fonética similar. En cualquier caso, para que se mantenga el espacio métrico, se debe asegurar que estas variantes preserven las propiedades de toda métrica.

La *Distancia de Damerau-Levenshtein* es una variante de la distancia de edición, que agrega al conjunto de operaciones básicas anteriormente nombradas la transposición, es decir, el intercambio posicional de dos símbolos adyacentes de la cadena.

2.3.4. Distancia de Edición de Árboles

La distancia de edición anteriormente nombrada se utiliza sobre secuencias o listas, y se puede adaptar para estructuras no-lineales como los árboles. La *Distancia de Edición de Árboles* se define como el mínimo costo necesario para convertir una estructura de árbol en otra, utilizando un conjunto de operaciones básicas predefinidas, tales como la inserción o eliminación de un nodo. Ya que los documentos XML se pueden ver como estructuras de árboles etiquetados, esta función es apropiada para medir la similitud estructural entre documentos XML.

2.3.5. Coeficiente de Jaccard

El *Coeficiente de Jaccard* es una medida sencilla del grado de similitud entre dos conjuntos. Dados dos conjunto A y B, se define como:

$$d(A, B) = 1 - \frac{(A \cap B)}{(A \cup B)}$$

Esta función expresa la razón entre los elementos que tienen en común los conjuntos, sobre el total de elementos que forman parte de los mismos.

2.4. Métodos de acceso

Las búsquedas por similitud se pueden resolver recorriendo y comparando el objeto consultado con cada uno de los elementos de la base de datos. Sin embargo, en grandes colecciones de objetos este procedimiento no es adecuado debido al alto costo (en tiempo) que tendrá la consulta, por lo cual, es neceario hacer uso de estructuras auxiliares para descartar elementos sin tener que recorrer la base completa. Estas estructuras auxiliares se denominan *Índices* o también *Métodos de Acceso*.

En el caso particular de los espacios vectoriales, en principio existen varias estructuras que facilitan las búsquedas, tales como el *KD-Tree* [Ben75, Ben79], *R-Tree* [Gut88], *Quad-Tree* [Sam84] o el *X-Tree* [BKK96]. Estos métodos utilizan las dimensiones de los vectores como coordenadas para clasificar y agrupar los objetos en el espacio. Sin ambargo, su eficiencia se degrada significativamente cuando crece la cantidad de dimensiones del espacio vectorial: el costo de las búsquedas por similitud depende exponencialmente de la dimensionalidad del espacio métrico [Cha94]. Estrictamente hablando, el factor de mayor influencia en la performance de la consulta no es la cantidad de dimensiones, sino la *Dimensionalidad Intrínseca* del espacio, como se verá más adelante.

Por esta razón, se han diseñado índices sobre espacios métricos generales, que superan la eficiencia de las estructuras anterioremente nombradas cuando la dimensionalidad intrínseca del espacio crece. En [CPRZ97] se muestra que un índice para espacios métricos como el *M-tree* puede ser más eficiente que una de las estructura más utilizadas en espacios vectoriales de baja dimensionalidad, el *R*-tree*. Además, si el espacio métrico no es un espacio vectorial (tal como sucede cuando los objetos son cadenas), dichas estructuras no son aplicables.

2.5. Modelo Unificado

Existen varias estrategias de diseño de índices métricos y cada una de ellas posee sus particularidades, sin embargo, en un mayor nivel de abstracción se puede decir que todos los métodos de acceso constan de dos etapas: preprocesamiento de la base de datos y procesamiento de

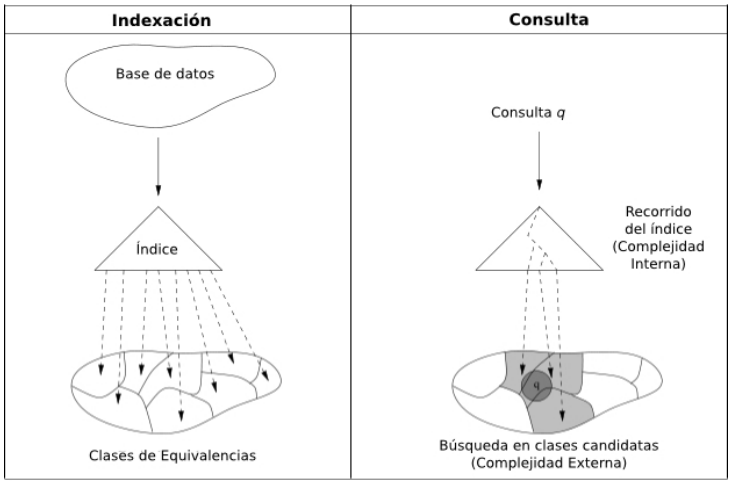

Figura 2.2: *Modelo Unificado de los métodos de acceso*

las consultas. En la primer etapa se realiza la construcción del índice mediante el recorrido de la base de datos y la extracción de información sobre sus elementos y en la segunda parte del proceso se responden las consultas utilizando dicho índice para descartar objetos sin necesidad de compararlos con el objeto que se busca. Por otro lado, se debe notar que todos los índices se construyen particionando el conjunto de objetos de la base de datos B en clases de equivalencia B_i. Durante una búsqueda se utilizan una o más propiedades de los espacios métricos para descartar clases sin necesidad de comparar todos sus elementos con la consulta y así hallar el conjunto de los objetos candidatos, es decir, los objetos que podrían formar parte de la respuesta [CNBYM01]. Al final de esta segunda etapa, se compara la consulta con cada uno de los candidatos y se obtiene el conjunto respuesta. Es decir que la etapa de búsqueda consta a su vez, de dos partes:

1. Recorrer el índice y utilizar propiedades para descartar clases de tal manera de obtener el conjunto C de candidatos. El costo de este proceso se denomina *complejidad interna*.

2. Recorrer el conjunto C de candidatos y comparar sus elementos con la consulta para hallar los elementos que constituirán la respuesta. El costo de este proceso se denomina *complejidad externa*.

Esta abstracción constituye un *modelo unificado* [CNBYM01] del proceso realizado por los métodos de acceso en espacio métricos (Figura 2.2).

Los índices métricos se pueden agrupar en dos clases de acuerdo a la estrategia de diseño que utilizan: *Métodos basados en Particiones Compactas* y *Métodos basados en Pivotes*. A continuación se presentan estas estrategias y se describen algunos índices representativos de cada clase. Posteriormente se describe en detalle el índice FHQT, utilizado en la construcción de los nuevos índices métrico-temporales aquí presentados.

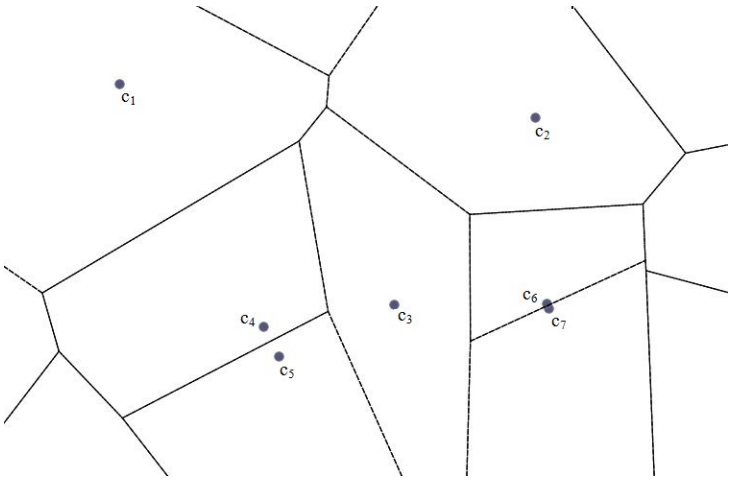

Figura 2.3: *Diagrama de Voronoi (líneas punteadas) para un conjunto de puntos en R^2 con distancia L_2*

2.5.1. Métodos de acceso basados en Particiones Compactas

Consiste en seleccionar un conjunto de puntos llamados *centros* y utilizar los *Diagramas de Voronoi* para dividir el espacio en particiones asociadas a dichos puntos. Los *Diagramas de Voronoi* (Figura 2.3) constituyen una importante estructura en Geometría Computacional, que facilita el tratamiento de problemas relacionados al problema de hallar el *punto más cercano* a uno determinado. Este diagrama se define sobre un conjunto de objetos asociando un área de influencia o alcance de cada uno de ellos. De esta manera, todos los puntos que se encuentran más cerca de un centro que de los demás, estarán ubicados en la partición correspondiente a dicho centro.

A pesar de que esta noción fue desarrollada para espacios vectoriales, donde los vectores representan información sobre coordenadas, su generalización a espacios métricos es posible.

En los métodos de acceso basados en este concepto se definen clases de equivalencia en función de la similitud de los elementos de la base de datos al conjunto de centros. Sea $C = \{c_1, c_2, ..., c_k\}$ el conjunto de centros, la relación de equivalencia (que denotaremos $\sim_C$) se define de la siguiente manera:

$$x \sim_C y \Leftrightarrow NN_C(x) = NN_C(y)$$

$$\text{donde } NN_C(z) = \{c \in C \ / \ \forall c' \in C : d(z, c) \leq d(z, c')\}$$

La partición que esta relación genera en el espacio se conoce con el nombre de *partición compacta* y resulta de dividir el espacio asociando a cada c_i la clase formada por el conjunto de puntos que tiene a c_i como su centro más cercano. Dado un elemento cualquiera x de la base de datos, la clase a la que pertenece x se denota $[x]$. Además, se asume que la partición usa un subconjunto de la base de datos X como el conjunto de centros, aunque estrictamente hablando, estos elementos podrían ser externos.

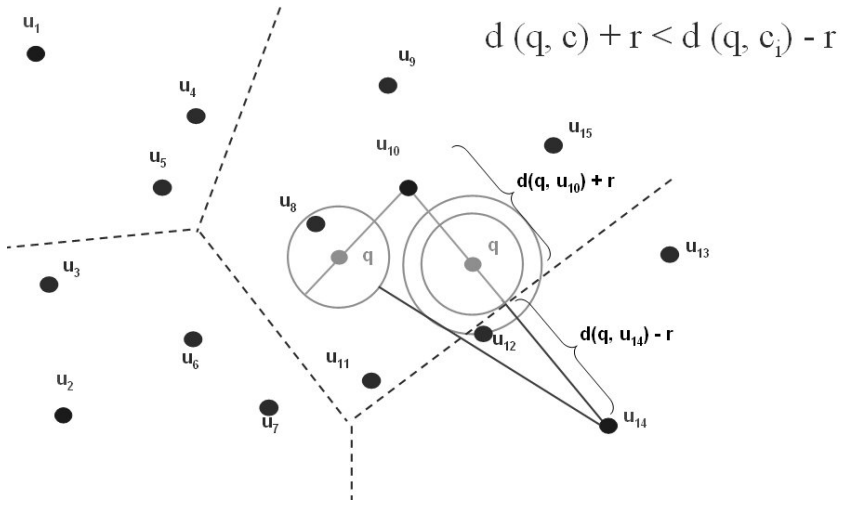

Figura 2.4: *Particiones Compactas - Criterio del hiperplano*

Existen distintos criterios para determinar si una clase completa se puede descartar sin necesidad de comparar sus elementos con la consulta. Los más utilizados son:

- **Criterio del hiperplano:** consiste en determinar si la esfera de la consulta intersecta o no, a la partición que se quiere descartar. Obviamente, la partición en que se encuentra la consulta q, no se puede descartar. Si c_j es el centro de la clase $[q]$, entonces la esfera con centro q no intersecta $[c_i]$ si $d(q, c_j) + r < d(q, c_i) - r$ (Figura 2.4). La expresión $d(q, c_j) + r$ representa el punto más lejano del centro c_j, que pertenece a la esfera de la consulta, mientras que $d(q, c_i) - r$ es la distancia mas cercana de c_i a la esfera. En particiones adyacentes, estos valores son iguales en el límite de las mismas.

- **Criterio del radio de cobertura:** otra manera de determinar si una clase $[c_i]$ se puede descartar es considerar la esfera centrada en c_i que contiene a todos los elementos de X pertenecientes a dicha clase, y determinar si se intersecta con la esfera de la consulta. Definimos el radio de cobertura de un centro c para el conjunto X de la siguiente manera:

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

  La clase $[c_i]$ se puede descartar sin comparar sus elementos si $d(q, c_i) - r > cr(c_i)$ (Figura 2.5). El valor de $d(q, c_i) - r$ es la distancia de c_i a la esfera de la consulta, mientras que el radio de cobertura define el área en la cual están los elementos de la clase.

Algunos de los índices más importantes basados en particiones compactas son:

- **Bisector Tree (BST):** Es un árbol binario donde cada nodo contiene dos centros. Los elementos de la base de datos se asocian al centro más cercano y este proceso se repite en cada subárbol hasta alcanzar las hojas. Para cada centro c_i se determina y almacena su radio de cobertura y ante una consulta (q, r) se descarta aquellos subárboles donde se cumpla que $d(q, c_i) - rc(c_i) > r$ [KM83].

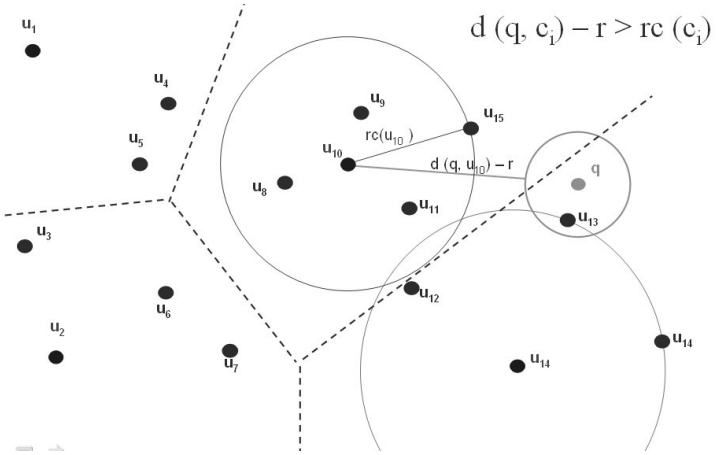

Figura 2.5: *Particiones Compactas - Criterio del radio de cobertura*

- **Generalized-Hyperplane Tree (GHT):** El GHT se construye de la misma manera que el BST, pero no utiliza el radio de cobertura como criterio de descarte, sino el criterio del hiperplano. Sean c_1 y c_2 los centros correspondiente a un nodo y (q, r) una consulta, se puede descartar el subárbol de c_1 si $d(q, c_1) > d(q, c_2) + 2r$ y viceversa [Uhl91].

- **Geometric Near-neighbor Access Tree (GNAT):** El GNAT es un árbol m-ario, es decir que contiene m centros en cada nodo. Junto a los centros se almacena una tabla de tamaño $O(m^2)$ que contiene la distancia de cada centro c_i hacia el elemento más cercano y más lejano de cada clase $[c_j]$. A este intervalo lo llamaremos $p(c_i, c_j)$. Luego, sea (q, r) una consulta por rango, se descarta el subárbol correspondiente al centro c_j si para algún otro centro c_i se cumple que la intersección del intervalo $[d(c_i, q) - r, d(c_i, q) + r]$ con el intervalo $p(c_i, c_j)$ es vacía [Bri95].

- **Spatial Approximation Tree (SAT):** El SAT o sa-tree utiliza un enfoque distinto a los anteriores. Se basa en la *Triangulación de Delaunay* para definir un grafo donde los nodos son elementos de la base de datos y las aristas unen cada objeto con un conjunto de vecinos. Durante la búsqueda del vecino más cercano, se parte de un nodo arbritario y se calcula la distancia de la consulta a dicho objeto y a sus vecinos. Si la consulta se encuentra más cerca del nodo considerado, éste es el resultado buscado. En caso contrario, se realiza un movimiento al vecino de menor distancia a la consulta y se repite el proceso [Nav99].

2.5.2. Métodos de acceso basados en Pivotes

Estos métodos utilizan una estrategia distinta a la de las particiones compactas, basada en la selección de un conjunto de objetos del universo denominados *pivotes* que se utilizan para definir las clases de equivalencia. Así, una clase estará formada por todos los objetos de la base de datos con igual distancia a cada uno de los pivotes. Formalmente, sea $P = p_1, p_2, ..., p_k$ el conjunto de pivotes y x e y dos elementos de la base de datos, la relación de equivalencia se define como:

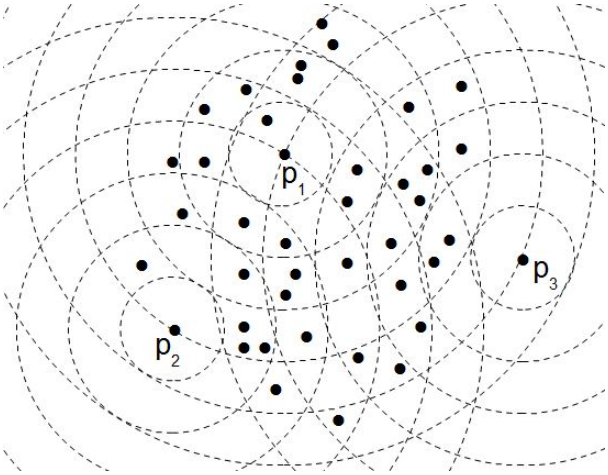

Figura 2.6: *Ejemplo de clases de equivalencia inducida por la intersección de anillos centrados en los pivotes* p_1, p_2 *y* p_3

.

$$x \sim_P y \Leftrightarrow d(x, p_i) = d(y, p_i), \forall i = 1 \dots k$$

Gráficamente, cada clase de equivalencia quedará definida por la intersección de varias capas de esferas o anillos centrados en los pivotes p_i. En la Figura 2.6 se puede ver una relación de equivalencia inducida por los pivotes p_1, p_2 y p_3.

Durante la indexación, estos algoritmos asignan a cada elemento a de la base de datos el vector $\phi(a) = (d(a, p_1), d(a, p_2), ..., d(a, p_k))$, denominado *firma* de a. Posteriormente, durante la búsqueda, se utiliza la desigualdad triangular junto con la firma de cada elemento para eliminar clases de equivalencia sin medir su distancia a q, es decir:

Dada una consulta por rango $(q, r)_d$, en primer lugar se computa la firma de la query q,

$$\phi(q) = (d(q, p_1), d(q, p_2), ..., d(q, p_k))$$

Luego, suponiendo que existe un pivote p_i tal que para algún elemento a de la base de datos se cumple:

$$r < d(q, p_i) - d(a, p_i)$$

Por la desigualdad triangular se sabe que $d(q, p_i) \leq d(q, a) + d(a, p_i)$, por lo tanto reemplazando en la fórmula anterior:

$$r < d(q, p_i) - d(a, p_i) < d(q, a) + d(a, p_i) - d(a, p_i) = d(q, a)$$

En consecuencia, el elemento a puede eliminarse sin necesidad de evaluar su distancia real a la query q. Análogamente se demuestra que si $d(a, p_i) - d(q, p_i) > r$ entonces $d(a, q) > r$.

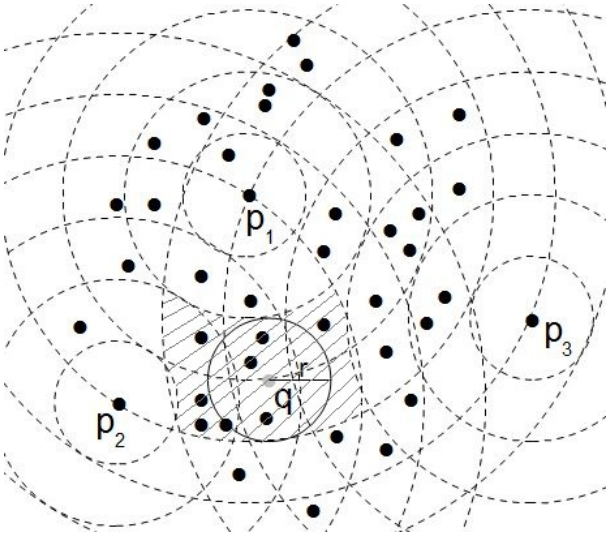

Figura 2.7: *El área resaltada indica las clases que no se pueden descartar*

.

Resumiendo, durante la búsqueda se pueden descartar todos aquellos elementos a tales que para algún pivote p_i se cumple que

$$|d(q, p_i) - d(a, p_i)| > r \tag{2.4}$$

Los elementos no descartados forman parte de una lista de candidatos, que son los únicos que tienen la posibilidad de formar parte de la respuesta 2.7. Estos candidatos posteriormente se comparan directamente con la query q para determinar si su distancia es menor que el radio de tolerancia.

Si la base de datos X posee n elementos, se construye un índice con las nk distancias $d(x, p_i)$, por lo cual al momento de resolver la consulta (q, r) solo es necesario calcular las k distancias $d(q, p_i)$ para obtener la lista de candidatos. Es fácil ver que mientras mayor sea el conjunto de pivotes, mayor será la complejidad interna de la consulta.

Los elementos x no descartados por la condición de la expresión 2.4 deberán ser comparados directamente con q y comprobar si cumplen con la restricción del rango. A este cálculo de distancias adicionales se le denomina complejidad externa de la consulta (q, r).

A continuación se presenta una breve descripción de los algoritmos más conocidos basados en pivotes:

- **Burkhard-Keller Tree (BKT):** Este método asume una función de distancia discreta y se construye recursivamente de la siguiente manera: se elige un elemento arbritario de la base de datos como pivote para la raíz del árbol y se agrupan los demás elementos de tal manera de que todos los que tienen la misma distancia al pivote, se agrupan en una misma rama cuya etiqueta es dicha distancia. Este mismo proceso se realiza para cada grupo hasta que las hojas del árbol contengan solo un elemento. Todos los nodos internos actúan como pivotes. Durante una búsqueda por rango $(q, r)_d$, para el nodo p, si

$d(q,p) <= r$ se devuelve p como parte del resultado. Posteriormente se realiza el mismo proceso con las ramas cuya etiqueta se encuentra en el intervalo $[d(q,p) - r, d(q,p) + r]$ y se descartan las demás [BK73].

- **Fixed Queries Tree (FQT):** es una modificación del *BKT* en la cual para cada nivel del árbol se define un solo pivote. Esto disminuye la cantidad de cálculos de la función de distancia requerida por el método anterior, consecuencia de la comparación de la consulta q con cada nodo interno del árbol. Además se define una cantidad mínima necesaria de elementos para que se proceda a subdividir un nodo. Durante la búsqueda se utiliza un procedimiento similar al anterior hasta alcanzar las hojas del árbol. Todos los elementos de las hojas seleccionadas constituirán el conjunto de *candidatos*, los cuales deberán se comparados con la consulta para obtener el resultado final [BYCMW94].

- **Fixed Height Fixed Queries Tree (FHQT):** el *FHQT* o *FHFQT* es una variante del *FQT* en la cual todas las hojas se encuentran al mismo nivel. Más adelante se explica su funcionamiento con mayor detalle ya que se utilizó como base para los índices métrico-temporales desarrollados en esta tesis [BY97].

- **Fixed Queries Array (FQA):** está estrechamente relacionado al *FHQT*. Es un arreglo de tamaño nh donde h es la altura del *FHQT* asociado (o cantidad de pivotes) y n la cantidad de firmas diferentes. Las firmas se encuentran ordenadas, por lo cual es posible utilizar una búsqueda binaria para determinar cuales firmas definirán el conjunto de candidatos a ser comparados con la consulta [CMN01].

- **Vantage Point Tree (VPT):** los métodos anteriores se utilizan fundamentalmente para distancias discretas. El *VPT* fue diseñado explícitamente para distancias continuas y es un árbol binario en el cual cada nodo interno contiene un pivote y se calcula la mediana de las distancias del pivote al conjunto de elementos considerados para subdividir dicho conjunto en dos grupos: el de los menores a la mediana como hijo izquierdo y el de los mayores o iguales a dicho valor como hijo derecho. Mediante este proceso el árbol queda balanceado. Durante una búsqueda $(q,r)_d$, para cada nodo p, si $d(q,p) <= r$ este elemento se incluye en el resultado y luego se decide cuales subárboles se deben visitar. Sea m la mediana correspondiente a los elementos descendientes del nodo p, si $d(q,p) + r < m$ entonces se descarta el hijo derecho, mientras que si $d(q,p) - r > m$ se descarta el hijo izquierdo. Nótese que es posible que sea necesario entrar en ambos subárboles [Yia93]. Existen variantes de este árbol, tales como el *Multi-way Vantage Point Tree (mw-VPT)* [BO97] que sigue la misma estrategia pero utiliza percentiles para generar un árbol r-ario en lugar de binario y el *Excluded Middle Vantage Point Forest (VPF)* que excluye los elementos centrales (cercanos a la mediana) para generar otro árbol VPT, dando como resultado un bosque. De esta manera se reduce la probabilidad de que ante una consulta se requiera visitar tanto el subárbol izquierdo como el derecho [Yia99].

- **Approximating Eliminating Search Algorithm (AESA):** esta estructura es una matríz donde se registran todas las distancias entre los elementos de la base de datos. En este caso cada elemento juega el rol de pivote respecto a los demás. Durante una búsqueda se elige un elemento de partida al azar y se utiliza la desigualdad triangular para descartar elementos sin compararlos con la consulta. Este proceso se realiza recursivamente pero

tomando solo los elementos no descartados. El método es muy eficiente en tiempo, pero su costo espacial $O(n^2)$ es demasiado alto como para ser útil en grandes bases de datos [Vid86]. En [MOV94] se presenta una variante denominada **Linear AESA (LAESA)** que disminuye el costo espacial a través de la reducción de la cantidad de pivotes.

2.5.2.1. Fixed Height Queries Tree (FHQT)

Es esta sección se describe con mayor profundidad el $FHQT$ [BY97], método que fue utilizado en esta tesis como base para el diseño de índices métrico-temporales. Una característica importante de este índice es que es dinámico, es decir que si la cantidad de elementos de la base de datos crece o disminuye significativamente, se pueden agregar o eliminar pivotes (niveles) sin necesidad de modificar el resto de la estructura.

Este índice es una variante del Fixed Queries Tree (FQT) [BYCMW94] en donde todas las hojas se encuentran a la misma altura. En otras palabras, los caminos más cortos se extienden hasta el último nivel de profundidad del árbol. Esta extensión tiene como consecuencia que la capacidad de descarte durante la búsqueda sea mayor, ya que se utilizarán la totalidad de los pivotes para tratar de eliminar cualquier elemento de la base de datos.

El $FHQT$ se construye a partir de k pivotes que pueden ser elegido al azar o mediante algún procedimiento de selección de pivotes [BNC01] de la base de datos X. También es posible que los pivotes se obtengan generándolos especialmente para un conjunto de datos, aunque no se tiene conocimiento de estudios que desarrollen esta posibilidad. Sea p un pivote, para cada distancia i de p hacia algún elemento de la base de datos, 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 se crea un hijo del nodo correspondiente a p, con rótulo i, y se construye recursivamente el árbol teniendo en cuenta que todos los subárboles del mismo nivel usarán el mismo pivote como raíz. Este proceso se continúa hasta lograr que todas las hojas tengan menos de b (tamaño del $bucket$) elementos y estén en un mismo nivel. La Figura 2.8 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. Cuando se alcanzan las hojas se obtiene al conjunto de los elementos $candidatos$, que deberán ser comparados con la consulta para determinar si forman parte del resultado.

2.5.2.2. Ejemplo de Búsqueda

Por ejemplo, para la consulta $(q, 1)_d$, siendo $(4, 6)$ la firma de q, el FHQT de la Figura 2.8 se comporta de la siguiente manera: como el radio de búsqueda es 1 y la distancia de q al primer pivote es 4, aplicando la condición de exclusión 2.4, se seleccionan solo las ramas etiquetadas con las distancias del intervalo $[4 - 1, 4 + 1]$, es decir 3, 4 y 5 y se descartan las ramas 0, 2 y 6. En el siguiente nivel, se realiza el mismo proceso considerando en este caso solo las ramas $[6 - 1, 6 + 1]$ de los subárboles seleccionados en el nivel 1. Por lo tanto el conjunto resultante

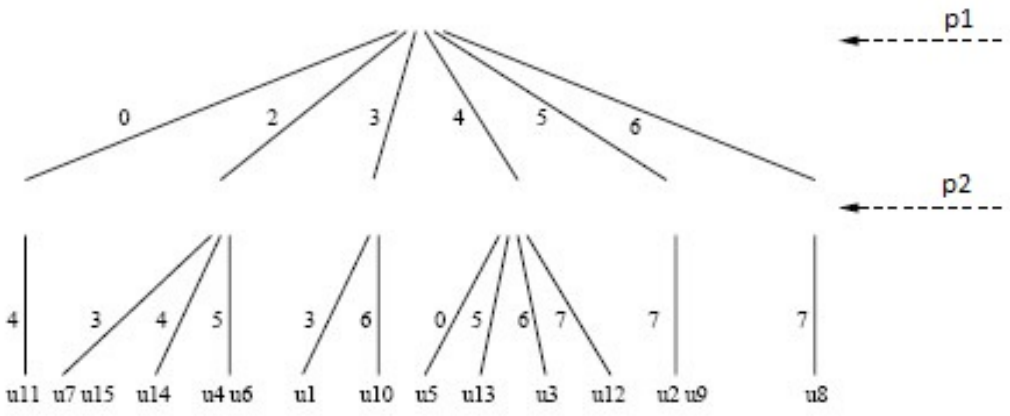

Figura 2.8: *FHQT con pivotes p_1 y p_2*

está formado por los siguientes objetos: $u_{10}, u_{13}, u_3, u_{12}, u_2, u_9$. En la Figura 2.9 se muestra el pseudocódigo del algoritmo de consulta del $FHQT$.

FHQT $(q, r)_d$
1. calcular f de q – f es la *firma de* q
2. resultado := Consultar(q, r, 1, f, $raiz$)
3. **return** resultado

donde **Consultar** se define recursivamente como:

Consultar(q, r, n, f, x) – n es el nivel del nodo actual x
1. resultado:=$\emptyset$
2. **if** esHoja(x) **then**
3. **for all** objeto ($o \in x$)
4. **if** $(d(q, o) <= r)$ **then**
5. resultado :=resultado $\cup$ $\{o\}$
6. **else**
7. **for all** hijo ($h_i \in x$)
8. **if** $|f_n - d_i| \leq r$ **then**
9. resultado :=resultado $\cup$ **calcularCandidatos**$(q, r, n + 1, f, h_i)$
10. **return** resultado

Figura 2.9: *Pseudocódigo de consulta del FHQT*

La complejidad espacial del $FHQT$ está entre $O(n)$ y $O(nh)$, donde h el la altura del árbol y n la cantidad de elementos de la base de datos. Se considera que la cantidad óptima de pivotes es $\log(n)$, aunque depende de la distribución del histograma de distancias. Originalmente estas estructuras fueron propuestas para funciones de distancias discretas, pero se pueden adaptar a distancias continuas discretizando los valores de las mismas [RCH04, CHRV05]. También es posible transformarlas para que manejen distancias continuas asignando rangos de valores a cada rama, como veremos más adelante en las propuestas de los nuevos índices métrico-temporales.

2.6. Maldición de la Dimensionalidad

Uno de los grandes obstáculos en el diseño de algoritmos de búsqueda por similitud es la llamada *maldición de la dimensionalidad* [CMN99, CNBYM01]. Si se representa el conjunto de elementos de un espacio métrico como puntos en un espacio vectorial, su dimensión corresponde al número de coordenadas que tienen los puntos que lo componen. Todos los algoritmos de búsqueda por similitud disminuyen su rendimiento cuando aumenta la dimensión del conjunto, hecho conocido como *maldición de la dimensionalidad*. En [CNBYM01] se muestra que el concepto de dimensión de un espacio vectorial se puede extender a espacios métricos arbitrarios utilizando el concepto de *dimensión intrínseca*, y que la maldición de la dimensionalidad tiene consecuencias similares en dichos espacios.

Se define la *dimensión intrínseca* de un espacio métrico como:

$$\rho = \frac{\mu^2}{2\sigma^2} \tag{2.5}$$

Los parámetros μ y σ de la Ec. 2.5 son la media y la varianza respectivamente, del histograma de distancias de los puntos que conforman dicho espacio métrico.

A medida que aumenta la dimensión intrínseca del espacio métrico los algoritmos basados en pivotes necesitan una mayor cantidad de pivotes para mantener, en cierta medida, su rendimiento y el rendimiento empeora cuando crece la dimensión del espacio.

Cuando se realiza una consulta por rango, se quieren recuperar los elementos del espacio métrico que se encuentran a una distancia menor que un radio dado de la query q. Los algoritmos basados en pivotes buscan frente a esta consulta descartar la mayor cantidad de elementos del espacio métrico antes de realizar una búsqueda exhaustiva, es decir, generan una lista de elementos candidatos en la que se verifica exhaustivamente la condición de búsqueda.

Considerando el histograma de distancias de un espacio métrico (U, d) es fácil ver que según la condición de exclusión de elementos (Ec. 2.4), dada una consulta (q, r) y un conjunto de k pivotes p_i, se pueden descartar todos los elementos $x \in E$ que no cumplan para algún $i \in 1...k$ la condición de la Ec. 2.6:

$$d(p_i, x) \notin [d(p_i, q) - r, d(p_i, q) + r] \tag{2.6}$$

Cuanto mayor sea la dimensión intrínseca del espacio métrico, la media del histograma aumenta y su varianza disminuye.

Una disminución de la varianza del histograma tiene como consecuencia que la cantidad de elementos que se encuentran dentro del rango $[d(p_i, q) - r, d(p_i, q) + r]$ sea mayor, es decir, cada vez son menos los elementos que se pueden descartar. Por otro lado, si la media crece entonces se necesita un r mayor para que la búsqueda no devuelva un resultado vacío, por lo que nuevamente, la capacidad de descarte se ve disminuida. En espacios de alta dimensionalidad,

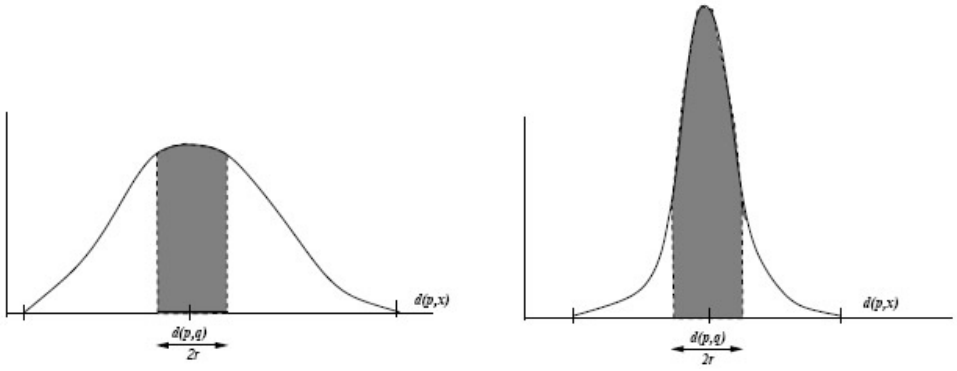

Figura 2.10: *Histogramas de distancias de baja dimensionalidad (izquierda) y de alta dimensionalidad (derecha)*

prácticamente todos los elementos se transforman en candidatos a ser verificados exhaustivamente (complejidad externa). A esto se le denomina *maldición de la dimensionalidad*, y es independiente de la naturaleza del espacio métrico al que pertenecen los elementos.

La Figura 2.10 muestra intuitivamente la maldición de la dimensionalidad. El histograma de distancias de la izquierda representa un espacio métrico de dimensión baja, y el de la derecha de dimensión alta. Se puede apreciar que en el caso del espacio con dimensión alta casi ningún elemento puede excluirse de la lista de candidatos, por lo que debe realizarse prácticamente una búsqueda exhaustiva para responder la consulta. Para poder descartar más elementos es necesario utilizar más pivotes, pero esto aumenta la complejidad interna de la consulta.

Por lo tanto, los algoritmos basados en pivotes necesitan cantidades enormes de memoria si el espacio a indexar es de alta dimensionalidad, lo que hace que muchas veces la memoria sea insuficiente para manejar el problema. En la práctica, para espacios con dimensión intrínseca mayor o igual a 20 se considera el problema como intratable. Por ese motivo, varios autores proponen el uso de técnicas de indexación basadas en distancias, que utilizan solo la distancia entre puntos y evitan las referencias a coordenadas, intentando de ese modo evitar la maldición de la dimensionalidad.

En [CNBYM01] se muestra que el número óptimo de pivotes es $O(\log(n))$ si se eligen al azar, donde n es el número de elementos de la base de datos. Por otro lado, en [FLL93] se demuestra formalmente que si la dimensión es constante y a través de una selección adecuada de una cantidad fija de pivotes, el costo de la búsqueda por similitud es $O(1)$, lo que muestra que la forma en que los pivotes son escogidos afecta el rendimiento del algoritmo. Esto implica que una mejora en la forma de elegir los pivotes permite usar menos memoria y produce mejoras significativas en el redimiento. Por razones de espacio la descripción detallada de los métodos de selección de pivotes queda fuera el alcance de este informe.

Capítulo 3

El Modelo Temporal

En este capítulo se presenta una introducción al *Modelo Temporal*, que constituye el segundo aspecto en el cual se basa el modelo métrico-temporal. Como en el captítulo anterior, se muestran además los métodos de acceso temporales relevantes para este trabajo y se incluye el índice $SEST_L$, que está orientado a consultas espacio-temporales basadas en eventos, a partir del cuál se tomaron ideas para el diseño del $Event - FHQT$.

3.1. Conceptos Básicos

Una *Base de Datos Temporal* es aquella que soporta algun aspecto de tiempo distinto del tiempo definido por el usuario. En las últimas dos décadas el interés por las bases de datos temporales se incrementó significativamente con la contribucion de una gran cantidad de investigadores. Sin embargo, la falta de métodos de acceso eficientes es tal vez una de las razones por las cuales los diseñadores de DBMS (Sistema de Gestión de Bases de Datos) comerciales han sido reticentes en adoptar los avances alcanzados en esta línea de investigación.

Las bases de datos convencionales usualmente capturan el estado actual de la realidad modelada. Cuando una base de datos cambia de un estado a otro debido a alguna actualizacion, los datos viejos se sobreescriben con los nuevos. Sin embargo, existen muchas aplicaciones donde es necesario mantener la historia o evolución de los datos hasta el momento actual, e inclusive en algunos casos es necesario calcular y almacenar los valores mas probables a tomar en el futuro.

Las bases de datos temporales soportan algún tipo de dominio de tiempo manejado internamente por el DBMS. El manejo de datos que varían con el tiempo no está completamente implementado en la mayoría de las aplicaciones ya que muy pocos sistemas administradores de bases de datos comerciales soportan datos temporales. En la actualidad, cuando una aplicación requiere mantener datos temporales, en la mayoría de los casos esta funcionalidad se debe programar específicamente, tarea que resulta bastante ardua [Sno00]. Actualmente algunas bases de datos comerciales incluyen extensiones temporales, como Oracle Timeseries, Oracle Flash-

Back y el Informix TimeSeries Data-Blade, pero esas extensiones no dan un soporte completo para el manejo eficiente de datos que varían con el tiempo y en algunos casos solo se limitan a series de tiempos.

A continuación se definen algunos conceptos importantes del modelo temporal:

Tiempo De acuerdo a los diccionarios de la Universidad de Oxford, el tiempo es el *"progreso continuio e indefinido de la existencia y de los eventos en el pasado, presente y futuro, considerado como un todo"*. El diccionario de la Real Academia Española lo define como: *"magnitud física que permite ordenar la secuencia de los sucesos, estableciendo un pasado, un presente y un futuro"*, mientras que de acuerdo al "Webster's New World College Dictionary", el tiempo es *"la duración en la cual todas las cosas ocurren o un preciso instante en el cual algo ocurre"*. El tiempo se considera totalmente ordenado y en constante aumento hacia una dirección, el futuro. Esta interpretación del tiempo se toma desde una percepción humana, asumiendo la capacidad de memorizar el pasado y de realizar predicciones sobre el futuro. Solo se puede saber si esas predicciones fueron exactas cuando el tiempo que era futuro se vuelve presente o pasado. Resulta necesario comprender como medir y modelar el tiempo en modelos de datos temporales. En la vida diaria para cuantificar el tiempo se toman diferentes granularidades como años, meses, días, horas, segundos y se utilizan distintos tipos de calendarios para estructurar estas variadas granularidades. Es importante diferenciar entre los conceptos de *tiempo relativo* y *tiempo absoluto*. Tiempo absoluto se refiere a un instante particular, por ejemplo el *29 de Octubre de 2012, 14:05 horas*, mientras que el tiempo relativo se refiere a una duración, tal como *22 horas*.

Tiempo actual Una problemática en relación a las bases de datos es cómo almacenar la información relativa al tiempo actual (*now*). Los eventos ocurren en el tiempo actual y éste separa el pasado del futuro. El problema es identificar el tiempo que debería asignarse a un hecho que se registra como actual, ya que el tiempo actual será diferente en el instante siguiente, es decir, que la interpretación temporal del objeto cambia. Esto también sucede en las consultas, por ejemplo, si se pregunta: *¿Cuál es el sueldo de Pedro?*, implícitamente la consulta se refiere al momento actual, y puede producir distintas respuestas dependiendo del momento en que se ejecute. Pero si se pregunta: *¿Cuál era el sueldo de Pedro en Octubre de 2012?* la consulta devolverá siempre la misma respuesta. Las semánticas entre los modelos que soportan solo instantes fijos y aquellos que soportan instantes variables son evidentemente distintas [Sta05]. Existen varios enfoques para representar *now*. Uno de ellos tiene en cuenta el constante avance del tiempo actual, entonces se utiliza un valor de *now* que también varíe para reflejar el nuevo tiempo actual. Este enfoque propone el uso de la variable *UC (until changed)* para representar *now* [WJL93]. Un enfoque más actual para representar el tiempo actual en bases de datos bitemporales es el denominado *POINT*, que ha superado algunos inconvenientes de los enfoques anteriores y mejorado la eficiencia en las consultas [STS03, SST09].

Granularidad del tiempo Según el propósito de la base de datos pueden usarse diferentes granularidades de tiempo. Por ejemplo, en algunas aplicaciones puede ser necesario almacenar

el tiempo con una granularidad de segundos o milisegundos, mientras que en otras es suficiente registrarlo con una granularidad de días. El término *quanta* o *chronon* es utilizado para hacer referencia a la mínima unidad de tiempo indivisible de duración fija [DGK$^+$94]. Es importante señalar que un *quanta* no es un punto o valor discreto, sino un segmento en la escala de tiempo. El tamaño de cada *quanta* generalmente es fijo y depende de la granularidad que se quiera representar (segundos, días, años). Debido a la diversidad de términos utilizados en bases de datos temporales es importante explicar dos términos más: *instante* y *evento*. Un *instante* es un punto en una línea de tiempo, mientras que un *evento* es un hecho que ocurre en un instante específico. Debido al uso de un modelo discreto, solo es posible registrar que ese evento ocurrió en un determinado *quanta*. Por ejemplo, si el tiempo se almacena con granularidad de días, el nacimiento de una persona (que ocurre en un instante determinado) solo tendrá asociado el día y no su hora, minutos, etc. En otras palabras, un *quanta* es una granularidad usada en el modelo temporal.

Intervalo de Tiempo Es el tiempo transcurrido entre dos instantes. Si el sistema se basa en *quanta*, un intervalo puede considerarse como una secuencia contigua de estos elementos.

Timestamp Es un valor de tiempo asociado a un objeto, por ejemplo, un valor de un atributo con información temporal del objeto.

3.2. Modelo de datos temporal

Se puede considerar que todo modelo de datos está constituido por estructuras de datos, operaciones de recuperación y actualización y restricciones de integridad, y se representa como sigue:

$$M = (ED, Op, R)$$

donde M es el modelo de datos, ED son las estructuras de datos, Op las operaciones y R las restricciones. Al incorporar el tiempo al modelo de datos, éste debe agregarse a cada uno de sus componentes por separado. Por lo tanto, el modelo de datos temporal puede representarse como:

$$M_t = (ED_t, Op_t, R_t)$$

donde M_t es el modelo de datos temporal, ED_t son las estructuras de datos temporales, Op_t las operaciones temporales y R_t las restricciones de integridad temporales. Las estructuras de datos deben adaptarse para que puedan almacenar datos que varían con el tiempo. El álgebra y las operaciones de modificación deben redefinirse usando semánticas temporales. Y para cada restricción expresable en el modelo de datos no temporal M, hay una homóloga en el modelo de datos temporal M_t.

Existen tres modelos básicos propuestos para representar el tiempo: *continuo*, *denso* o *discreto*. En todos los modelos el tiempo se hace corresponder a un conjunto de números totalmente ordenado respecto al predicado de comparación $<$. En el modelo continuo cada número

real corresponde a un punto en la línea de tiempo, pero se considera que entre dos puntos consecutivos no hay separación. En el modelo denso, como en el continuo, los puntos en la línea de tiempo corresponden a números reales, pero entre dos puntos pueden existir otros puntos. En el modelo discreto se asocia a cada número natural una unidad de tiempo indivisible con una duración predefinida, el *quanta* [CU85]. Como las computadoras digitales solo soportan una granularidad limitada para números reales, la mayoría de las propuestas para agregar tiempo al modelo relacional se basan en el **modelo de tiempo discreto totalmente ordenado** [Sta05].

3.3. Dimensionalidad del tiempo

En la literatura de bases de datos temporales [DD02, Jen00, Sno00] se mencionan varias líneas de tiempo de interés, necesarias para capturar distintas nociones y la relación entre tiempo y datos. Esas líneas de tiempo se utilizan para distinguir distintas clases de bases de datos temporales, según su capacidad para modelar la realidad y capturar diferentes tipos de datos temporales.

Tiempo definido por el usuario El modelo de datos relacional soporta este tipo de tiempo al asignar un tipo *date* a los atributos. Los valores que puede tomar el tipo *date* pueden ser cualquier instante de tiempo referido al pasado, presente o puntos en el tiempo futuro, pero dentro del intervalo soportado por el DBMS particular. Los valores para estos tiempos son suministrados por el usuario y, como cualquier otro tipo de datos, pueden insertarse, actualizarse o eliminarse. El DBMS trata a este dato temporal como a cualquier otro tipo de atributo.

Tiempo Válido Se trata de la líneas de tiempo en el modelo de datos temporal que se utiliza para capturar el tiempo en que un hecho es verdadero en la realidad modelada. Por ejemplo, se puede querer saber desde cuando una propiedad perteneció a un dueño, y hasta cuando. También puede ser necesario registrar eventos futuros probables, tales como la reserva de una habitación de un hotel para un día determinado. Al registro de la información respecto a en qué momento fue, es o será válido un hecho en el mundo real, se le denomina *tiempo válido* [DGK$^+$94]. Todos los hechos tienen un tiempo válido por definición. Sin embargo, el tiempo válido de un hecho puede no ser registrado en la base de datos (porque puede ser que no se conozca, o que no resulte relevante para la aplicación en cuestión). Si una base de datos modela diferentes realidades, los hechos en la base pueden tener varios tiempos válidos, uno por cada realidad modelada. Un hecho puede tener asociado cualquier cantidad de instantes o intervalos de tiempo, aunque en la práctica la mayoría de los hechos tienen solo uno. El tiempo válido generalmente es suministrado por el usuario.

Tiempo Transaccional En general no resulta posible almacenar información sobre hechos en la base de datos en el tiempo exacto en que éstos ocurren en la realidad. Por ejemplo, es muy difícil que se actualice la base de datos en el momento exacto en que una persona accede a un nuevo puesto laboral, debido a demoras en el proceso de información. Por lo tanto, existe

cierta diferencia entre el tiempo en que un dato se vuelve verdadero en la realidad modelada y el tiempo en que efectivamente se registra. Esta última noción de tiempo se denomina *tiempo transaccional* y representa el tiempo en que un hecho está vigente en una base de datos y, por lo tanto, puede ser recuperado. La dimensión de tiempo transaccional representa la historia de la actividad de la base de datos y no la historia del mundo real. Es decir, esta dimensionalidad se corresponde con el orden en que se ejecutan las transacciones, por lo tanto no puede extenderse en el futuro, es decir, el tiempo transaccional no puede ser mayor al tiempo actual. A diferencia del tiempo válido, el tiempo transaccional puede asociarse con cualquier entidad de una base de datos, no sólo con hechos. Así, todas las entidades de una base de datos tienen un aspecto de tiempo transaccional que tiene una duración: desde la inserción hasta la eliminación, siendo posibles muchas inserciones y/o eliminaciones de la misma entidad. La eliminación de una entidad no remueve físicamente la entidad de la base, sino que simplemente deja de ser parte del estado actual de la misma. El aspecto transaccional puede ser o no capturado en la base. Las aplicaciones que requieren trazabilidad se basan en bases de datos que registran esta noción de tiempo.

Las nuevas estructuras que se presentan en esta tesis consideran solo la línea de tiempo válido como dimensionalidad de tiempo, pero podrían extenderse a las demás.

3.4. Clases de Bases de Datos Temporales

Según la capacidad de una base de datos temporal para modelar la realidad y gestionar datos temporales, se puede hablar de tres tipos de bases de datos: *históricas*, *rollback* y *bitemporales*.

3.4.1. Bases de Datos Históricas

Los hechos que ocurren en la realidad modelada se capturan a lo largo de la línea de tiempo válido. Para capturar esos hechos y almacenarlos se usan las *Bases de Datos Históricas* o de *Tiempo Válido*. Pueden registrarse estados de la base pasados, presentes y futuros. Estas nociones son relativas al tiempo actual *now*. Un hecho es futuro si su intervalo de tiempo asociado es posterior a *now*; presente si su intervalo de tiempo contiene a *now*; y pasado si su intervalo de tiempo es previo a *now*.

Al registrar un cambio en un hecho de la realidad se genera un nuevo estado y un nuevo registro. El tiempo válido del hecho que se quiere registrar debe ser suministrado por el usuario. Durante la actualización de tiempo válido, el estado previo de un hecho no puede preservarse, es decir que en este tipo de bases de datos los datos se eliminan físicamente cuando se modifican.

Es importante mencionar que las bases de datos de tiempo válido soportan todas las operaciones sobre intervalos de datos, eliminación, inserción y actualización en cualquier tiempo. Estas bases de datos requieren un lenguaje de consulta temporal que soporte esta lógica; por ejemplo, las sentencias de actualización necesitan especificar qué estados de la base a lo largo del tiempo válido serán actualizados por dicha operación.

3.4.2. Bases de Datos Rollback

Estas bases de datos, también llamadas de *Tiempo Transaccional*, registran los cambios que ocurren en el tiempo de la transacción. Obviamente, no pueden registrar transacciones futuras. Siempre que se ejecuta una sentencia de modificación, el sistema registra un nuevo estado de la base de datos según el tiempo en que la modificación se realizó en el sistema y mantiene el viejo estado. La gestión del tiempo transaccional queda a cargo del sistema, no del usuario. Los datos no se eliminan físicamente, por ese motivo en estas bases de datos solo se realizan incorporaciones de datos. Para distinguir las diversas versiones de un objeto puede utilizarse el tiempo de modificación del mismo. Como la base de datos de tiempo transaccional registra la historia de su actividad, más que la historia de la realidad modelada, se puede *regresar* a uno de sus estados previos, pero no pueden realizarse modificaciones sobre las tuplas existentes. Para introducir cambios en la base se debe crear una nueva tupla con un nuevo tiempo transaccional. Este tipo de bases de datos es fundamental, por ejemplo, para aplicaciones de auditoría.

3.4.3. Bases de Datos Bitemporales

Las bases de datos de tiempo válido capturan estados de los hechos reales durante la línea de tiempo válida, almacenan hechos pasados, actuales o incluso, futuros. Si se descubren errores cuando se trabaja con este enfoque temporal, se corrigen actualizando la base de datos. En este caso, se descartan los valores previos. Por este motivo, a partir de la actualización no será posible ver el estado previo de la base. Por otra parte, las bases de datos de tiempo transaccional registran un nuevo estado de la base según el tiempo en que fue hecha la modificación en el sistema y además mantienen los estados anteriores. Solo es posible conservar el estado previo de un hecho si el tiempo en que el hecho se almacena en la base de datos (el tiempo transaccional), también se registra, lo que significa que es necesario almacenar diferentes tiempos válidos de un hecho.

Una base de datos bitemporal es una combinación de bases de datos de tiempo válido y transaccional. Cuando se realiza la inserción de un objeto, su atributo temporal correspondiente al tiempo de transacción es de la forma [t_i, *now*), lo que indica que la tupla es actual y se desconoce el tiempo final. Solo se permiten actualizaciones sobre las versiones más recientes de los objetos y no se permite modificar el pasado. La eliminación es lógica, no existen eliminaciones físicas en las bases de datos bitemporales. Cuando se elimina un objeto, su atributo de tiempo transaccional se cambia de [t_i, *now*) a [t_i, t_f) donde t_f es el tiempo en el que se ejecuta la sentencia de eliminación.

En las bases de datos bitemporales, el tiempo válido y el transaccional son ortogonales [SA89]. Esto significa que todas las operaciones de recuperación y las restricciones referidas al tiempo válido pueden también ser utilizadas para el tiempo transaccional. Esto no aplica para las operaciones de modificación, ya que las correspondientes al tiempo transaccional son manejadas por el DBMS, y las de tiempo válido por el usuario. El lenguaje de actualización del tiempo transaccional debe ser diferente al del tiempo válido, ya que el usuario puede actualizar el tiempo válido de un hecho pero no le está permitido modificar su tiempo de transacción.

En este tipo de base de datos es posible realizar consultas sobre la historia de las modificaciones, así como también sobre la historia de los hechos de la realidad modelada. Un DBMS bitemporal es capaz de interpretar tanto tiempo transaccional como tiempo válido durante la evaluación de una consulta o el chequeo de restricciones. Los valores del atributo de tiempo transaccional pueden ser como máximo iguales al tiempo actual, ya que esta dimensión representa el tiempo en que se ejecuta una operación en la base de datos, y el propio DBMS lo registra. No es posible cambiar el tiempo de transacción y no es posible deshacer una transacción confirmada. La única manera de modificar una transacción confirmada es ejecutar una transacción inversa, que será ejecutada y confirmada en un punto de tiempo posterior y, por lo tanto, dará lugar a una nueva tupla.

3.5. Registro tiempos asociados a datos (timestamp)

Los hechos en los modelos de datos temporales se representan como unidades de datos con tiempos asociados. Esos tiempos representan el período de tiempo en que los datos son válidos en la realidad modelada y/o en que son almacenados en la base de datos. Estas unidades de datos pueden representarse de diferentes maneras:

- Como un valor único
- Como una combinación de valores de propiedades que pertenecen a la misma entidad
- Como un conjunto de varias entidades agrupadas

Al registrar datos con tiempos se debe decidir si el tiempo estará asociado a un atributo particular, a una tupla completa, a una colección de tuplas, o incluso a una base de datos completa. Las publicaciones del área de bases de datos temporales generalmente discuten dos enfoques básicos de registro de datos con tiempos: registro de tupla y registro de atributo.

Tiempo asociado a tuplas Los modelos de datos que usan este enfoque agregan marcas de tiempo a cada tupla en una relación. En las bases de datos históricas, cada tupla se registra con un intervalo de validez del hecho, en las bases de datos rollback con intervalos de tiempo transaccional y en una base de datos bitemporal, con intervalos de tiempo válido y transaccional. La principal desventaja del registro de tuplas con tiempos es el hecho de que la información sobre una entidad del mundo real se propaga a varias tuplas, es decir que causan redundancia de datos.

Tiempo asociado a atributos Al contrario que las tuplas con tiempos, los atributos con marcas de tiempo superan la desventaja de la redundancia de datos. En este enfoque, el instante se agrega a cada valor de un atributo, así los valores de una tupla que no se ven afectados por una modificación, no se repiten. Básicamente la historia de los valores se almacena de forma separada para cada atributo.

3.6. Asociación de tiempos a datos

La elección de a qué datos se le asociarán tiempos y de cómo hacerlo está determinada por el modelo de datos subyacente. Usualmente se asocia un instante de tiempo, un intervalo de tiempo o un elemento temporal a los datos que requerien información temporal. Los modelos de datos relacionales temporales que se encuentran en la primera forma normal, solo permiten asociar tiempos mediante la adición de atributos a una tupla. Por otro lado, los modelos que no están en la primera forma normal o que contienen objetos, permiten asociar valores de atributos termporales más complejos. En todos los casos los intervalos de tiempo se consideran cerrados en su límite inferior y abiertos en su límite superior.

Modelos de Datos basados en Puntos e Intervalos Las bases de datos de tiempo válido almacenan hechos y el momento o período en que estos hechos son verdaderos en el mundo real. Desde un punto de vista teórico, todos los instantes de tiempo en los cuales el hecho es verdadero para la realidad modelada son almacenados en la base de datos. Cuando ese conjunto de instantes de tiempo contiene solo instantes continuos, obviamente resulta necesario representar dicho hecho como un intervalo de tiempo. Por lo tanto pueden utilizarse intervalos de tiempo para modelar el período durante el cual un hecho fue verdadero en el mundo real. Un intervalo de tiempo es un período de tiempo que posee un instante de tiempo inicial t_i y un instante de tiempo final t_f. Dichos valores constituyen el límite inferior y superior en la escala de tiempo, del conjunto de instantes correspondientes al intervalo.

Registro de Instantes de Tiempo Para poder registrar los datos mediante un instante de tiempo, se asume que dichos datos son válidos sólo en ese instante de tiempo específico. En la literatura las relaciones que contienen datos representados con instantes de tiempo se denominan tablas de eventos. Para asociar instantes a las tuplas registradas basta con extender el esquema con un atributo de tipo fecha (*date*). Este atributo registra el instante de tiempo en que el evento correspondiente ocurrió.

Registro de Intervalos de Tiempo En este caso se registran dos instantes asociados a cada tupla, que representan el intervalo en que un hecho es válido en la realidad modelada. Para ello se agregan dos atributos del tipo *date*. Las tablas bitemporales se extienden con cuatro atributos, dos para capturar el tiempo válido y dos para el tiempo transaccional. Como se mencionó anteriormente, las tuplas registradas con intervalos de tiempo asociados introducen redundancia. Esto se debe a que la actualización de valores en una tupla genera una nueva tupla en la relación, por lo que todos los valores de los atributos se repiten, incluyendo aquellos atributos que no se ven afectados por la actualización. Además es posible que una tupla que es válida durante períodos de tiempo no superpuestos se almacene de forma separada para cada período de tiempo, lo que hace que la historia del objeto se represente mediante varias tuplas. Lo mismo puede ocurrir al calcular el resultado de una consulta, ya que los intervalos de tiempo no son cerrados para las operaciones de intersección, diferencia y unión. El conjunto de las fechas asociadas a un atributo contiene la historia de dicho atributo en una tupla. Esto genera

redundancia si un mismo valor aparece para el mismo atributo varias veces durante períodos de tiempo no superpuestos.

Registro de Elementos Temporales Para evitar la redundancia en el caso de que aparezcan los mismos valores varias veces en períodos de tiempo no superpuestos, puede utilizarse el registro de tiempo con elementos temporales. Este método permite modelar un hecho que fue válido durante varios períodos de tiempo no superpuestos. Además, existe la ventaja de que los elementos temporales son cerrados para las operaciones de intersección, diferencia y unión. La extención de las estructuras de datos de un modelo de datos no temporal para soportar el registro de datos con elementos temporales solo puede hacerse si dicho modelo soporta valores de atributos no atómicos. El uso de atributos con elementos temporales no introduce redundancia pero conduce a relaciones muy complejas y dificulta la escritura de código que asegure las restricciones de integridad y el acceso eficiente a los datos.

3.7. Tipos de consultas

Desde el punto de vista de las consultas, tanto una base de datos de tiempo válido como una de tiempo transaccional, pueden ser vistas como colecciones de intervalos y los tipos de consultas asociados tienen características similares. Por esto, usualmente se analizan las consultas en un solo tipo de tiempo (*válido* o *transaccional*) y se extienden al restante [ST99].

Los tipos de consultas no triviales más importantes son:

1. Dado un intervalo contiguo T, hallar todos los objetos *vigentes* durante ese intervalo. Por ejemplo, *devolver todos los productos de la compañía cuyo lanzamiento se produjo entre el 5 de noviembre de 2010 y el 30 de junio de 2012*.

2. Dados un rango de claves y un intervalo de tiempo contiguo T, hallar todos los objetos cuyas claves forman parte del rango con vigencia dentro del intervalo T. Por ejemplo, *hallar los docentes de la universidad que trabajaron entre el 31 de marzo de 2005 y el 24 de diciembre de 2010 y cuyos legajos se encuentran en el rango 30000 a 36000*.

3. Dado un rango de claves, devolver la historia de todos los objetos cuyas claves se encuentran dentro de ese rango. Por ejemplo, *devolver los cambios de domicilios de los alumnos cuyos legajos están en el rango 15200 a 15250*.

Los tres casos descriptos se transforman en casos especiales cuando los intervalos de tiempo se reducen a instantes o cuando en lugar de considerar un rango de claves, solo se tiene en cuenta una. Por lo tanto, se pueden tener *intervalos* o *instantes* de tiempo, *rangos* de claves o clave *única* y tiempos *válido*, *transaccional* o *bitemporal*.

La clasificación anterior de las consultas no es completa; solo contiene consultas básicas no triviales. En particular, los tipos 1 y 2 se refieren a consultas basadas en intersección, es

decir, la respuesta consiste en objetos cuyo intervalo contiene algún punto de tiempo en común con la consulta. Existen otros tipos de consultas que puede ser de importancia, de acuerdo al problema. Por ejemplo, *hallar el objeto con tiempo posterior más cercano a un punto temporal dado* u *obtener todos los objetos con intervalos contenidos estríctamente en un intervalo dado*.

Una forma más general de distinguir tipos de consultas temporales es la notación de tres entradas llamada: *key/valid/transaction* [SJ96]. Esta notación determina cuales entradas están involucradas en la consulta y de qué manera. Cada entrada puede contener alguno de los valores *point*, *rank*, $*$, o $-$. El valor *point* indica que la consulta hace referencia a una única clave si se refiere al atributo clave del objeto, o a un instante si se refiere al tiempo. El valor *rank* representa un rango de claves o un intervalo de tiempo. Y los valores $*$ y $-$ significan "todos" y "no aplicable", respectivamente. Cuando alguno de estos cuatro valores se ubica en el segundo lugar, se refiere al tiempo válido; y si está en el tercer lugar de la terna, será tiempo transaccional. De esta manera, es posible especificar consultas en tiempo válido, transaccional o bitemporales.

Algunos ejemplos de tipos de consultas especificados en esta notaciíon son:

***/-/point** también llamada *transaction pure-timeslice query*, denota la consulta en tiempo transaccional instantánea pura, es decir, en la cuál se devuelven todos los objetos que fueron registrados en la base de datos en un instante de tiempo dado.

***/-/rank** similar a la anterior, pero especifica un intervalo de tiempo transaccional en lugar de un instante.

point/*/- denominada *valid pure-key query*, devuelve la historia (en tiempo válido) correspondiente al objeto cuya clave se especifica.

rank/-/rank devuelve todos los objetos cuyas claves se encuentran en el rango ingresado, cuyos tiempos transaccionales se encuentran dentro del intervalo.

point/point/rank consulta bitemporal que consiste en hallar el objeto correspondiente a la clave ingresada, con vigencia en un instante de tiempo determinado, que fue registrado dentro del intervalo de tiempo ingresado.

rank/rank/rank esta es la consulta más generica que permite esta notación, es decir, especificando un rango de claves, un intervalo de tiempo válido y un intervalo de tiempo transaccional.

La notación de tres entradas utiliza solo la intersección de rangos y puntos, pero puede extenderse para incorporar consultas temporales del tipo antes/después o de algún otro tipo.

3.8. Métodos de Acceso

Generalmente, si un método de acceso resuelve eficientemente las consultas instantáneas, también es eficiente para consultas por intervalos, por lo cual usualmente se considera la con-

sulta instantánea como representativa (en cuanto a eficiencia) de ambas. Como los métodos de acceso en la mayoría de los casos se han diseñado para ser eficientes en la resolución de una clase de consultas particular, en esta sección se presentan los métodos separados en clases [ST99].

En el caso de las bases de datos transaccionales, todo método de acceso requiere al menos:

- Almacenar sus estados lógicos pasados
- Capacidad para realizar cambios (agregar/modificar/eliminar) de los estados lógicos actuales de los objetos
- Consultar eficientemente los objetos en cualquiera de sus estados

Debido a que un hecho puede ingresarse en la base de datos en un tiempo diferente al que ha sucedido en la realidad, el intervalo del tiempo transaccional asociado al hecho usualmente no coincide con la validez del mismo en la realidad, y solo representa el proceso de actualización de la base de datos.

Una base de datos de tiempo válido tiene una abstracción diferente. Para entenderla, considere una colección dinámica de *objetos-intervalo*. Usamos este término para enfatizar el hecho de que el objeto tiene un intervalo asociado que indica el período de validez de alguna de sus propiedades (en contraste, en una base de datos de tiempo transaccional, hablamos de *objetos-planos*). Los cambios admitidos son inserción/eliminación/modificación de los *objetos-intervalo*, pero la evolución de la colección (estados pasados) no se guarda. Por ejemplo, considere el conjunto de proyectos que lleva a cabo una empresa constructora. Cada proyecto tiene un intervalo asociado (al principio con la fecha de inicio y el valor *now*) representando el periodo de ejecución del proyecto. Asumimos que si se aplica una modificación al contrato, solo se mantiene el último estado. Una base de datos de tiempo válido es la adecuada para esta situación.

La noción de tiempo en este caso se relaciona al eje de tiempo válido. Dado un punto de tiempo, los *objetos-intervalo* pueden clasificarse como pasados, futuros o actuales (vigentes) en relación a ese punto, si su intervalo de tiempo es anterior, posterior o contiene al punto dado. Las bases de datos que contemplan esta dimensión pueden corregir errores en todo el dominio del tiempo (pasado, actual o futuro) ya que los registros de cualquier *objeto-intervalo* pueden modificarse independientemente de su posición en el eje de tiempo.

Los requisitos mínimos a tener en cuenta en los método de acceso de tiempo válido son:

- almacenar la colección de objetos-intervalos actual,
- permitir inserciones/eliminaciones/modificaciones sobre dicha colección,
- consultar eficientemente los *objetos-intervalo* que contiene la colección.

El modelo de la realidad es más completo si se soportan ambas dimensiones, es decir, con una base de datos bitemporal. Este tipo de base de datos puede verse como una colección de

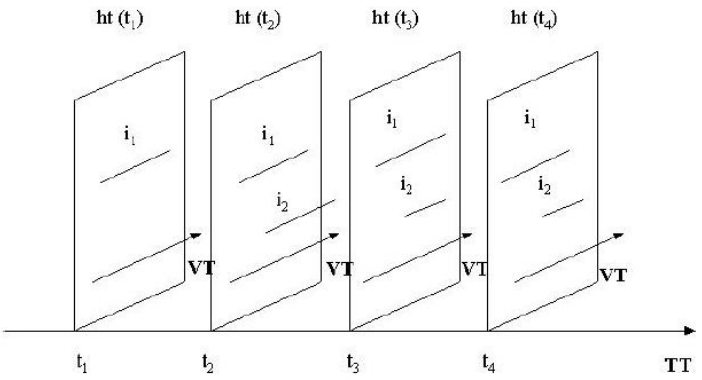

Figura 3.1: *Vista conceptual de una base de datos bitemporal*

objetos-intervalos de tiempo válido que evolucionan en el tiempo. Dicha evolución representa el tiempo transaccional. La Figura 3.1 muestra una vista conceptual de una base de datos bitemporal. En lugar de una simple colección de *objetos-intervalo* hay una secuencia de colecciones indexadas por el tiempo transaccional [ST99].

Para los métodos de acceso bitemporal, los requisitos mínimos son:

- almacenar los estados lógicos pasados,
- soportar inserción/eliminación/modificación sobre los *objetos-intervalo* de su estado lógico actual, y,
- consultar eficientemente los *objetos-intervalo* en cualquiera de sus estados.

En resúmen, una base de datos de tiempo transaccional difiere de una base de datos bitemporal en que mantiene la evolución histórica de *objetos-planos* en lugar de *objetos-intervalo*, mientras que una base de datos de tiempo válido difiere de una bitemporal ya que mantiene solo la última colección de *objetos-intervalo*. En la Figura 3.1, cada colección $ht(t_i)$ puede verse como una base de datos de tiempo válido separada.

3.8.1. Métodos de Acceso de Tiempo Transaccional

Estos métodos soportan cambios que suceden en orden, una característica propia del tiempo transaccional (Figura 3.2). Esta propiedad tiene una influencia enorme en la forma de actualización de la estructura. Si se permitiesen cambios fuera de orden temporal, el costo de actualización se volvería prácticamente inaceptable.

Los métodos orientados al tiempo transaccional se pueden clasificar en *key-only*, *time-only* y *time-key*, de acuerdo a la manera en que los datos son agrupados.

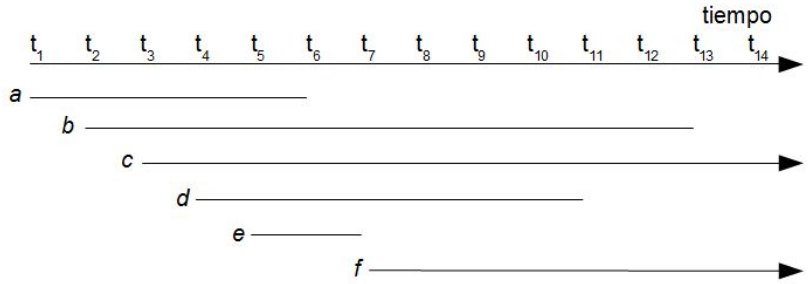

Figura 3.2: *Ejemplo de objetos en una base de datos de tiempo transaccional. Los cambios suceden en orden creciente en el tiempo. Las líneas representan la "vida"de cada objeto. Las flechas significan que el objeto sigue vigente en el tiempo actual.*

3.8.1.1. Métodos Key-Only

La característica fundamental de estos métodos es la organización de los datos agrupados por clave, de tal manera de que todas las versiones de un mismo objeto se encuentran juntas lógica o físicamente, y ordenadas en forma ascendente. Dicha organización hace a estos métodos más eficientes ante consultas por clave pura, ya que solo se debe devolver la evolución de los objetos cuyas claves se encuentran en el rango dado (consultas tipo $point/-/*$ y $rank/-/*$).

A continuación se presentan los métodos de acceso de tiempo transaccional orientados a consultas *key-only* y sus características más importantes.

- **Reverse Chaining:** en esta estructura, todas las versiones de un mismo objetos se encadenan en orden cronológico inverso. También se introduce la posibilidad de separar los datos actuales de los antiguos, considerando situaciones en las cuales es más probable que se consulten con mayor frecuencia los primeros. De esta manera la estructura que mantiene la situación actual es de menor tamaño y más eficiente. Cada versión del objeto se representa por una tupla que contiene la clave, atributos, el tiempo y un puntero a la versión anterior [BZ82].

- **Accession Lists:** es una mejora del método anterior en la cual se indexan las distintas versiones de un objeto por tiempo para que la búsqueda de una versión en particular de un objeto se resuelva con costo logarítmico en lugar de lineal (recorrido en forma secuencial la lista de versiones) [AS88].

- **Time Sequence Arrays (TSAs):** conceptualmente es un array de dos dimensiones que contiene una fila por cada clave creada. Cada columna representa un instante de tiempo. Es decir, cada entrada (x, y) almacena una clave x en el instante de tiempo y. Es eficiente en cuanto al tiempo de consulta, pero con un alto costo de actualización y de almacenamiento [SK86].

- **C-Lists:** estructura similar a las *Accession Lists*, con dos diferencias importantes: el acceso a cada *C-List* se realiza mediante otra estructura, denominada *Multiversion Access*

Structure (MVAS) y su forma de actualización tiene mayor complejidad y costo. Su ventaja principal es que responde con gran eficiencia tanto a las consultas de clave pura ($point/-/*$) como a las por rango-instantáneas ($rank/-/point$) [VV97].

3.8.1.2. Métodos Time-Only

En general, estos métodos asocian a los cambios (inserciones, eliminaciones, etc) el tiempo en el que se ejecutó la transacción, y los agregan generando una especie de *"history log"*. Estos métodos intentan optimizar las consultas de tipo $*/-/point$ y $*/-/rank$. Ya que las modificaciones ocurren en orden cronológico, los métodos *Time-Only* pueden procesar actualizaciones en forma continua (porque todo cambio consiste en insertar un registro al final del *history log*) a un bajo costo. Esto es muy importante en aplicaciones donde los cambios son frecuentes y se requiere que se registren *on-line*. Para que las consultas sean eficientes, la mayoría de estos métodos utiliza algún tipo de índice sobre los *history logs*.

A continuación se presenta una breve descripción de algunos métodos de acceso de tiempo transaccional orientados a consultas *time-only* y sus características más importantes.

- **Append-Only Tree (AP-Tree):** es un árbol de búsqueda de muchos caminos, que combina características del índice *ISAM* con el $B^+ - Tree$. A cada tupla se le asocia un intervalo $(startTime, endTime)$ y se indexan las tuplas por el tiempo de inicio. Cada nodo hoja tiene entradas de la forma (t, b) donde t es un instante de tiempo y b es un puntero al grupo de tuplas con tiempo de inicio mayor que el registrado en la entrada anterior, y menor o igual a t [SG89].

- **Nested ST-Tree:** es una variante del *AP-Tree* que permite realizar consultas de clave pura ($point/-/*$) y rango-instantánea ($rank/-/point$) mediante el agregado de un $B^+ - Tree$ sobre todas las claves insertadas en algún momento en la base de datos [GS93].

- **Time Index:** es un metodo de acceso basado en la construcción de un $B^+ - Tree)$ sobre el eje del tiempo. Cada punto del eje corresponde al instante en que se produjo algún cambio. Cada nodo hoja contiene un instante de tiempo y un apuntador al grupo de todos los objetos que están *"vivos"* en ese instante (es decir, una ïnstanánea" del sistema). Como resultado de este diseño, el índice no necesita saber de antemano el tiempo final de un objeto, lo cual representa una ventaja sobre el *AP-Tree* [EWK90].

- **Archivable Time Index:** este índice no indexa en forma directa instantes de tiempo, sino números de versiones. El instante de tiempo de la primer transacción toma el número de versión 0 y los cambios posteriores se mapean a números consecutivos de versión. Un intervalo se representa por los números de versión correspondientes al tiempo de inicio y tiempo final. Este índice requiere una estructura especial que convierte los números de versión en tiempos reales y viceversa. El método divide los registros en actuales (sin tiempo final) y pasados. Para los actuales utiliza un $B^+ - Tree)$ sobre los tiempos de inicio, y los registros pasados los indexa mediante una árbol binario lógico denominado *PVAS* [VV94].

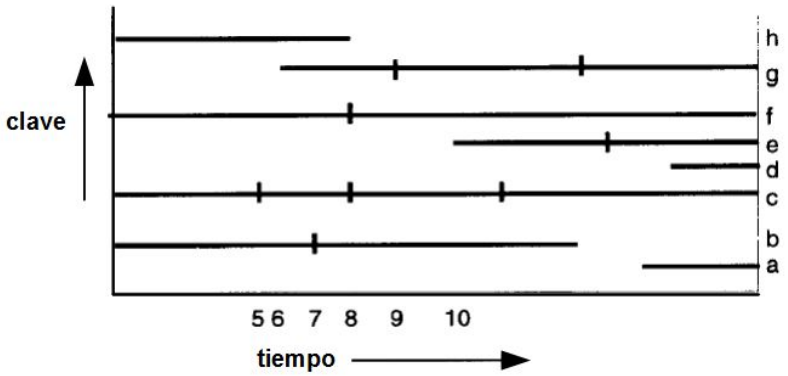

Figura 3.3: *Una página almacenando un rango del espacio bidimensional tiempo-clave. Las líneas verticales representan nuevas versiones del mismo objeto, es decir, actualizaciones. El final de una línea significa que el objeto fue eliminado*

- **Snapshot Index:** este índice logra la solución óptima para las consultas pura-instantáneas de tiempo transaccional ($*/-/point$). Conceptualmente consiste en tres estructuras de datos: un índice multinivel que permite el acceso al pasado en un tiempo t; una estructura multiencadenada entre las páginas de cada hoja del índice multinivel, que facilita la respuesta a la consulta sobre el instante t; y una función *hashing* que provee acceso a los registros por clave, utilizada para mejorar la velocidad de las actualizaciones. [TK95].
- **Windows Method:** es una alternativa al método anterior para la resolución de consultas $*/-/point$ que posee el mismo grado de eficiencia. Es una versión adaptada a disco de una estructura anterior que resuelve eficientemente el problema en memoria principal. Para ello se particiona el tiempo en porciones (*"windows"*) contiguas y se asocia a cada porción una lista con los intervalos que intersectan el intervalo de la misma. Cada porción es indexada por un *B-Tree* [Ram97].

3.8.1.3. Métodos Time-Key

Para responder consultas del tipo $rank/-/point$ eficientemente, es una buena estrategia almacenar los datos relacionados por clave y tiempo dentro de una misma página, para minimizar la cantidad de acccesos a disco (Figura 3.3). Los métodos desarrollados para resolver este ipo de consultas utilizan árboles balanceados cuyas hojas almacenan dinámicamente porciones del espacio bidimensional *time-key*. Mientras que los cambios ocurren en orden de tiempo creciente, no sucede los mismo con las claves ya que pueden ingresarse en cualquier orden, teniendo como consecuencia que el proceso de actualización tenga como mínimo, costo logarítmico.

Hay dos enfoques principales en el diseño de estos métodos: basados en variantes de *R-Trees* y basados en variantes de B^+*-Trees* [ST99]. Una ventaja muy importante del uso de métodos basados en *R-Trees* es que éstos pueden representar dimensiones adicionales sobre el mismo índice (en especial, permiten representar ambas dimensiones de tiempo). Una desven-

taja de estos métodos es que no pueden garantizar una buena performance de las consultas y actualizaciones para el peor de los casos; sin embargo, estos casos raramente ocurren. En la práctica, los *R-Trees* han mostrado una buena performance para el caso promedio. Otra característica de los métodos basados en *R-Trees* es que el tiempo final de un intervalo se asume conocido cuando se agrega un objeto, que no es una propiedad del tiempo transaccional.

A continuación se presenta un breve resúmen de los métodos de acceso de tiempo transaccional orientados a consultas *time-key* y sus características más importantes.

- **Métodos basados en R-Tree:** En [KS89] se presenta el *Dual R-Tree*, que mantiene dos R-Tree con sus raíces en memoria secundaria. Uno de ellos completamente almacenado en el disco principal y el otro, salvo su raíz, en memoria secundaria de archivo. Cuando el primer árbol alcanza la altura del segundo, se dispara un proceso que transfiere los nodos de este árbol al segundo, salvo su raíz, que siempre queda almacenada junto al primer árbol. Las consultas se realizan recorriendo ambos árboles. El método prioriza la aficiencia de las consultas cercanas al presente. En [KS91] se propone el *Segment R-Tree (SR-Tree)* que combina propiedades del *R-Tree* con el *Segment Tree*, un ábol binario que se utiliza para almacenar segmentos de líneas. Un *SR-Tree* es un *R-Tree* que permite guardar intervalos tanto en las hojas como en los nodos interiores. Los intervalos de mayor duración se guardan en los nodos superiores, de tal manera de disminuir la superposición en los nodos hoja. Los mismos autores proponen una variante de este último, denominada *Skeleton SR-Tree* que preparticiona el dominio completo en regiones, basado en la distribución de los datos y la cantidad de intervalos a ser agregados. El *SR-Tree* y sus variantes se pueden utilizar también en bases de datos de tiempo válido.
- **Métodos basados en B-Tree:** El *Write-Once B-Tree* propuesto en [Eas86] estuvo originalmente diseñado para bases de datos que residen en dispositivos que sólo se pueden grabar una vez (*WORMs*). Esto implica que una vez que una página se escribe, no puede modificarse. Por lo tanto, cada nueva entrada en el índice ocupa una página completa, ya sea el ingreso de un nuevo objeto o el registro de una nueva versión del mismo. Los nodos del árbol son colecciones de objetos (almacenados, por ejemplo, en un pista). Cada registro almacena solo el tiempo de inicio. El tiempo final es el tiempo de comienzo de la nueva versión. El *Time-Split B-Tree* [LS89], el *Persistent B-Tree* [LM91], el *Multiversion B-Tree* [BGO$^+$96] y el *Multiversion Access Structure* [VV97] son variantes de este método que hacen uso eficiente tanto de dispositivos *WORMs* como de discos magnéticos.

3.8.2. Métodos de Acceso de Tiempo Válido

Una base de datos de tiempo válido debe mantener una colecciíon dinámica de *intervalos-objetos*, por lo cual el problema de indexar este tipo de base de datos se puede reducir al de indexar colecciones dinámicas de intervalos. Sin embargo, el hecho de que sean dinámicas aumenta la complelidad del problema ya que las eliminaciones son ahora físicas y las inserciones pueden ocurrir en cualquier instante de la dimensión del tiempo. Para realizar esta indexación, se pueden usar R-Trees o métodos de acceso dinámicos similares. En general se han propuesto pocos enfoques para la indexación de bases de datos de tiempo válido y dada la dificultad

del problema, la mayoría de esos enfoques sólo intentan lograr un buen comportamiento para el peor caso, dando como resultado índices cuyo importancia principalmente es teórica. En [AV96] se presenta el *External Interval Tree*, que constituye una solución óptima en cuanto al tiempo de E/S, pero solo para las consultas de tipo $*/point/-$.

En el Cuadro 3.1 se enumeran algunos métodos de acceso orientados a tiempo válido y sus características principales.

Métodos de Tiempo Válido		
Método	**Referencia**	**Características**
Priority Search Tree	[McC85]	Representa intervalos como puntos en un espacio bidimensional. Provee una solución óptima en memoria principal para la consulta instantánea válida
Cell-tree	[Gün89]	Representa los intervalos como segmentos en un espacio bidimensional clave-tiempo. Tiene buena eficiencia para el caso promedio, pero el manejo de la superposición es un problema, especialmente si la distribución de los intervalos es altamente no-uniforme
External Segment Tree (EST)	[BG94]	Adaptación del Segment-Tree a memoria secundaria. Además de responder eficientemente a consultas $*/point/-$, puede ser modificado para manejar consultas del tipo $point/point/-$ o $rank/point/-$
Path Caching	[RS94]	Resuelve consultas en dos dimensiones de rangos. Se basa en la adaptación a memoria secundaria de estructuras orientadas a memoria principal, tales como el Interval-Tree o el Priority Search Tree
MAP21	[ND99]	Transforma un intervalo en un único valor, mediante una simple fórmula, e indexa los puntos resultantes mediante un B-Tree. Asume un dominio de tiempo limitado y conocido
Relational Interval Tree (RI-Tree)	[KPS00]	Orientado al acceso a datos con intervalos contenidos en tablas relacionales u objeto-relacionales. Eficiente, pero con un alto costo en espacio de almacenamiento.
External Interval Tree	[AV96]	Solución óptima en cuanto a tiempo de E/S, pero sólo para consultas instantáneas
Triangular Decomposition Tree (TD-Tree)	[STTS10]	Eficiente ante una gran variedad de tipos de consulta, incluyendo consultas instantáneas, consultas por intersección y consultas no triviales sobre relaciones entre intervalos. Es un método de acceso de particionamiento del espacio. La idea básica es manejar los intervalos temporales mediante una estructura virtual basada en la representación bidimensional de los intervalos y en el método de descomposición triangular. El resultado es un árbol binario (usualmente desbalanceado) que almacena una limitada cantidad de intervalos en cada hoja. Puede ser implementado sobre un DBMS relacional

Cuadro 3.1: *Métodos de Acceso de Tiempo Válido*

3.8.3. Métodos Bitemporales

Un índice bitemporal es una estructura de datos que soporta tanto el tiempo transaccional como el tiempo válido. A pesar de que los datos *now-relative* son parte natural de las bases de datos bitemporales, sólo unos pocos métodos de acceso tienen en cuenta su problemática. Una tupla se considera *now-relative*, si representa un hecho vigente en el presente. Es decir, no tiene un tiempo final fijo sino que se incrementa con el paso del tiempo, dando lugar a intervalos dinámicos. Varios índices asumen que el intervalo de tiempo válido es conocido cuando la información se registra en la base. Esta restricción hace que dichos métodos de acceso no resulten aptos para muchas aplicaciones temporales reales.

En el Cuadro 3.2 se enumeran algunos métodos de acceso bitemporales y sus características principales.

Métodos Bitemporales		
Método	***Referencia***	***Características***
Bitemporal Interval Tree (BIT)	[KTF98]	Es una versión modificada del Interval-Tree. Indexa objetos con rangos cerrados de tiempo válido y tiempo transaccional *now-relative*. Es un índice particalmente persistente y tiene buena paginación. Responde consultas tipo $*/point/point$ y $*/rank/point$ en tiempo logarítmico. Tiene alto costo espacial y sólo es eficiente si la duración del tiempo válido es pequeña en comparación con la del tiempo transaccional
Multi Incremental Valid Time Tree (M-IVTT)	[NDE96]	Representa datos bitemporales almacenando una secuencia de colecciones de intervalos junto con los cambios que se suceden entre dichas colecciones. Cada colección representa el estado de la base de datos en un instante de tiempo transaccional y se indexa utilizando el método MAP21.
Bitemporal R-Tree (BRT)	[KTF98]	Es un R-Tree con persistencia parcial (cada modificación genera una nueva versión, manteniendo la anterior intacta y sólo se puede hacer cambios en la última). Su estructura está conformada por una secuencia de R-Trees correspondientes a la evolución de los objetos en el tiempo transaccional. Utilizando la persistencia parcial, cada R-Tree representa un ïnstante de historia". Se le puede agregar una dimensión más para representar el atributo clave de cada objeto-intervalo y así poder responder con eficiencia consultas del tipo $rank/rank/point$
2R-Tree	[KTF98]	Utiliza dos R-Trees; el primero de ellos almacena todas las tuplas que no poseen tiempo final, es decir, que están vigentes desde el punto de vista transaccional, mientras que el segundo mantiene las tuplas con tiempo final ya definido. Cuando se actualiza el primer árbol, las tuplas modificadas (a las cuales se les agregó el tiempo final) se eliminan de dicha estructura y se agregan al segundo árbol.
HR-Tree	[NS98]	Inicialmente se propuso para indexar bases de datos espacio-temporales. Utiliza como base un único R-tree de Hilbert. No causa duplicacíón de los datos, evita la re-inserción y es eficiente en las consultas, pero no permite trabajar con datos *now-relative* de tiempo válido
GR-Tree	[BJSS98]	Basado en R-Trees, soporta datos *now-relative* tanto de tiempo válido como de tiempo transaccional. Introduce los valores *now* y *until change (UC)* para representar el tiempo actual en tiempo válido y tiempo transaccional respectivamente. Su implmenetación en un BDMS relacional se considera costosa, ya que sería necesario modificar el kernel
Virtual Binary Tree (VB-Tree) y Virtual Binary Forest	[Sta05]	El VB-Tree permite datos *now-relative* de tiempo válido y responde con eficiencia a una gran variedad tipos de consultas sobre esta dimesión del tiempo. El VB-Forest está formado por una secuencia de VB-Tree que representa la evolución de los objetos en el sentido del tiempo transaccional, con el propósito ambas dimensiones estén indexadas

Cuadro 3.2: *Métodos de Acceso Bitemporales*

3.8.4. Método de acceso espacio-temporal $SEST_L$

En esta sección se describe el índice espacio-temporal orientado a eventos $SEST_L$ [Gut07]. Si bien no es un método de acceso temporal puro, se incluye aquí ya que se utilizó parte de su estructura en el diseño del aspecto temporal del *Event-FHQT*; uno de los índices métrico-temporales que se presentan en este trabajo.

Este método permite gestionar eventos discretos que representan cambios en atributos espaciales de los objetos de la base de datos. Combina instantáneas y eventos en estructuras de *logs* asociadas a particiones del espacio y está diseñado especialmente para responder consultas orientadas a eventos (aunque permite además las tradicionales consultas instantáneas y por intervalos). Es una mejora al índice denominado $SEST - Index$ presentado anteriormente.

Su estructura (Figura 3.4) se basa en único R-Tree que indexa todos los objetos de la base de datos en los distintos instantes de tiempo y utiliza instantáneas y listas de eventos ocurridos entre dos instantáneas consecutivas (*logs*) para representar los cambios. Estos *logs* se encuentran a nivel de las hojas del árbol y contienen dos tipos de entradas: eventos e instanáneas. La primer entrada de un *log* siempre es una instantánea. Los eventos se registran respetando el orden temporal y son de dos tipos: *move in* y *move out*. El primero representa la creación de un objeto y el segundo su eliminación. Para modelar el traslado de un objeto de un lugar a otro se

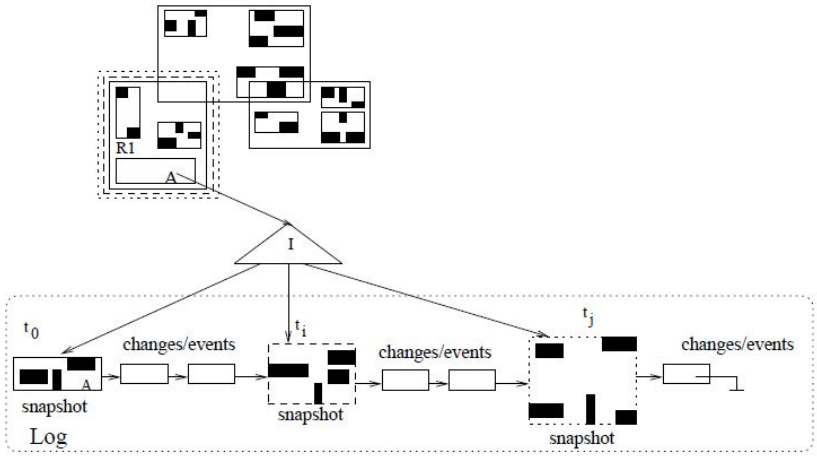

Figura 3.4: *Estructura del* $SEST_L$

utiliza un *move out* del lugar de partida y un *move in* al lugar de llegada.

Para resolver una consulta instantánea (*time-slice*), como primer paso se recorre el R-Tree para calcular las hojas cuyos rectángulos asociados se intersectan con la consulta. Luego, para el *log* de cada hoja se busca la instantánea más actual que sea menor o igual al instante de tiempo de la consulta y se aplican las actualizaciones registradas en el *log* hasta el instante consultado. Las consultas por intervalo de tiempo (*time-interval*) se resuelven en forma similar, buscando el instante de inicio y extendiendo los cambios hasta el instante final. Una novedad del $SEST_L$ es que además permite consultas orientadas a eventos, por ejemplo, *encontrar todos los objetos que ingresaron a una zona o salieron de ella en un instante dado*.

En el Caítulo 4 se presenta el índice *Event-FHQT* que utiliza la idea de registrar eventos a través de instantáneas y *logs* para responder eficientemente a consultas métrico-temporales.

Capítulo 4

El Modelo Métrico-Temporal

4.1. Introducción

En este capítulo 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, soportadas por el Modelo Temporal. También se presentan dos métodos de acceso métrico-temporales y sus variantes.

Consideremos el ejemplo descripto anteriormente en el cual se registran las huellas digitales de las personas que entran y salen de un museo o una exposición de arte, junto a su fecha y hora. Algunas consultas métrico-temporales que tienen sentido son:

1. Encontrar todos los ingresos/egresos de las personas cuya huella digital coincide (con un cierto radio de tolerancia) con una dada.
2. Encontrar los ingresos/egresos sucedidos dentro de un intervalo de tiempo, de las personas cuya huella digital coincide (con un cierto radio de tolerancia) con una dada.
3. Hallar la persona cuya huella digital tiene mayor parecido a una dada y que estuvo en el lugar en una fecha y hora determinada.
4. Hallar las diez personas con mayor parecido a una dada, que estuvieron en el lugar antes de una fecha y hora determinada.

Estas consultas no pueden ser resueltas eficientemente con índices métricos ya que no tienen en cuenta el aspecto temporal, ni con índices temporales, debido a que es necesario que la comparación sea por similitud. Así como ha sucedido anteriomente en la integración de los modelos espacial y temporal, que ha dado lugar a los índices *espacio-temporales*, es necesario definir un modelo que combine el modelo métrico con el temporal y permita generar métodos de acceso adecuados para realizar este tipo de consultas eficientemente.

El modelo *métrico-temporal* está orientado a satisfacer búsquedas sobre objetos no estructurados que poseen uno (una sola dimesión temporal) o dos (bitemporal) instantes o intervalos de tiempo asociado y que además no pueden ser recuperados a través de un atributo clave a través de una búsqueda exacta.

4.2. 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 $d : U \times U \to \mathbb{R}^+$. 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.

4.2.1. Caracterización de un problema Métrico-Temporal

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 sí no contiene la clave. Por ejemplo, en una base de datos de música, cada canción puede contener una clave que la identifica, pero se puede requerir buscar una canción que sea similar a un fragmento de canción dado, justamente porque no se conoce su identificador.
- 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. Puede ser requerido 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.

4.3. 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 [SJ96]. 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, algunos de los tipos de consultas métrico-temporales más importantes son:

1. *range/rank/-* Devuelve todos los objetos similares a uno dado, dentro de un radio de tolerancia, que poseen un tiempo de validez 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:

   $$(q, r, t_{vi}, t_{vf})_d = \{o/(o, t_{vio}, t_{vfo}, t_{tio}, t_{tfo}) \in X \wedge d(q, o) \leq r \wedge (t_{vio} \leq t_{vf}) \wedge (t_{vi} \leq t_{vfo})\} \tag{4.1}$$

   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. En este tipo de consulta no se utiliza el tiempo transaccional.

2. *range/point/-* Consulta similar a la anterior, pero con $t_{vi} = t_{vf}$.

3. *range/-/rank* Devuelve todos los objetos similares a uno dado, dentro de un radio de tolerancia, que poseen un tiempo transaccional que se intersecta con un intervalo dado.

4. NN_k*/rank/-* Devuelve los k objetos con mayor similitud a uno dado, que poseen un tiempo de validez que se intersecta con un intervalo dado.

5. NN_k*/rank/rank* Devuelve los k objetos con mayor similitud a uno dado, que poseen un tiempo de validez que se intersecta con un intervalo dado y un intervalo transaccional con algún punto en común con un segundo intervalo de entrada.

De la misma manera se pueden definir las demás combinaciones posibles. Las consultas del tipo *range/-/-* y NN_k*/-/-* representan consultas métricas puras (que devuelven la evolución de los objetos en el tiempo), mientras que *-/rank/-* o *-/point/rank* representan consultas temporales puras; la primera de tiempo válido y la segunda bitemporal.

Algunas formas triviales de resolver consultas métrico-temporales sin recorrer la base de datos completa son:

- *Utilizando un índice métrico*: ante una consulta se procede 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

- *Utilizando un índice temporal o bitemporal*: ante una consulta se procede de la siguiente manera:

  1. Realizar una búsqueda sobre el índice temporal
  2. Realizar una búsqueda secuencial por similitud sobre el conjunto de objetos resultantes del paso anterior

- *Utilizando un índice métrico y un índice temporal*: ante una consulta se procede de la siguiente manera:

  1. Realizar una búsqueda por similitud utilizando el índice métrico
  2. Realizar una búsqueda utilizando el índice temporal
  3. Realizar la intersección de los conjuntos generados en los dos pasos anteriores

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.

A continuación se presentan los índices métrico-temporales desarrollados en esta tesis, orientados a consultas de los tipos *range/rank/-* (por rango de similitud e intervalo válido) y *range/point/-* (por rango de similitud e instante válido).

4.4. Métodos de Acceso Métrico-Temporales

Como parte de este trabajo de tesis, se desarrollaron los índices métrico-temporales *FHQT-Temporal*, que toma como base un *FHQT* al cual se le agregan intervalos de tiempo a cada nodo para manejar el aspecto temporal; el *FHQT*$^+$*-Temporal*, que es una variante del anterior que trabaja con funciones de distancia continuas, el *Event-FHQT*, basado en un índice espacio-temporal que además de permitir este nuevo tipo de consultas, soporta eventos temporales y el *Event-FHQT*$^+$, variante del anterior también para funciones de distancia continuas. Los cuatro métodos están definidos para una sola dimensión temporal; en este estudio, para tiempo válido.

4.4.1. 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.

Este índice permite resolver consultas por similitud puras, temporales puras y métrico-temporales. Como se basa en un FHQT, solo permite funciones de distancia discretas.

4.4.1.1. Estructura

Se define como un árbol r-ario donde el valor de r está determinado por la cantidad de los distintos valores posibles de la función de distancia. Ya que trabaja con distancias discretas, r tiende a ser un valor no muy grande.

Formalmente, un *FHQT-Temporal* es un árbol donde:

- todo nodo interior V es una 3-upla (t_{ini}, t_{fin}, {(d_1, h_1), (d_2, h_2),..., (d_m, h_m)}) donde:
  - $h_1..h_m$ son los m hijos del nodo V,
  - las d_i , para $i = 1..m$, son las distancias entre el pivote correspondiente al nivel de V y los objetos contenidos 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 tienen una estructura similar; están representadas por una 3-upla (t_{ini}, t_{fin}, {e_1, e_2,..., e_l}), donde:
  - los e_i para $i = 1..l$ son los l elementos que contiene la hoja, que a su vez están formados por tres componentes: el objeto o, el tiempo inicial del mismo t_{io}, y su tiempo final t_{fo}.
  - los valores t_{ini}, t_{fin} poseen el mismo significado que para los nodos interiores, pero aplicados a los elementos e_i.

En la Figura 4.1 se muestra el esquema genérico del *FHQT-Temporal*. La estructura es dinámica, permitiendo tanto altas como bajas ya sea de instantes o intervalos contenidos en el intervalo que el índice posee hasta el momento, como 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.

4.4.1.2. Consulta

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

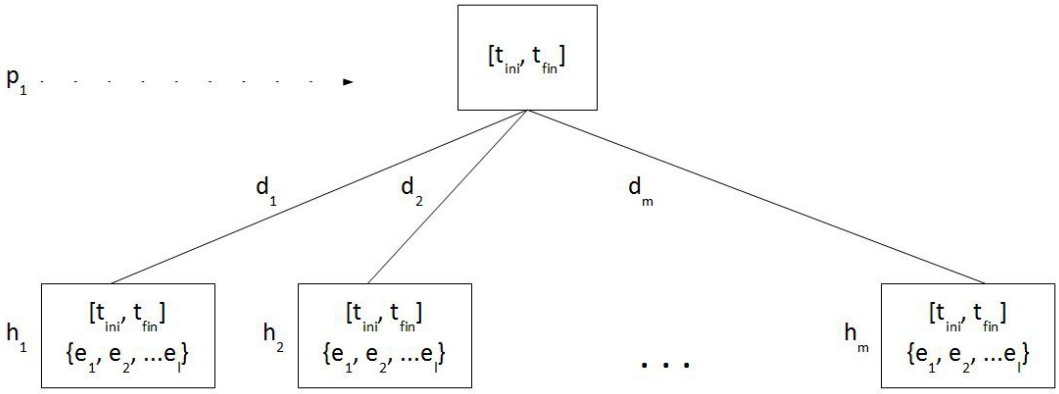

Figura 4.1: *Esquema del FHQT-Temporal*

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: $|d(q, p_{niv}) - d_j)| \leq r$, donde p_{niv} es el pivote correspondiente al nivel del nodo padre. 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 $d(q, o_i) \leq r$.

En la Figura 4.2 se muestra un ejemplo concreto del *FHQT-Temporal* con dos pivotes. Ante la consulta $(q, 2, 40, 52)_d$ y siendo $(4, 2)$ la firma de q, se procede de la siguiente manera. En primer lugar se controla si hay superposición temporal entre el intervalo de la consulta $[40, 52]$ y el intervalo correspondiente a la raíz $[1, 85]$. Como tienen instantes en común, se procede a verificar la condición de similitud con las distancias asociadas a los hijos.

Los nodos con distancias $3, 4, 6$ cumplen con este requisito, ya que se encuentran en el rango $[4 - 2, 4 + 2] = [2, 6]$. El subárbol asociado a la distancia 3, directamente se descarta ya que su intervalo temporal es $[9, 37]$; que es anterior a $[40, 52]$. El subárbol correspondiente a 4 tiene asociado el intervalo $[1, 55]$ y contiene 3 hijos. De sus hijos, el correspondiente a la distancia 7 se descarta y se tratan el 1 y el 4, ya que la condición de similitud en este segundo nivel, es que se superpongan al intervalo $[2 - 2, 2 + 2] = [0, 4]$. La hoja correspondiente a la rama $(4, 4)$ tampoco tiene superposición temporal (intervalo asociado $[1, 37]$) por lo cual solo se procesa la hoja $(4, 1)$ –cuyo intervalo es $[50, 55]$–, recorriendo los objetos o_8 y o_9 y verificando las restricciones temporal y de similitud con la consulta q. De la misma manera se procesan

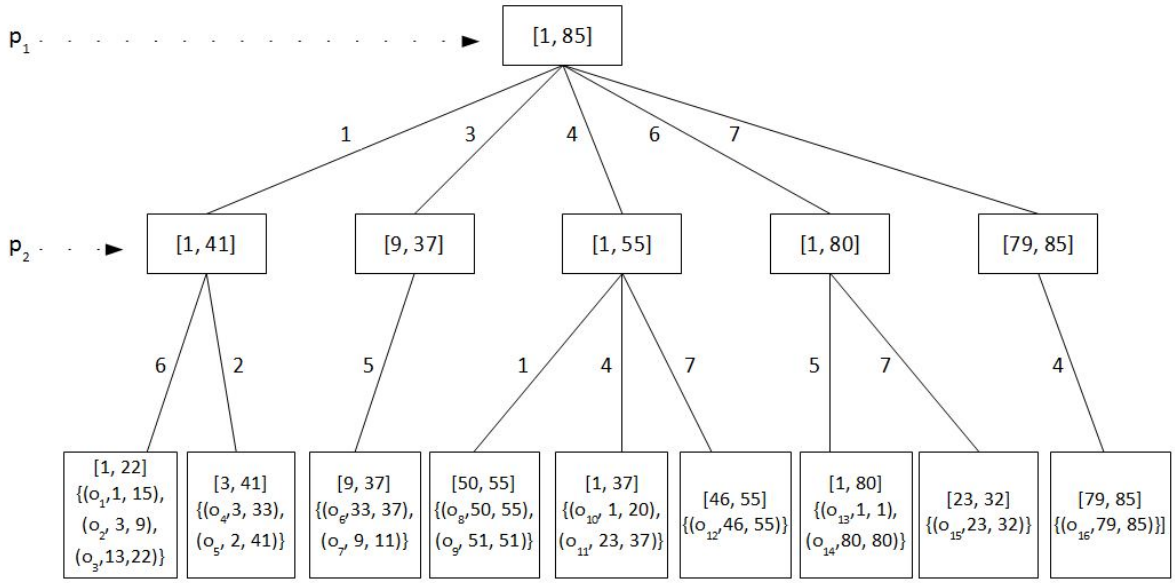

Figura 4.2: *Ejemplo de un FHQT-Temporal*

las ramas $(6, 5)$ y $(6, 7)$, aunque sin objetos resultantes, ya que no satisfacen ninguna de las condiciones.

En este ejemplo, el costo de la consulta es 4; dos evaluaciones de la función de distancia para calcular la firma de q, y otras dos evaluaciones para comparar q con los objetos candidatos o_8 y o_9.

El algoritmo de consulta por rango de similitud y tiempo válido (Figura 4.3) 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.

4.4.1.3. Construcción del índice

Una vez seleccionados los pivotes, ya sea al azar o mediante algún algoritmo particular, se registran en el índice y se construye el árbol insertando uno a uno los elementos de la base de datos. El procedimiento es similar al que se utiliza en el *FHQT*, pero además se actualizan los intervalos de tiempo de los nodos correspondientes a 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, se modifica este último asignándole el tiempo inicial del nuevo objeto. En forma similar, si el tiempo final del objeto es mayor que el tiempo final del nodo, se asigna a este último el nuevo valor. En la Figura 4.4 se presenta el algoritmo de inserción que utiliza el *FHQT-Temporal*. Este algoritmo también está organizado en dos partes. La primera calcula la

FHQT-Temporal $(q, r, t_{iq}, t_{fq})_d$
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** $|f_n - d_i| \leq r$ **then**
11. resultado :=resultado $\cup$ **Consultar**$(q, r, t_{iq}, t_{fq}, n+1, f, h_i)$
12. **return** resultado

Figura 4.3: *FHQT-Temporal: pseudocódigo de consulta por rango y tiempo válido*

firma del nuevo objeto y la segunda realiza la inserción.

4.4.1.4. Comparación del FHQT-Temporal versus la solución trivial

Para el análisis de la eficiencia de estos índices se toma como costo de una consulta métrico-temporal por rango y tiempo válido, la cantidad de evaluaciones de la función de distancia, ya que los índices se encuentran en memoria principal y se sabe que las funciones de distancia poseen un orden de complejidad mayor que las demás operaciones que forman parte del procesamiento de la consulta.

Las soluciones triviales planteadas en la Sección 4.3 tienen como costo mínimo el correspondiente a la búsqueda en un *FHQT* ($O(\log(n))$, donde n es la cantidad de objetos de la base de datos, y considerando $\log(n)$ pivotes) y en el peor de los casos, un recorrido secuencial de los objetos que se encuentran dentro del intervalo temporal de la consulta. Por otro lado, una consulta métrico-temporal utilizando un *FHQT-Temporal*, posee como *costo máximo* el costo de una consulta a un *FHQT*. Este caso se produce cuando no hay filtrado por tiempo o el filtrado es poco efectivo, ya sea porque el intervalo de tiempo de la consulta es muy amplio o porque existe una gran cantidad de pares de objetos con la misma firma que a su vez son muy distantes en el tiempo, lo que tiene como conecuencia que los intervalos de los nodos de la rama sean grandes y por lo tanto que disminuya significativamente la capacidad de filtrado temporal. En el ejemplo de la Figura 4.2 se ve que la rama$(6, 5)$ tiene el intervalo $[1, 80]$ para los nodos del segundo y tercer nivel y que los objetos que poseen la firma correspondiente a dicha rama son o_{13} y o_{14}. Estos objetos tienen asociados los instantes de tiempo 1 y 80, que se encuentran alejados en el tiempo. Como poseen la misma firma, los objetos quedan juntos y

FHQTT-Insercion $(o, t_{io}, t_{fo})_d$
1. calcular f de o – f es la *firma de o*
2. Insertar(o, t_{io}, t_{fo}, 1, f, $raiz$)

donde **Insertar** se define recursivamente como:

Insertar$(o, t_{io}, t_{fo}, n, f, x)$ – n es el nivel del nodo actual x
1. Sea $[t_{ix}, t_{fx}]$ el intervalo asociado al nodo x
2. **if** $t_{io} < t_{ix}$ **then**
3. $t_{ix} := t_{io}$
4. **if** $t_{fo} > t_{fx}$ **then**
5. $t_{fx} := t_{fo}$
6. **if** $n = k$ **then** –k es la cantidad de pivotes
7. agregar (o, t_{io}, t_{fo}) a x
8. **else**
9. nuevaRama := **true**
10. **for all** hijo $(h_i \in x)$
11. **if** $f_n = d_i$ **then**
12. **Insertar**$(o, t_{io}, t_{fo}, n+1, f, h_i)$
13. nuevaRama := **false**
14. **if** nuevaRama **then**
15. agregar el hijo h_{m+1} a x, con distancia f_n
16. **Insertar**$(o, t_{io}, t_{fo}, n+1, f, h_{m+1})$

Figura 4.4: *FHQT-Temporal: pseudocódigo de inserción de un nuevo objeto*

generan un intervalo amplio donde no existe ningún otro objeto, teniendo como consecuencia la degradación de la performance.

Por otro lado, cuando este tipo de situaciones no es frecuente y además las consultas son instantáneas o por intervalos reducidos, la eficiencia del *FHQT-Temporal* aumenta considerablemente ya que el filtrado temporal se hace más importante y supera a la solución trivial. En el Capítulo 5 se muestran resultados experimentales que muestran comparativamente la eficiencia de este índice versus la solución trivial.

Respecto al costo espacial, el *FHQT-Temporal* tiene prácticamente los mismos requisitos que un *FHQT*, ya que solamente se deben agregar los dos valores numéricos correspondientes al intervalo de tiempo de cada nodo. El complejidad espacial de un *FHQT*, medida en cantidad de punteros necesarios para construir la estructura, es nh, donde n es la cantidad de objetos y h la altura del árbol (considerando que la cantidad de pivotes $k = log(n)$ [CNBYM01]. El *FHQT-Temporal* tiene el mismo costo que el *FHQT*, si sólo se tiene en cuenta la cantidad necesaria de punteros para mantener la estructura.

4.4.2. FHQT$^+$-Temporal

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, que está incluido en el primero, 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, como veremos más adelante.

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 actualmente 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

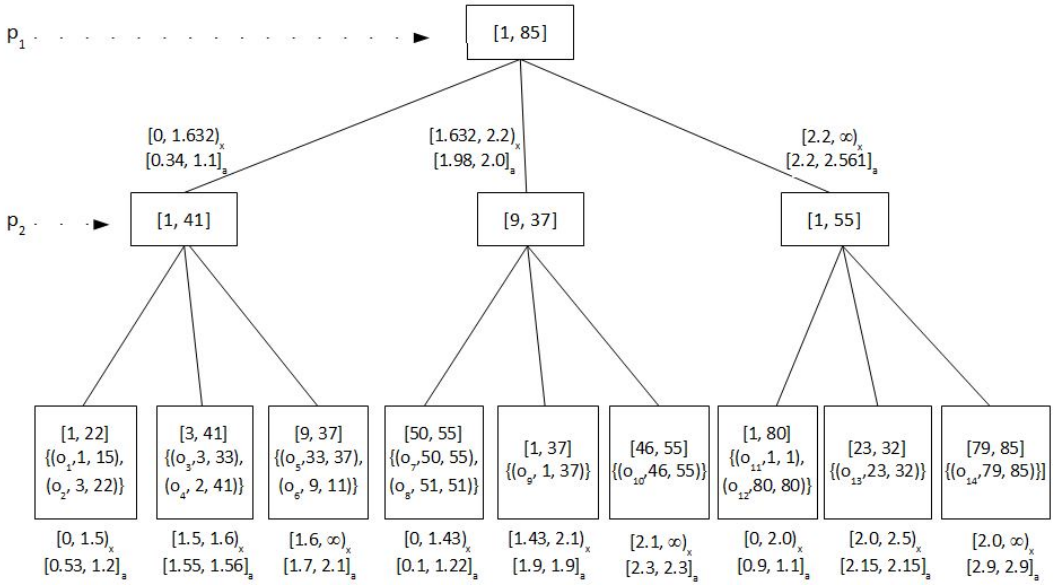

Figura 4.5: *Ejemplo de un FHQT$^+$-Temporal*

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*.

Ante una consulta, para visitar un nodo se comprueba que la distancia de la consulta al pivote del nivel pertenezca al intervalo actual asociado al nodo mas menos el radio de búsqueda, es decir, que $f_n \in [d_{ai} - r, d_{af} + r]$. En la figura 4.5 se muestra un ejemplo del *FHQT$^+$-Temporal*. Es interesante notar que los intervalos actuales correspondientes a los nodos de un mismo nivel, usualmente no cubren todo el espacio posible. Al reducir el tamaño de estos rangos, aumenta la probabilidad de que una rama se descarte ya que la comprobación anterior se realiza sobre un intervalo más pequeño. Por ejemplo, si el radio de búsqueda es $0{,}3$ y la distancia entre la consulta y el pivote del primer nivel es $1{,}5$, si se utilizan los rangos máximos se deben procesar el primer y segundo hijo del nodo raíz, ya que $1{,}5 \in [0 - 0{,}3\ ,\ 1{,}632 + 0{,}3)$ y $1{,}5 \in [1{,}632 - 0{,}3\ ,\ 2{,}2 + 0{,}3)$, mientras que usando los rangos actuales ambos se descartan porque $1{,}5 \notin [0 - 0{,}34\ ,\ 1{,}1 + 0{,}3]$ y $1{,}5 \notin [1{,}98 - 0{,}3\ ,\ 2{,}0 + 0{,}3]$.

Este índice permite resolver consultas por similitud puras, temporales puras y métrico-temporales con funciones de distancia tanto continuas como discretas. Esta versión de la estructura es la que se utilizó en los experimentos realizados como parte de esta tesis, dado que se trabajó con funciones de distancia continuas.

4.4.3. Event-FHQT

El $Event - FHQT$ utiliza ideas del método de acceso espacio-temporal $SEST_L$ [Gut07] y del índice métrico $FHQT$ [BYCMW94] y las integra en una sola estructura que responde consultas métrico-temporales. Está orientado a eventos, por lo cual los intervalos asociados a los objetos no se almacenan en un único registro sino como uno o más eventos. El Event-FHQT

contiene una *Línea del Tiempo* compuesta por intervalos de tiempo consecutivos donde cada uno tiene asociado el *snapshot* (instantánea) de todos los objetos vigentes en el primer instante de tiempo del intervalo, y listas de eventos ocurridos hasta el último instante de tiempo del mismo. Los *snapshot* junto con sus cambios están implementados a través de $FHQTs$ con listas de eventos en sus hojas.

Este diseño presenta la ventaja de ser mucho más estable en su eficiencia cuando existen objetos similares distantes en el tiempo, que como se mostró en la sección anterior, es una debilidad del *FHQT-Temporal*. Cuando los intervalos asociados a los objetos son pequeños en relación al tamaño de los intervalos de la *Línea del Tiempo*, el $Event-FHQT$ en la mayoría de los casos sólo registra las entradas y salidas de dichos objetos. Si los intervalos son grandes, se utiliza además un evento especial que representa la permanencia del objeto desde un *snapshot* al siguiente.

4.4.3.1. Estructura

El *Event-FHQT* está compuesto por una *Línea del Tiempo* de intervalos consecutivos, en principio de tamaño fijo. Cada intervalo contiene un FHQT que representa las relaciones de similitud entre los objetos vigentes en el intervalo. El FHQT se construye a partir de los objetos del primer instante y luego se actualiza con las entradas y salidas de objetos durante todo el intervalo. Las hojas del FHQT contienen listas de eventos que registran estas entradas y salidas de los objetos. El Event-FHQT posee un parámetro de configuración que especifica el tamaño de los intervalos; este valor se puede calcular en función de la densidad temporal de los eventos, es decir, la cantidad de eventos ocurridos por intervalo o instante de tiempo. El conjunto de pivotes es el mismo para todos los FHQTs. En la Figura 4.6 se muestra la estructura del Event-FHQT.

Formalmente, un *Event-FHQT* es una 3-upla (L_p, Tam, L_i) en la cual:

- $L_p = (p_1, p_2, \ldots, p_k)$ es la lista de pivotes utilizados en la construcción de los FHQT, donde k es la cantidad de pivotes.
- Tam es el tamaño de todos los intervalos de tiempo.
- L_i es una lista de pares *intervalo-FHQT* $(IF_1, IF_2, \ldots, IF_n)$ donde cada IF_i es de la forma $(I_i, FHQT_i)$, siendo $I_i = [t_i, t_f]$ el intervalo de tiempo. El t_i de todo intervalo salvo el primero, es igual al $t_f + 1$ del intervalo anterior. El segundo elemento del par es un FHQT modificado que contiene todos los objetos vigentes en al menos un instante del intervalo. Se define de la misma manera que un FHQT, a excepción de sus hojas, que contienen listas de eventos ordenadas en forma ascendente por instante de tiempo.
- Cada evento almacenado en una hoja de un FHQT se representa mediante una $3-upla$ de la forma $(instante, objeto, tipoDeEvento)$. El $instante$ en que se produce un evento se representa mediante un número natural. El campo $tipoDeEvento$ puede tomar los valores *in*, *out* o *stay*:

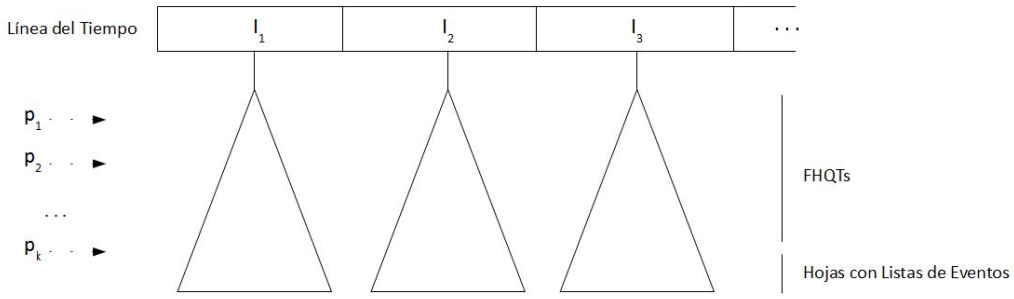

Figura 4.6: *Estructura del Event-FHQT*

- El evento *in* significa que el objeto comienza o comenzó a estar vigente en el instante asociado al evento. Este instante se corresponde con el $t_{inicial}$ del intervalo de vigencia del objeto.
- *out* representa que el objeto que deja o dejó de estar vigente en el instante asociado. Este instante es el t_{final} del intervalo de vigencia del objeto, es decir, el último instante en el cual el objeto es válido. Todo evento *out* debe ser precedido de un evento *in* o *stay* dentro del mismo FHQT, salvo que se encuentre en el primer instante del intervalo.
- *stay* indica que el objeto ya existía en el intervalo anterior y permanece vigente al menos en el primer instante del intervalo actual. Este último evento se utiliza únicamente por razones de eficiencia ya que de no hacerlo, para calcular los objetos vigentes en un intervalo habría que recorrer todos los FHQT de los intervalos anteriores. El instante correspondiente a un evento *stay* siempre es el primero del intervalo.

La estructura permite altas y bajas tanto históricas como de objetos en nuevos intervalos. Llamamos alta *histórica* a aquella cuyo intervalo o instante de tiempo está contenido en uno o más intervalos ya registrados en la *Línea del Tiempo*. Si el intervalo asociado a un objeto abarca más de un intervalo de la *Línea del Tiempo*, la inserción se realiza dividiendo dicho intervalo en subintervalos que coincidan o estén incluidos dentro de los intervalos ya definidos en la estructura, para luego dar de alta al objeto en los FHQT correspondientes a cada uno de ellos.

El costo del alta, medido en cantidad de evaluaciones de la función de distancia es constante e igual a k –la cantidad de pivotes del *Event-FHQT*– porque solo debe calcular la firma del objeto una vez, debido a que el conjunto de pivotes es el mismo para todos los FHQTs.

4.4.3.2. Consulta

El *Event-FHQT* resuelve consultas métrico-termporales, métricas puras y temporales puras. Sea $(q, r, t_{iq}, t_{fq})_d$ una consulta de tipo *range/rank/-* o *range/point/-* con función de distancia

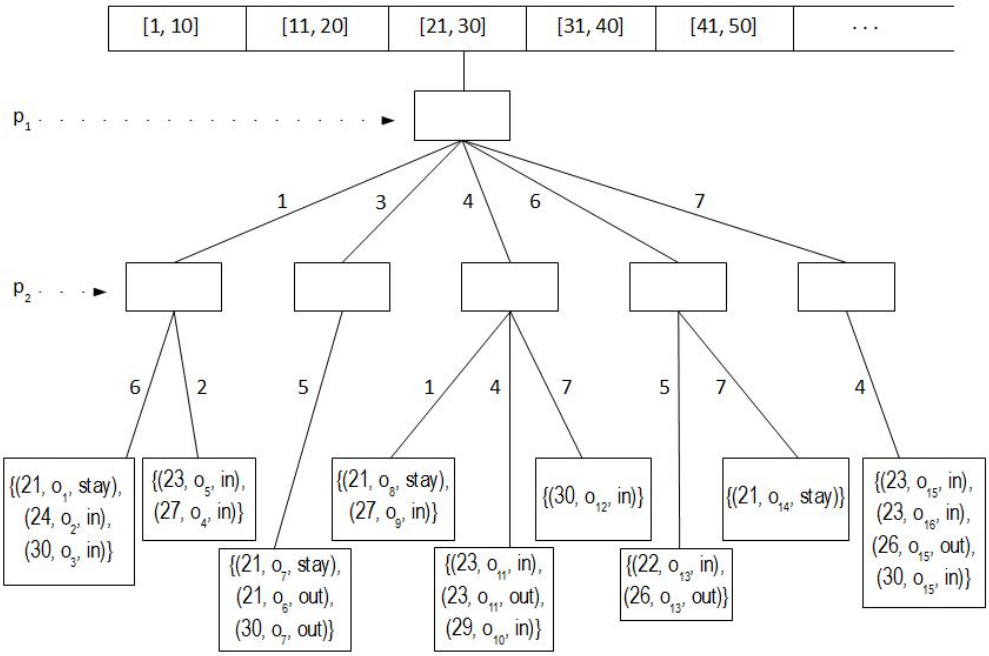

Figura 4.7: *Ejemplo de un Event-FHQT*

d, radio r, objeto q e intervalo o instante de tiempo $[t_{iq}, t_{fq}]$, para su resolución se procede conceptualmente en cuatro pasos:

1. Determinar qué intervalos de la lista se superponen con el intervalo de consulta (primer filtro temporal).
2. Para cada intervalo, realizar la consulta por similitud a los FHQT correspondientes, hasta el nivel de las hojas (primer filtro por similitud).
3. Recorrer las listas de eventos contenidas en las hojas determinando cuales objetos tienen superposición temporal con la consulta (segundo filtro temporal).
4. Unir los conjuntos resultantes -para eliminar objetos duplicados- y realizar la comparación final por similitud respecto a la consulta, obteniendo así el conjunto de objetos definitivo.

En el ejemplo de la la Figura 4.7, ante la consulta $(q, 2, 25, 28)_d$ y siendo $(4, 5)$ la firma de q, se procede de la siguiente manera: en primer lugar se selecciona el intervalo $[21, 30]$ de la *Línea del Tiempo*, ya que es el único que posee superposición temporal con el intervalo de la consulta $[25, 28]$. Luego se realiza una búsqueda en el FHQT asociado, tomando en cuenta las ramas $(3, 5)$, $(4, 4)$, $(4, 7)$, $(6, 5)$ y $(6, 7)$, que son las que cumplen las restricciones de similitud y descartando las demás, obteniendo como resultado parcial el conjunto de eventos $\{(21, o_7, stay), (21, o_6, out), (30, o_7, out)\} \cup \{(23, o_{11}, in), (23, o_{11}, out), (29, o_{10}, in)\} \cup \{(30, o_{12}, in)\} \cup \{(22, o_{13}, in), (26, o_{13}, out)\} \cup \{(21, o_{14}, stay)\}$.

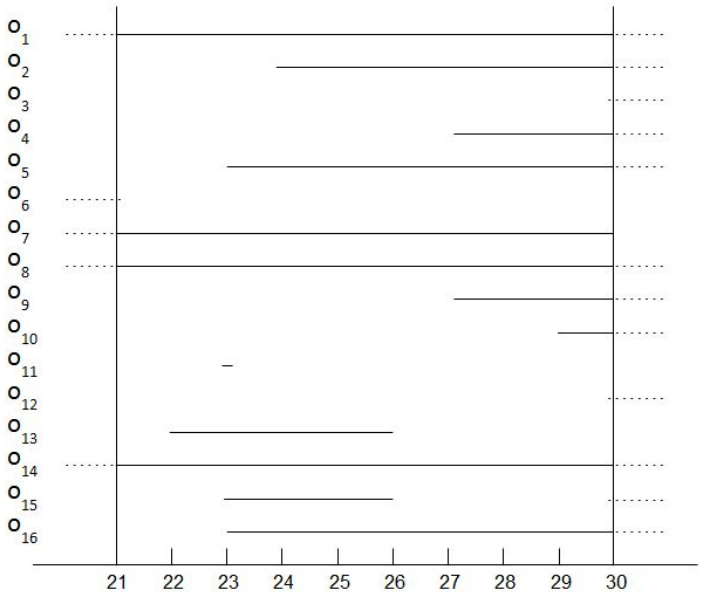

Figura 4.8: *Distribución temporal de los objetos del ejemplo*

Posteriormente se recorren los eventos para determinar cuales objetos estuvieron vigentes dentro del intervalo de consulta. En la Figura 4.8 se muestra gráficamente la vigencia temporal de los objetos contenidos en el FHQT que se está tratando. Ante un evento *in* o *stay* correspondiente a un instante anterior al tiempo final de la consulta, el objeto asociado se ingresa al conjunto de resultados. Si el evento es *out* y el instante es menor al tiempo inicial de la consulta, el objeto se da de baja del conjunto.

En el ejemplo, ya que el intervalo de consulta es $[25, 28]$, los objetos o_7 (permanece desde el intervalo anterior hasta el instante 30), o_{13} (ingresa en 22 y permanece hasta 26 inclusive) y o_{14} (permanece durante todo el intervalo) forman parte del conjunto de candidatos y los demás se descartan. Seguidamente, si el intervalo temporal de la consulta abarcase más de un intervalo de la *Línea del Tiempo*, se realizaría la unión entre este conjunto y los conjuntos correspondientes a los demás intervalos para evitar evaluar la función de distancia más de una vez con un mismo objeto. En este caso no es necesario porque el intervalo de consulta está incluido en un único intervalo.

Por último, se calcula la distancia entre cada objeto candidato y el objeto consulta y se descartan aquellos con distancia mayor al radio de búsqueda, obteniéndose el resultado final.

Este diseño produce que se filtren en primer lugar intervalos -y por lo tanto FHQTs- completos y luego, dentro de cada intervalo, subárboles completos. En la Figura 4.9 se muestra el pseudocódigo correspondiente al algoritmo de consulta del Event-FHQT. El algoritmo se compone de una rutina principal y una subrutina auxiliar que realiza las consultas a los FHQT.

El costo espacial del *Event-FHQT* está determinado por el tamaño de cada FHQT multiplicado por la cantidad de intervalos de la *Línea del Tiempo*. En esta versión del índice, la cantidad de pivotes es la misma para todos los FHQT, por lo tanto dichos árboles poseen estructuralmente el mismo tamño, y sólo se diferencian en el espacio ocupado por las listas de

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

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

Figura 4.9: *Event-FHQT: pseudocódigo de consulta por rango de similitud y tiempo válido*

eventos incluidas en sus hojas. Por otro lado, en lugar de considerar únicamente la cantidad de objetos, en el cálculo se debe tener en cuenta la cantidad de eventos asociados a cada uno de los mismos. Sea n la cantidad de intervalos, k la cantidad de pivotes, c la cantidad de objetos y e la cantidad promedio de eventos por objeto, y considerando que $k = log(ce)$, el costo espacial del *Event-FHQT* es $nkce$ punteros.

4.4.3.3. Construción del índice

Para la construcción del índice en primer lugar se seleccionan los pivotes para los FHQTs en forma aleatoria o mediante algún método de selección de pivotes. La cantidad de pivotes usualmente se calcula como el logaritmo de la cantidad de elementos de la base de datos.

Luego para cada evento a registrar, se ejecutan los siguientes pasos:

1. Encontrar el intervalo que contiene el instante correspondiente al evento. Si el intervalo no existe, agregar tantos intervalos como sea necesario hasta alcanzar el intervalo busca-

do. Al crear un nuevo intervalo se deben agregar en él, eventos *stay* para todos los objetos activos en el último instante del intervalo anterior.

2. Insertar el evento en el FHQT correspondiente a dicho intervalo, utilizando el algoritmo de inserción de un objeto en un FHQT.

3. Actualizar los intervalos relacionados:

   - Si el evento es de tipo *in*, agregar en los FHQTs de los intervalos posteriores un evento *stay* con instante igual al primer instante del intervalo. Este proceso se realiza hasta que no haya más intervalos o hasta que se encuentre otro evento ya registrado para el mismo objeto.
   - Si el evento es de tipo *out*, agregar en los FHQTs de los intervalos anteriores un evento *stay* con instante igual al primer instante del intervalo. Este proceso se realiza hasta alcanzar el inicio o encontrar otro evento asociado al mismo objeto.

Note que si el índice se actualiza en tiempo real, las actualizaciones de los intervalos posteriores para los eventos *in* y las anteriores para los eventos *out*, no son necesarias. Lo mismo sucede si el tiempo que se registra es el tiempo transaccional en lugar del tiempo válido.

4.4.3.4. Event-FHQT$^+$

El *Event-FHQT*$^+$ es una generalización del *Event-FHQT* que permite trabajar tanto con funciones de distancia continuas como discretas. Se utiliza la misma estrategia que para el *FHQT*$^+$*-Temporal* en el cual, como se mostró anteriormente, se asocian a cada rama intervalos de valores de distancias en lugar de un único valor natural. Para ello se define una cantidad fija de rangos de distinta amplitud en base al histograma de distancias, de tal manera de tener una alta probabilidad de que los FHQTs queden balanceados. Las consultas se resuelven de manera similar al *Event-FHQT*, teniendo en cuenta que las búsquedas por similitud en los FHQTs se realizan utilizando los rangos actuales de las ramas, más/menos el radio de búsqueda.

En el capítulo siguiente se muestra la evaluación experimental de los índices métrico-temporales *FHQT*$^+$*-Temporal* y *Event-FHQT*$^+$ para verificar su funcionamiento y determinar su eficiencia en comparación con la solución trivial.

Capítulo 5

Evaluación Experimental

En este capítulo se presentan resultados experimentales que muestran la eficiencia comparada de los índices *FHQT*$^+$*-Temporal* y *Event-FHQT*$^+$ entre sí y versus la solución trivial y se realiza el análisis de los mismos. También se describe la metodología empleada para llevar a cabo los experimentos y se incluyen gráficos estadísticos para una mejor comprensión de los resultados. Como parte de este trabajo se codificaron todos los programas correspondientes a los índices diseñados y se adaptaron bases de datos comúnmente utilizadas por la comunidad de espacios métricos, al modelo métrico-temporal. Los experimentos aquí presentados se realizaron en una computadora con microprocesador Intel SU4100 de 1.3 GHz, con 4 GB de RAM y disco de 320 GB.

5.1. Metodología empleada

5.1.1. Bases de datos utilizadas

Ya que el modelo métrico-temporal es relativamente reciente, no existen aún bases de datos disponibles para realizar experimentos, por lo cual se optó por adaptar dos bases frecuentemente utilizadas por investigadores del área de espacios métricos: $NASA$ y $COLORS$ [FNC07], que se pueden descargar del sitio *http://www.sisap.org/library/dbs/vectors/*.

La base de datos $NASA$ es un conjunto de 40.150 vectores 20-dimensionales de números reales, que representan características de imágenes obtenidas por la $NASA$. $COLORS$ es un conjunto de 112.544 histogramas de colores representados a través de vectores 112-dimensionales obtenidos a partir de una base de datos de imágenes. En ambos casos lo usual es utilizar la distancia euclidiana como medida de similitud.

Partiendo de estos dos conjuntos de datos se generaron las bases de datos métrico-temporales *NASA*^{MT} y *COLORS*^{MT} de la siguiente manera:

- a cada vector se le asignó un identificador y un intervalo de vigencia que indica el período de validez del objeto. El intervalo total considerado fue [1, 1000].
- mediante un proceso aleatorio se generaron lotes de 1.000, 5.000, 10.000, 20.000 y 30.000 elementos.

Como métrica de similitud se utilizó distancia euclidiana, ya que como se mencionó anteriormente, es la medida utilizada usualmente sobre estas bases de datos para realizar pruebas por similitud.

5.1.2. Lotes de Consultas

Se seleccionaron dos grupos de 100 objetos cada uno, extrayendo aleatoriamente los elementos de la base de datos $NASA$ para el primer caso y $COLORS$ para el segundo. A partir de estos grupos, se generaron lotes de consultas métrico-temporales mediante la asignación en forma aleatoria de intervalos/instantes de tiempo. Uno de los lotes para cada base de datos se compuso solamente de consultas instantáneas y los demás fueron construídos asociándoles intervalos correspondientes al 10 %, 25 % y 50 % del intervalo total ($[1, 1000]$). Como resultado se obtuvieron cuatro lotes de consultas de tipo *range/rank/-* y *range/point/-* para cada base de datos.

Para completar los parámetros requeridos en las consultas, se definieron tres radios de búsquedas distintos para cada base de datos, que devuelven en promedio aproximádamente el 1 %, 5 % y 10 % de los objetos contenidos ante las consultas por similitud de los lotes definidos anteriormente. Estos radios fueron calculados experimentalmente y son los siguientes:

- Radios para $NASA$:
  - $r_1 = 0,453$.
  - $r_2 = 0,69855$.
  - $r_3 = 0,8275$.
- Radios para $COLORS$:
  - $r_1 = 0,445$.
  - $r_2 = 3,65$.
  - $r_3 = 5,38$.

Como medida de costo para determinar la eficiencia de los índices, se utilizó la cantidad de evaluaciones de la función de distancia, ya que se asume que los índices se mantienen íntegramente en memoria principal.

5.1.3. Configuración de los índices

Tanto para el *FHQT$^+$-Temporal* como para el *Event-FHQT$^+$*, se utilizó el mismo conjunto de pivotes, elegidos al azar [BNC01] para cada base de datos. La cantidad de pivotes se calculó como $\lceil log_2(|X|) \rceil$, donde $|X|$ es la cantidad de elementos de la base de datos X. En este caso se tomó como tamaño, la máxima cantidad considerada: $|X| = 30.000$.

En ambos casos también, se definió como aridad de los FHQT el valor 10 y se calcularon los intervalos de distancias asociados a cada hijo tomando una muestra de la base de datos y calculando los deciles estadísticos de tal manera de que los árboles tiendan a estar balanceados. El tamaño de cada intervalo de la *Línea del Tiempo* del *Event-FHQT$^+$* se fijó en 50, es decir que para el intervalo total de $[1, 1000]$ fueron necesarios 20 elementos.

Todos los cálculos de costo se obtuvieron promediando los resultados de las 100 consultas de cada lote de prueba. Los índices se construyeron y consultaron completamente en memoria principal.

5.2. FHQT$^+$-Temporal: análisis de los resultados obtenidos

En esta sección se presentan, grafican y analizan los resultados obtenidos para el *FHQT$^+$-Temporal* en comparación con la primer solución trivial planteada en la sección 4.3, utilizando un FHQT como índice métrico. Es de notar que cualquiera de dichas soluciones tiene, como mínimo, el costo correspondiente a un índice métrico.

Por cuestiones de simplicidad en esta Sección sólo se muestran las gráficas que se consideran más representativas de los resultados y las demás se incluyen en el apéndice.

5.2.1. Base de datos NASA^{MT}

Variación del costo en función del tamaño de la base de datos

En los gráficos de la Figura 5.1 se muestran las curvas de costos correspondientes a los lotes de consultas instantáneas y 50 % del intervalo respectivamente, en comparación con la solución trivial. El eje x indica la cantidad de elementos de la base de datos NASA^{MT} y el eje y, el promedio de evaluaciones de la función de distancia.

Las curvas muestran que ante el aumento de la cantidad de elementos del conjunto de datos consultado, el costo se incrementa de forma similar a una curva lineal (tener en cuenta que los valores del eje x no están en escala), aunque los factores por los cuales se multiplica son muy distintos. Claramente se ve que el *FHQT$^+$-Temporal* supera ampliamente la performance de la solución trivial. En el mejor de los casos –cuando el tamaño es el mayor y las consultas son instantáneas–, su costo es de sólo el 7,89 % del correspondiente a la solución trivial, y en el

peor de los casos, un 12,6 %. Es importante notar que el costo de la solución trivial no varía en función del tamaño del intervalo de tiempo, por lo cual en ambos gráficos la curva es la misma.

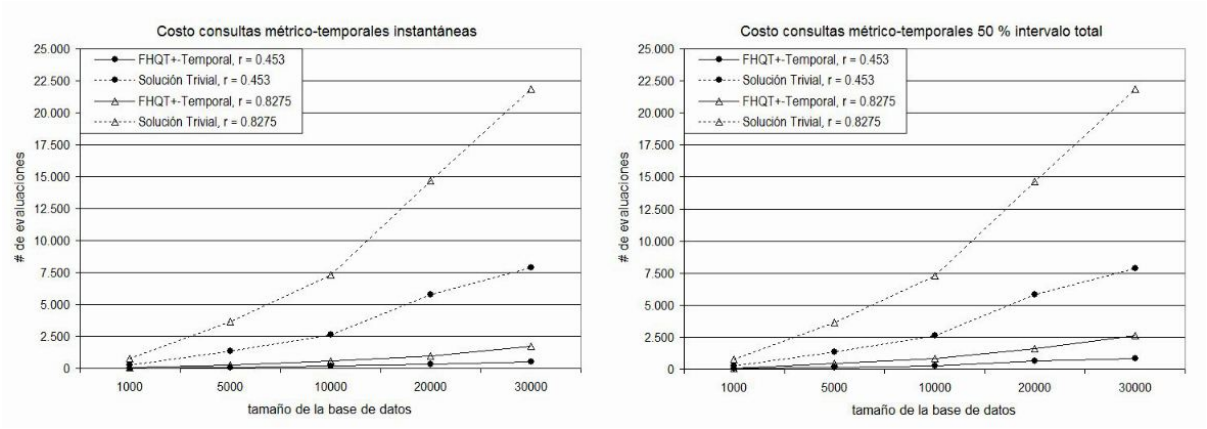

Figura 5.1: *Costo consultas métrico-temporales mediante un FHQT$^+$-Temporal a la base de datos NASA^{MT}, en función del tamaño*

El *porcentaje de aciertos*, medido como la cantidad de objetos resultantes sobre el costo promedio de las consultas, fue del $0{,}1$ % al $1{,}0$ % para la solución trivial, es decir que por cada elemento resultante tuvieron que ser evaluados 1000 objetos en el peor de los casos y 100 en el mejor caso. Para el *FHQT$^+$-Temporal* este porcentaje se eleva a $0{,}7$ % y $9{,}4$ % respectivamente, correspondientes a 143 y 11 evaluaciones de la función de distancia por cada objeto resultante. En ambos casos, las mejoras no son producidas por la variación de la cantidad de elementos, sino por la modificación del radio e intervalo de búsqueda.

El *FHQT$^+$-Temporal* supera claramente en eficiencia a la solución trivial en todos los casos, por lo cual en los apartados siguientes sólo se analizan las variaciones de este índice respecto a los distintos parámetros.

Variación del costo en función del radio de búsqueda

En la Figura 5.2 se presentan dos gráficas de costo del *FHQT$^+$-Temporal* en función del radio de búsqueda. La primera corresponde a consultas instantáneas y la segunda a intervalos de tiempo con tamaño promedio igual al 50 % del total.

Como es lógico, la cantidad de evaluaciones de la función de distancia se incrementa considerablemente al aumentar el radio de búsqueda. Sin embargo, el porcentaje de aciertos mejora sustancialmente. Este porcentaje varía entre $0{,}7$ y $0{,}9$ para el radio menor, y es alrededor de 9 veces más grande para el mayor radio. Esto significa que la eficiencia del *FHQT$^+$-Temporal* aumenta cuando se incrementa el radio de búsqueda. En todo índice métrico, es natural que el porcentaje de aciertos en algún momento aumente al consultar con mayores radios debido a que la cantidad de resultados también es mayor. En el caso extremo, cuando se consulta con un radio que incluye a todos los elementos de la base de datos, este porcentaje es cercano a 100. Sin embargo, en una base métrico-temporal, si sólo se aumenta el radio y el intervalo de tiempo

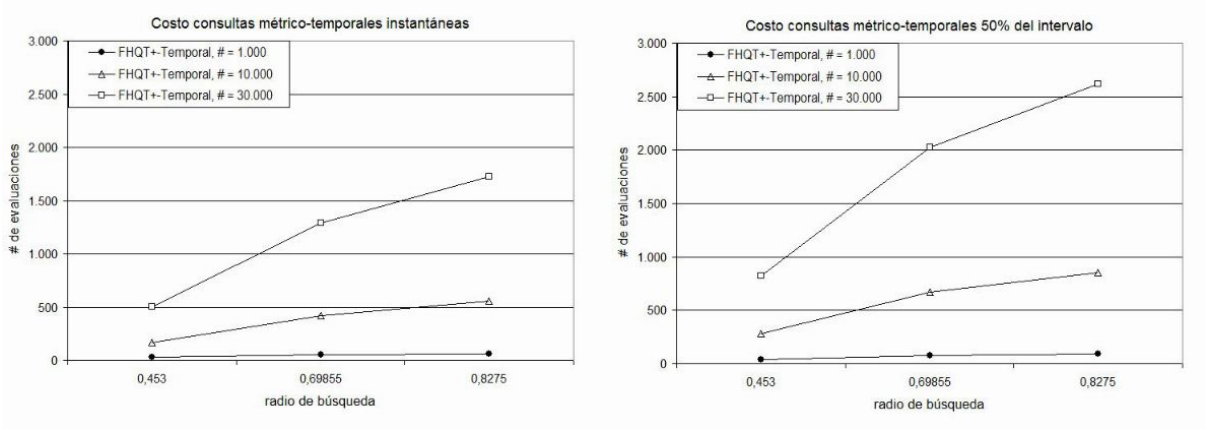

Figura 5.2: *Costo consultas métrico-temporales mediante un FHQT$^+$-Temporal a la base de datos NASA^{MT}, en función del radio*

de la consulta permanece constante, puede ser que la cantidad de resultados sea la misma ya que no todos los elementos cumplirán la restricción temporal.

En el *FHQT$^+$-Temporal*, cuando se aumenta el radio de búsqueda las restricciones temporales se hacen más importantes para el proceso de descarte de elementos, es decir que la cantidad de elementos que cumplen con la condición de similitud es mayor, pero la cantidad de elementos que cumplen con la restricción temporal se mantiene igual. Esta es una de las causas del aumento del porcentaje de aciertos.

Variación del costo en función de la amplitud del intervalo de tiempo

Para evaluar la influencia de las variaciones de la amplitud del intervalo temporal sobre el costo de las consultas se presentan los gráficos de la Figura 5.3. En el eje X se ubican en primer lugar las consultas instantáneas y a continuación los intervalos temporales correspondientes al $10\,\%$, $25\,\%$ y $50\,\%$ del intervalo total. Los valores del eje Y representan las cantidades promedio de evaluaciones de la función de distancia para los lotes de 100 consultas. El radio de consulta correspondiente al primer gráfico es $0{,}453$ y el del segundo $0{,}8275$.

Como se ve en ambos gráficos, el costo de las consultas que toman el $50\,\%$ del intervalo total es entre un $50\,\%$ y un $70\,\%$ mayor que el de las consultas instantáneas. Por otro lado, la cantidad de elementos resultantes es alrededor de dos veces mayor (lo que es coherente con el aumento del intervalo temporal, ya que los objetos se encuentran uniformemente distribuidos en cuanto al tiempo). Por esta razón el porcentaje de aciertos aumenta también, entre un $30\,\%$ y un $40\,\%$.

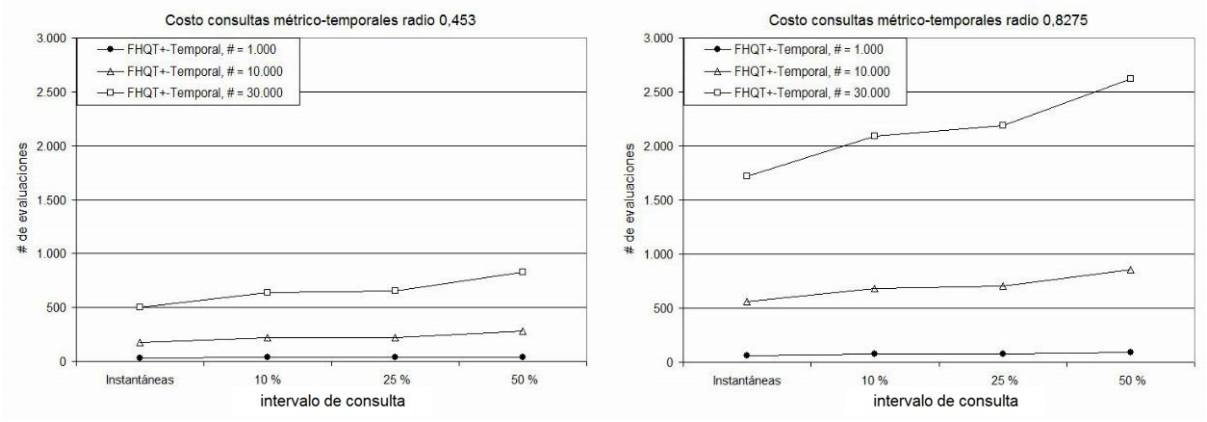

Figura 5.3: *Costo consultas métrico-temporales mediante un FHQT$^+$-Temporal a la base de datos NASA^{MT}, en función del intervalo temporal*

5.2.2. Base de datos COLORS^{MT}

Variación del costo en función del tamaño de la base de datos

Los gráficos de la Figura 5.4 corresponden a las curvas de costos del *FHQT$^+$-Temporal* ante consultas instantáneas y del 50 % del intervalo en comparación con la solución trivial. En este caso, el eje x expresa la cantidad de elementos de la base de datos COLORS^{MT}.

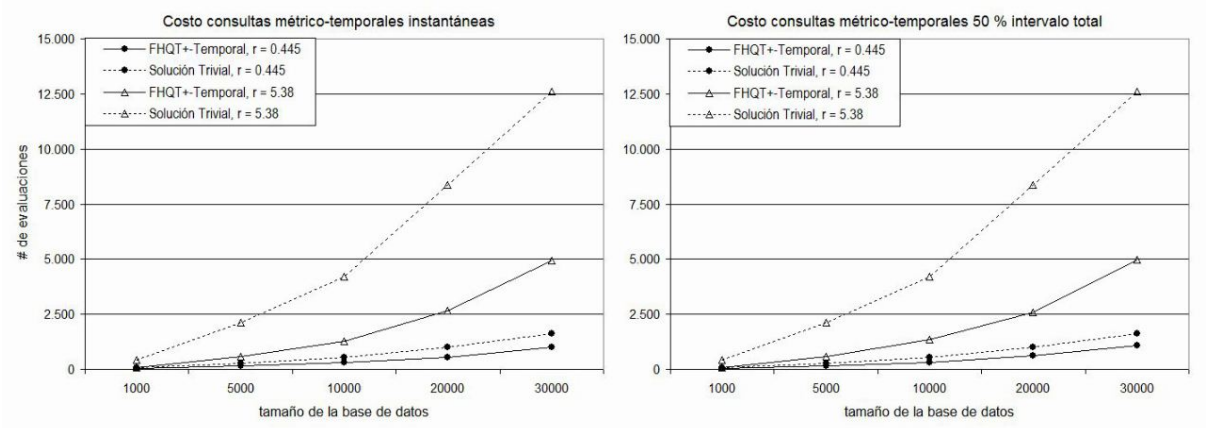

Figura 5.4: *Costo consultas métrico-temporales mediante un FHQT$^+$-Temporal a la base de datos COLORS^{MT}, en función del tamaño*

Las curvas indican el aumento del costo en proporción similar al aumento de la cantidad de elementos sobre los cuales se realizan las consultas, tal como era de esperar. Sin embargo, mientras que la solución trivial mejora su desempeño sobre la base de datos COLORS^{MT}, no sucede lo mismo con el *FHQT$^+$-Temporal*, que si bien sigue teniendo mejor rendimiento que la solución trivial, las diferencias son menores. En este caso, el *FHQT$^+$-Temporal* tiene un costo relativo del 40 % al 60 % del costo de dicha solución.

El porcentaje de aciertos aumenta también, alcanzando un valor de $4{,}1$ para la solución tri-

vial (1 resultado correcto por cada 24 elementos evaluados) y de 13,3 para el *FHQT$^+$-Temporal* (1 por cada 8).

Variación del costo en función del radio de búsqueda

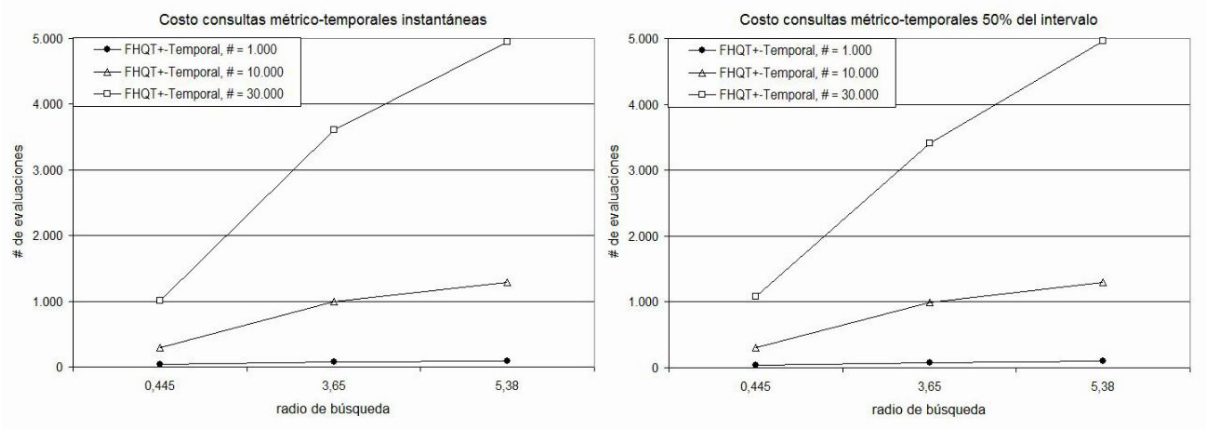

Figura 5.5: *Costo consultas métrico-temporales mediante un FHQT$^+$-Temporal a la base de datos COLORS^{MT}, en función del radio*

En la Figura 5.5 se presentan las gráficas correspondientes a la evolución del costo del *FHQT$^+$-Temporal* en función del radio, para consultas instantáneas y 50 % del intervalo temporal total.

El incremento del costo del *FHQT$^+$-Temporal* cuando aumenta el radio, es más pronunciado en la base de datos COLORS^{MT} (4,9 veces), que en la base NASA^{MT} (3,44 veces). El porcentaje de aciertos también aumenta, pero en menor medida, alcanzando mejoras de hasta 2,5 veces.

Variación del costo en función de la amplitud del intervalo de tiempo

En la Figura 5.6 se muestra la influencia de las variaciones de la amplitud del intervalo temporal sobre el costo de las consultas. El eje x contiene los valores correspondientes a las consultas instantáneas y los intervalos temporales 10 %, 25 % y 50 % del intervalo total. Los radios de consulta fueron 0,445 y 5,38 para el primer y segundo gráfico respectivamente.

Las curvas muestran que no existen variaciones importantes en el costo de las consultas cuando se aumenta el intervalo temporal de las consultas. Se concluye que, para el FHQT$^+$-Temporal, la variable costo posee una menor dependencia de la amplitud del intervalo de tiempo consultado.

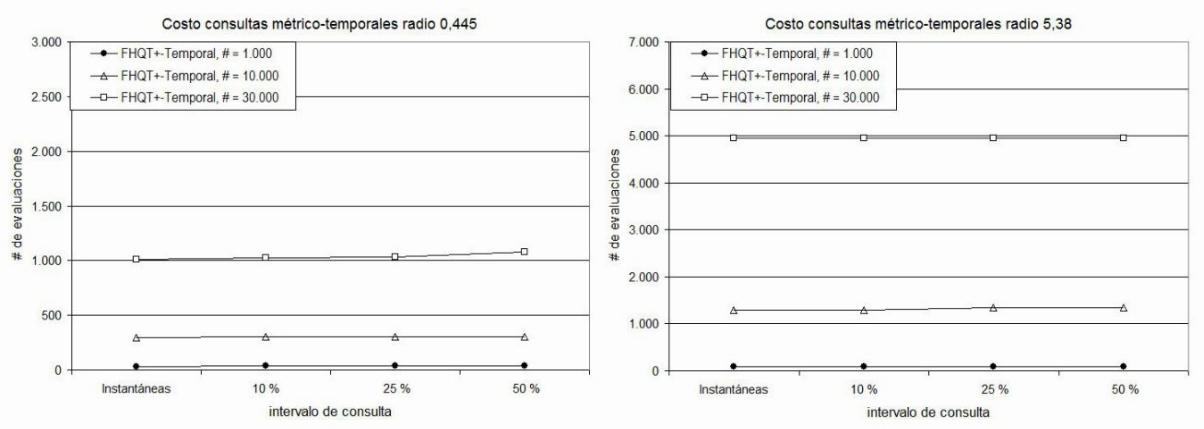

Figura 5.6: *Costo consultas métrico-temporales mediante un FHQT$^+$-Temporal a la base de datos COLORS^{MT}, en función del intervalo temporal*

5.3. Event-FHQT$^+$: análisis de los resultados obtenidos

De igual modo que en la Sección anterior, se presentan los resultados de la evaluación experimental, obtenidos promediando los costos para los distintos lotes de 100 consultas; en este caso para el índice métrico-temporal *Event-FHQT*$^+$. Inicialmente se lo compara con la solución trivial y luego se evalúa su comportamiento bajo distintas situaciones.

Sólo se muestran las gráficas que se consideran más representativas y las restantes pueden consultarse en el apéndice.

5.3.1. Base de datos NASA^{MT}

Variación del costo en función del tamaño de la base de datos

En la Figura 5.7 se muestran las curvas de costos correspondientes a los lotes de consultas instantáneas y 50 % del intervalo en comparación con la solución trivial, teniendo como eje x la cantidad de elementos de la base de datos NASA^{MT}.

Mientras que el costo de la solución trivial aumenta 30 veces al pasar de 1.000 a 30.000 elementos, el crecimiento del costo del *Event-FHQT*$^+$ es de entre 14 a 20 veces. La cantidad de evaluaciones de la función de distancia que el *Event-FHQT*$^+$ necesita para resolver las consultas, es sólo el 5 % - 9 % de la requerida por la solución trivial, por lo cual se verifica que la eficiencia de este índice es muy superior a las soluciones planteadas en la Sección 4.3.

Respecto al porcentaje de aciertos, el *Event-FHQT*$^+$ obtiene un promedio de $4{,}2$ % para la base de datos NASA^{MT}, con un valor mínimo de $0{,}7$ % (142 evaluaciones por cada resultado correcto) y un máximo $11{,}8$ % (8 evaluaciones por cada objeto resultante). El mínimo se obtiene cuando el radio de consulta es menor, mientras que el máximo se alcanza para los mayores valores del radio y del tamaño de la base de datos.

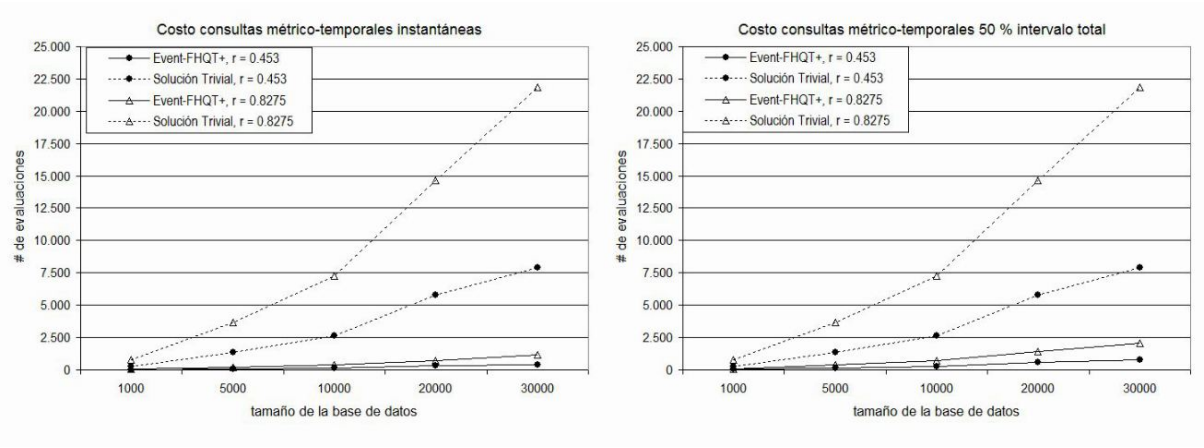

Figura 5.7: *Costo consultas métrico-temporales mediante un Event-FHQT$^+$ a la base de datos NASA^{MT}, en función del tamaño*

Variación del costo en función del radio de búsqueda

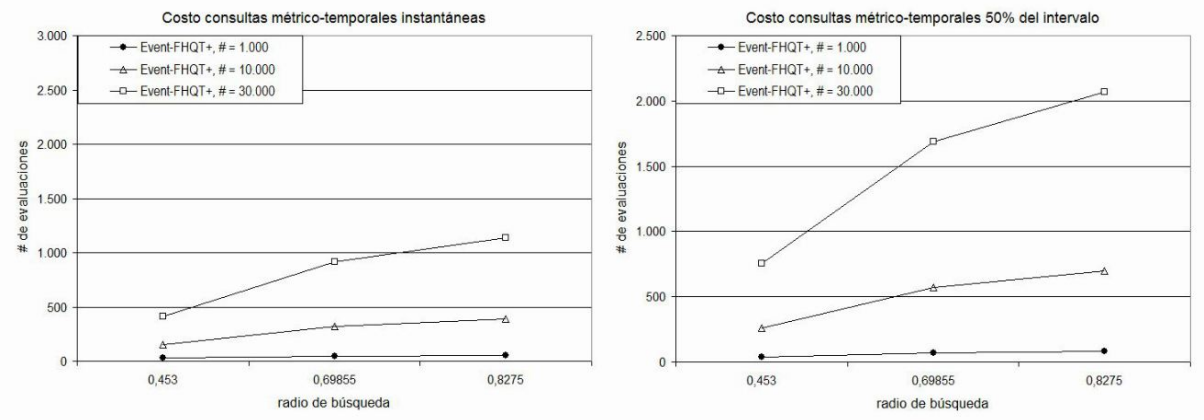

Figura 5.8: *Costo consultas métrico-temporales mediante un Event-FHQT$^+$ a la base de datos NASA^{MT}, en función del radio*

La Figura 5.8 muestra dos gráficos de costos, con variaciones de la amplitud del radio de consulta. El primero mediante consultas instantáneas y el segundo correspondiente a intervalos de tiempo del 50 % del total.

La cantidad de evaluaciones de la función de distancia se incrementa alrededor de $2{,}7$ veces en el peor de los casos, mientras que la cantidad de elementos resultantes aumenta más de 29 veces. Esto produce que el porcentaje de aciertos sea 9 veces superior para el mayor radio. Tal como sucede con el *FHQT$^+$-Temporal*, en el *Event-FHQT$^+$* también se hace más importante la restricción temporal durante el proceso de descarte de elementos a medida que se aumenta el radio de búsqueda.

Variación del costo en función de la amplitud del intervalo de tiempo

El *Event-FHQT*$^+$ mantiene su buena performance ante variaciones del intervalo de tiempo consultado. Su costo aumenta a menos del doble en el peor de los casos, tal como se ve en la Figura 5.9, y la cantidad de elementos devueltos entre las consultas instantáneas y la correspondiente al 50 % del intervalo es dos veces mayor.

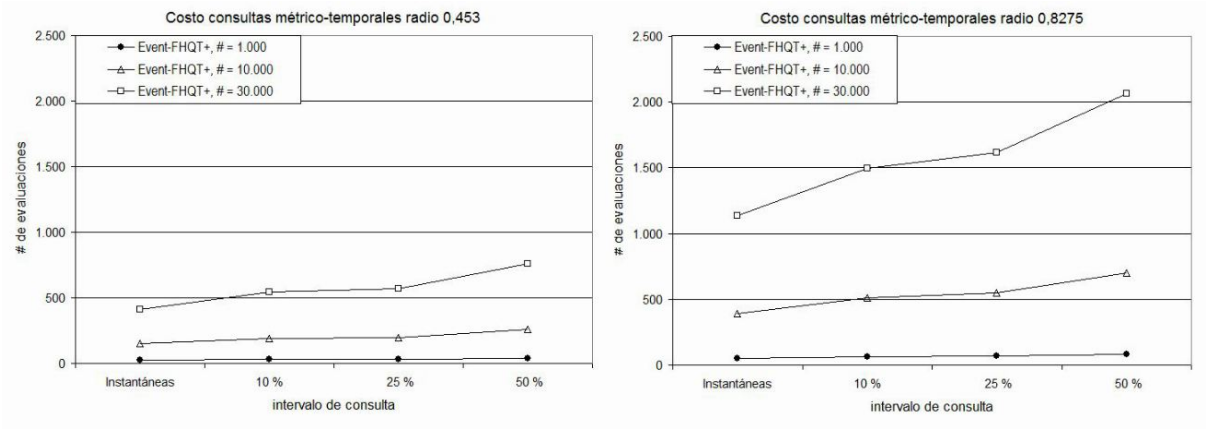

Figura 5.9: *Costo consultas métrico-temporales mediante un Event-FHQT*$^+$ *a la base de datos NASA*^{MT}*, en función del intervalo temporal*

5.3.2. Base de datos COLORS^{MT}

Variación del costo en función del tamaño de la base de datos

En los gráficos de la Figura 5.10 se muestran las curvas de costos correspondientes a los lotes de consultas instantáneas y 50 % del intervalo respectivamente, en comparación con la solución trivial, teniendo como eje x la cantidad de elementos de la base de datos COLORS^{MT}.

Mientras que la solución trivial tiene un crecimiento importante del costo de las consultas ante el aumento de la cantidad de elementos de la base de datos, el costo del *Event-FHQT*$^+$ crece lentamente. Este índice es significativamente superior en eficiencia a la solución trivial para la base de datos COLORS^{MT} también, requiriendo en promedio sólo un 8 % del costo de esta última.

Variación del costo en función del radio de búsqueda

Para visualizar la evolución del costo del *Event-FHQT*$^+$ sobre COLORS^{MT} cuando se varía el radio, se presenta la Figura 5.11.

La cantidad de evaluaciones de la función de distancia aumenta de 2 a 6 veces al pasar del radio menor (correspondiente al 1 % de resultados de la base de datos) al mayor (correspon-

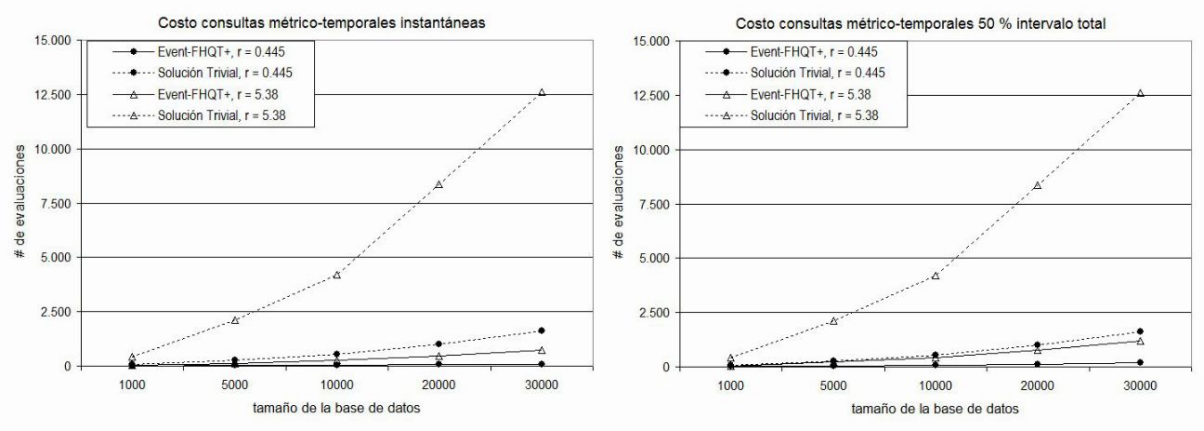

Figura 5.10: *Costo consultas métrico-temporales mediante un Event-FHQT$^+$ a la base de datos COLORS^{MT}, en función del tamaño*

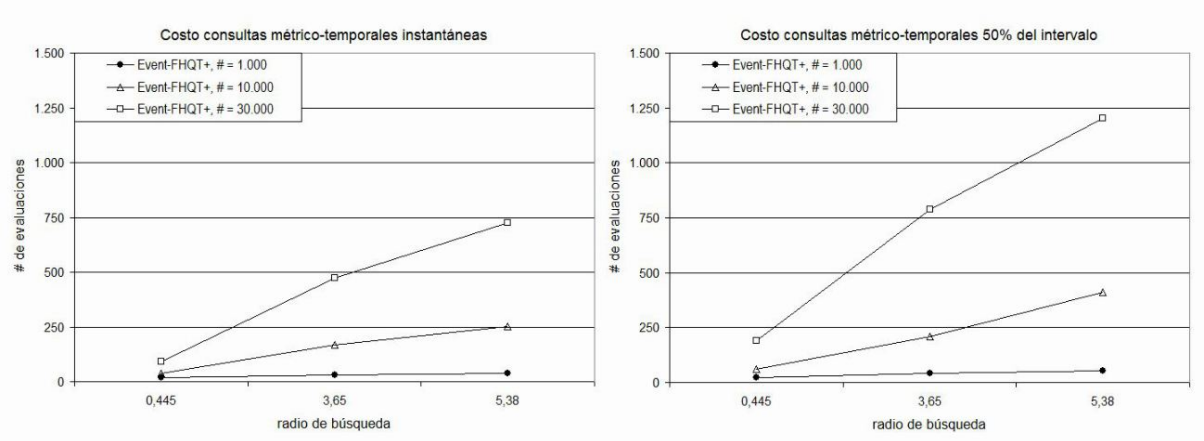

Figura 5.11: *Costo consultas métrico-temporales mediante un Event-FHQT$^+$ a la base de datos COLORS^{MT}, en función del radio*

diente al 10 %). Su porcentaje de aciertos, en la mayoría de los casos se mantiene prácticamente igual, alrededor de un 30 %, con leves aumentos y disminuciones.

Variación del costo en función de la amplitud del intervalo de tiempo

Por último, se presenta la Figura 5.12 que muestra la variaciones del costo frente al cambio de la amplitud del intervalo.

Notablemente, el *Event-FHQT$^+$* es muy estable ante el aumento de la amplitud del intervalo consultado. Al pasar de consultas instantáneas al 50 % del intervalo total, obtiene un incremento máximo que apenas duplica su costo.

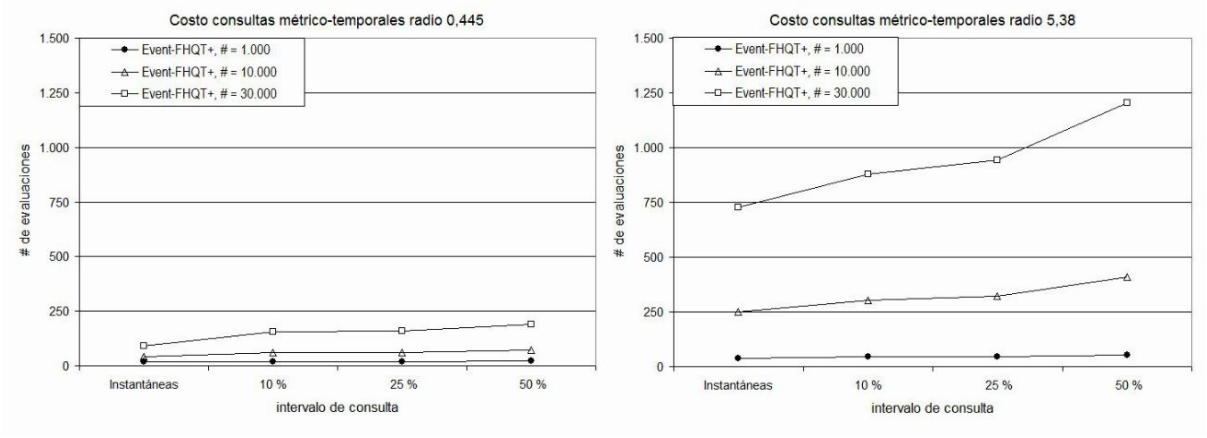

Figura 5.12: *Costo consultas métrico-temporales mediante un Event-FHQT$^+$ a la base de datos COLORS^{MT}, en función del intervalo temporal*

5.4. FHQT$^+$-Temporal versus Event-FHQT$^+$

En esta sección se realiza una comparación del rendimiento de ambos índices ante variaciones del tamaño, radio de consulta y amplitud del intervalo y se exponen las conclusiones sobre los experimentos realizados.

5.4.1. Comparación del costo en función de la cantidad de elementos

Inicialmente, se presenta las Figuras 5.13 y 5.14, las cuales contienen gráficos que comparan los costos correspondientes al *FHQT$^+$-Temporal* y al *Event-FHQT$^+$* ante variaciones de la cantidad de elementos de las bases de datos NASA^{MT} en la primera figura y COLORS^{MT} en la segunda.

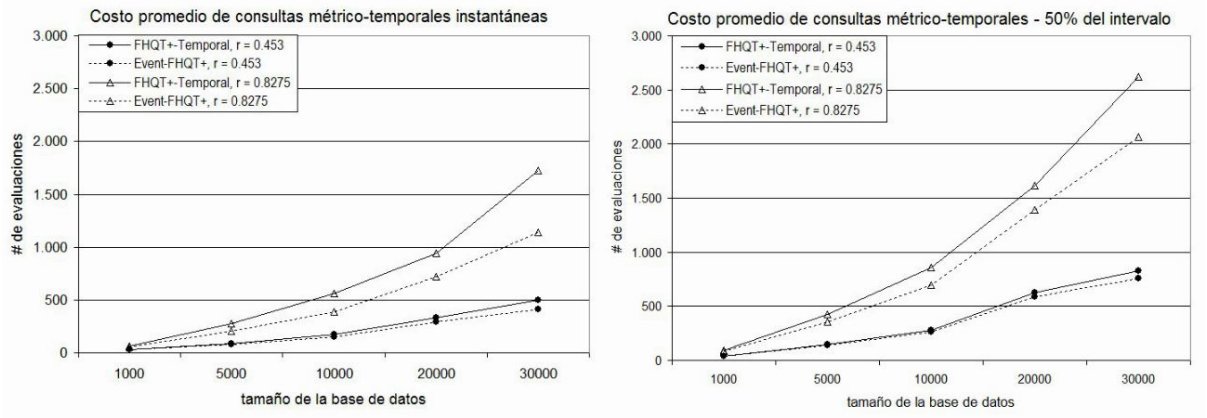

Figura 5.13: *Costo comparado, FHQT$^+$-Temporal vs Event-FHQT$^+$, base de datos NASA^{MT}, en función del tamaño*

En los gráficos se puede apreciar que en todos los casos el *Event-FHQT$^+$* supera en eficien-

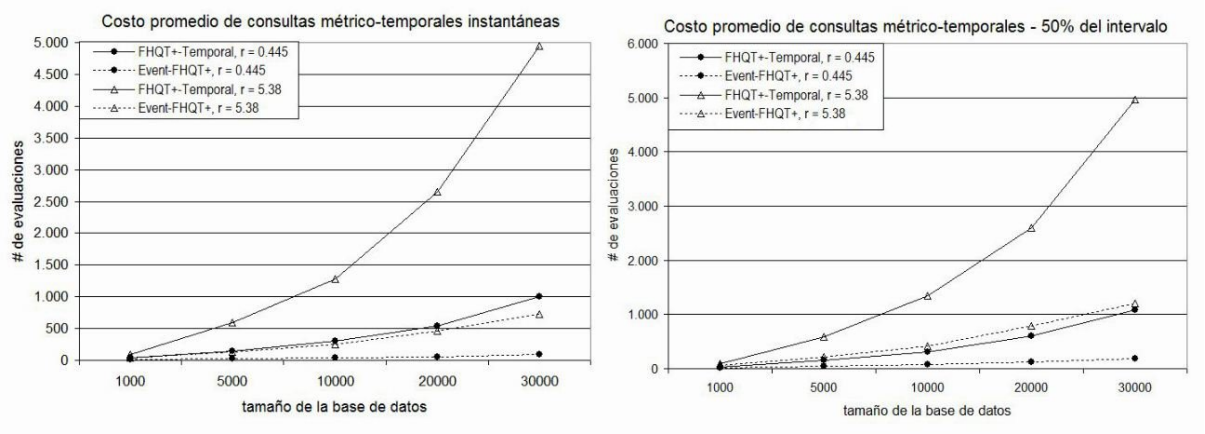

Figura 5.14: *Costo comparado, FHQT$^+$-Temporal vs Event-FHQT$^+$, base de datos COLORS^{MT}, en función del tamaño*

cia al *FHQT$^+$-Temporal*. Esta diferencia se hace más evidente en las pruebas realizadas sobre la base de datos COLORS^{MT}, lo que indica que las propiedades del conjunto de datos influyen de distinta manera en ambos índices. Una de las razones de este comportamiento es que dichos conjuntos poseen dimensionalidad intrínseca diferente y esto afecta en mayor medida al *FHQT$^+$-Temporal*.

Por otro lado, ante el aumento de la cantidad de elementos el *Event-FHQT$^+$* tiene un comportamiento mucho más estable. Esto es debido al problema del agrupamiento de objetos similares muy distantes en el tiempo que posee el *FHQT$^+$-Temporal*. Es decir, a medida que la cantidad de elementos se incrementa, la probabilidad de que existan objetos similares con intervalos de tiempo alejados es mayor. Al agrupar estos objetos, los intervalos de tiempo de los nodos interiores del *FHQT$^+$-Temporal* se hacen muy grandes y el índice pierde gran parte de su capacidad de filtrado por tiempo.

Para la base de datos NASA^{MT} el *Event-FHQT$^+$* obtiene una mejora de entre un $2{,}2\,\%$ y un $34\,\%$. El mínimo corresponde a consultas con el $50\,\%$ del intervalo, el menor radio y la menor cantidad de elementos, mientras que el mayor porcentaje de mejora se obtiene ante consultas instantáneas y con el radio y el tamaño con sus valores máximos.

En el caso de COLORS^{MT}, los porcentajes de mejora son mucho mayores; $43{,}4\,\%$ el mínimo y $91\,\%$ el máximo, correspondientes a los mismos casos planteados en el párrafo anterior.

El porcentaje promedio de mejora del *Event-FHQT$^+$* sobre el *FHQT$^+$-Temporal* es el $15{,}5\,\%$ para el conjunto NASA^{MT} y el $72{,}7\,\%$ para COLORS^{MT}.

5.4.2. Comparación del costo en función del radio de búsqueda

En las Figuras 5.15 y 5.16 se muestra la evolución de los costos de ambos índices cuando se varía el radio de búsqueda.

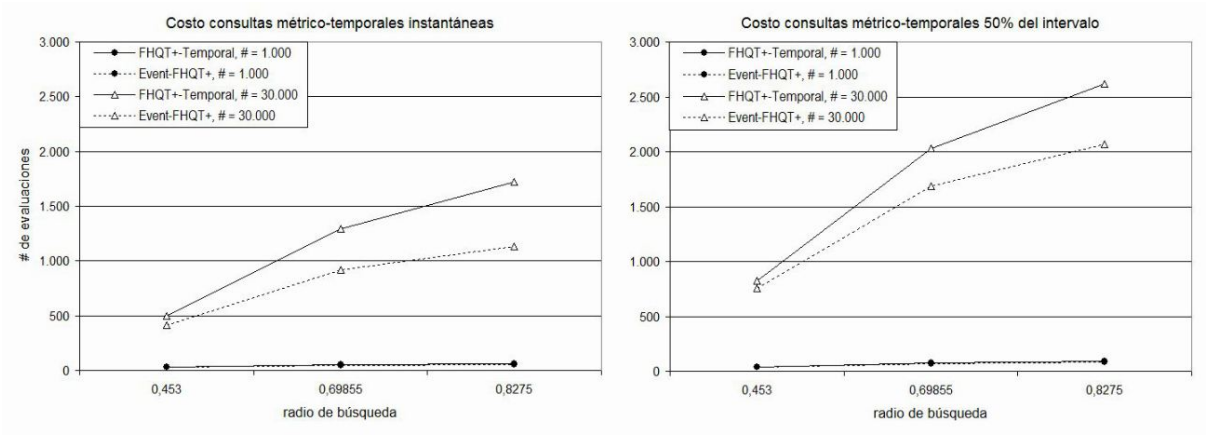

Figura 5.15: *Costo comparado, FHQT$^+$-Temporal vs Event-FHQT$^+$, base de datos NASA^{MT}, en función del radio*

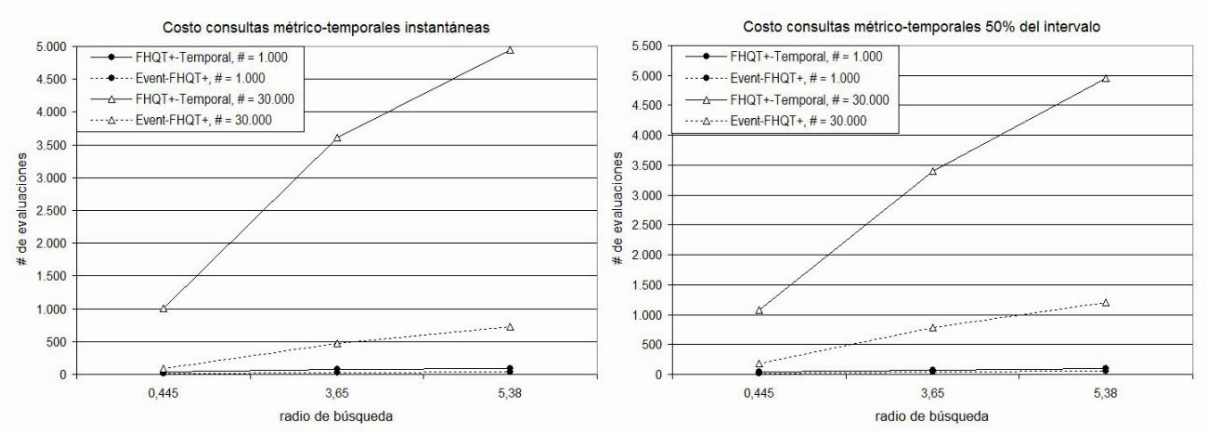

Figura 5.16: *Costo comparado, FHQT$^+$-Temporal vs Event-FHQT$^+$, base de datos COLORS^{MT}, en función del radio*

Con pequeñas cantidades de elementos en las bases de datos, no se muestran diferencias importantes en la eficiencia de los índices para ninguno de los conjuntos. Pero al considerar el mayor tamaño del conjunto NASA^{MT}, la variación del radio produce un aumento del costo del *FHQT$^+$-Temporal* de $3{,}2$ veces, mientras que el aumento del *Event-FHQT$^+$* es sólo de $2{,}7$ veces. Esta situación se invierte para el caso de COLORS^{MT}, donde el aumento del costo del primer índice es de $4{,}9$ veces, mientras que para el segundo es 8 veces más. A pesar de tener mayor pérdida de eficiencia para este caso, el *Event-FHQT$^+$* mantiene sus costos muy por debajo de los del *FHQT$^+$-Temporal*.

5.4.3. Comparación del costo en función de la amplitud del intervalo de tiempo

Para analizar el funcionamiento de ambos métodos de acceso ante variaciones del intervalo de búsqueda, se muestran las Figuras 5.17 y 5.18.

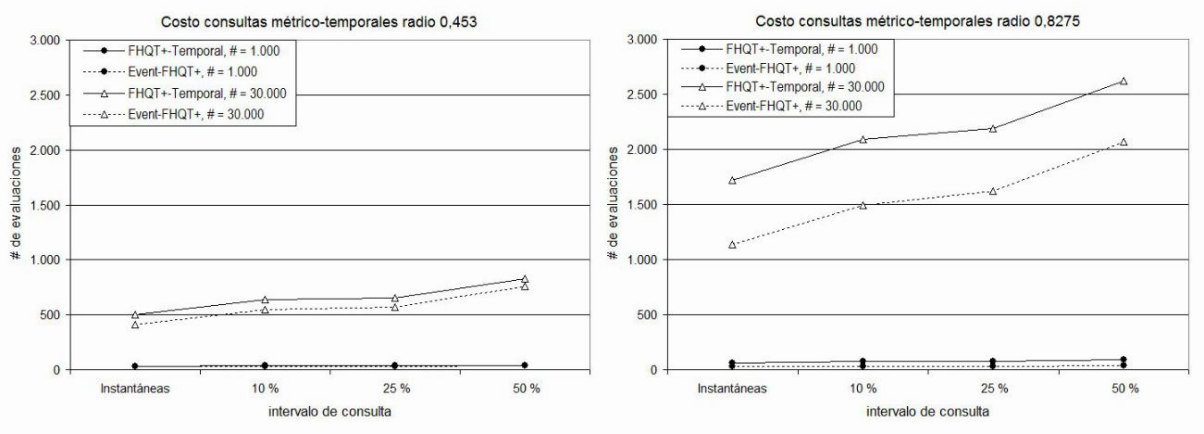

Figura 5.17: *Costo comparado, FHQT$^+$-Temporal vs Event-FHQT$^+$, base de datos NASA^{MT}, en función del intervalo temporal*

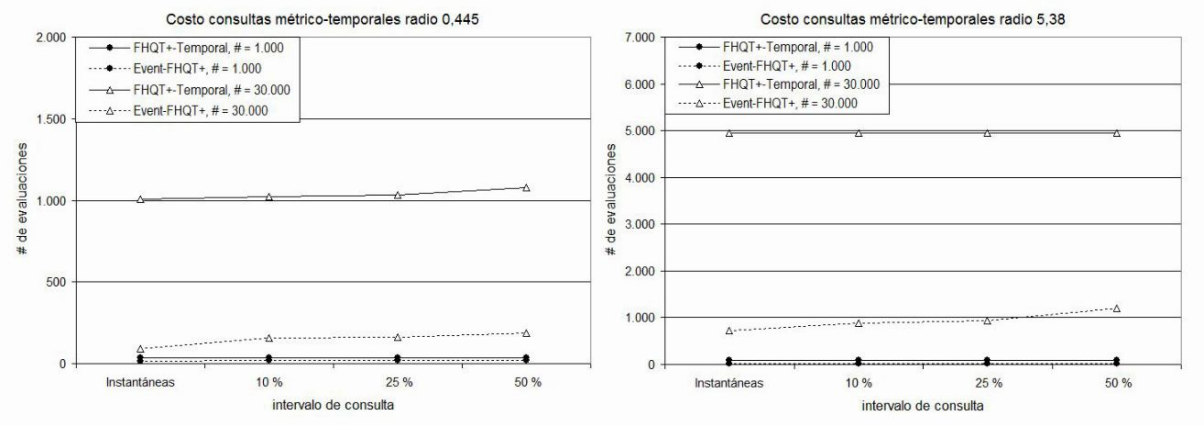

Figura 5.18: *Costo comparado, FHQT$^+$-Temporal vs Event-FHQT$^+$, base de datos COLORS^{MT}, en función del intervalo temporal*

Los dos índices tienen un comportamiento similar ante variaciones del intervalo de búsqueda. El incremento del costo es de $1{,}1$ a $1{,}8$ veces más, que no es significativo considerando que la cantidad de elementos resultantes es dos veces mayor cuando se pasa de las consultas instantáneas a consultas con el 50 % del intervalo de tiempo total.

5.4.4. Resumen de resultados

En este Capítulo se presentó la evaluación experimental de los índices métrico-temporales *FHQT$^+$-Temporal* y *Event-FHQT$^+$* sobre las bases de datos NASA^{MT} y COLORS^{MT}, para 1.000, 5.000, 10.000, 20.000 y 30.000 elementos, consultas instantáneas y con intervalos de tiempo del 10 %, 25 % y 50 % del intervalo total, y radios correspondientes al 1 %, 5 % y 10 % del conjunto de datos.

A continuación se enuncian los resultados más importantes:

- Ambos índices obtuvieron un rendimiento muy superior al de la solución trivial. El *FHQT*$^+$*-Temporal* requirió sólo el 7,89 % de la cantidad de evaluaciones de la función de distancia correspondiente a la solución trival en el mejor de los casos, y el 60 % en el peor. Para el *Event-FHQT*$^+$, estos porcentajes disminuyeron al 5 % y 9 %.

- El *FHQT*$^+$*-Temporal* es sensible al aumento de la cantidad de elementos del conjunto de datos consultado. Al pasar de 1.000 elementos a 30.000, su costo promedio se incrementó 24 veces para la base de datos NASA^{MT} y 43 veces para COLORS^{MT}.

- El *FHQT*$^+$*-Temporal* es relativamente estable ante el incremento del radio de búsqueda. Ante un aumento de 10 veces la cantidad de elementos resultantes debido a la variación del radio, este índice obtuvo un incremento de $2{,}8$ veces su costo para NASA^{MT} y $3{,}7$ veces para COLORS^{MT}.

- El *FHQT*$^+$*-Temporal* es estable ante las variaciones del intervalo de búsqueda. Con un intervalo 50 veces mayor, el costo promedio de las consultas aumentó sólo $1{,}5$ veces para NASA^{MT} y se mantuvo prácticamente igual para COLORS^{MT}.

- El *Event-FHQT*$^+$ es más estable ante variaciones del tamaño del conjunto de datos. Al pasar de 1.000 elementos a 30.000, su costo promedio se incrementó 20 veces para la base de datos NASA^{MT} y 15 veces para COLORS^{MT}.

- El *Event-FHQT*$^+$ es relativamente estable a las variaciones de radio. Cuando se aumentó 10 veces la cantidad de elementos resultantes debido a la variación del radio, este índice obtuvo un incremento de $2{,}4$ veces su costo promedio para NASA^{MT} y $4{,}8$ veces para COLORS^{MT}.

- El *Event-FHQT*$^+$ también es estable ante las variaciones del intervalo de búsqueda. Ante un intervalo 50 veces mayor, su costo promedio aumentó $1{,}7$ veces para NASA^{MT} y $1{,}6$ veces para COLORS^{MT}.

- El *Event-FHQT*$^+$ verificó ser más eficiente que el *FHQT*$^+$*-Temporal* en todas las pruebas realizadas, obteniendo hasta un 91 % de mejora sobre este último.

Se comprueba que el *Event-FHQT*$^+$ es más eficiente que el *FHQT*$^+$*-Temporal*, aunque con la desventaja de poseer mayor complejidad en su estructura interna y su funcionamiento y de poseer un costo espacial n veces mayor, donde n es la cantidad de intervalos que posee. Por otro lado, el problema de la pérdida de eficiencia (por los objetos similares distantes) que posee el *FHQT*$^+$*-Temporal*, puede tener una solución relativamente simple, por lo cual ambos métodos deberían ser tomados en cuenta como opciones para resolver eficientemente consultas métrico-termporales.

Capítulo 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 estudió el problema de aquellas consultas por similitud sobre grandes bases de datos de objetos no estructurados que involucran al menos una dimensión temporal. Para ello se definió el modelo métrico-temporal para ambas dimensiones del tiempo –válido y transaccional–, se diseñaron métodos de acceso para resolver consultas métrico-temporales eficientemente y se realizaron experimentos que verificar el comportamiento de los mismos ante variaciones de distintos parámetros.

En este capítulo se discuten los resultados de este estudio y se presentan sus limitaciones y el trabajo que aún resta hacer sobre a este tema.

6.1. Resumen

A continuación se enumeran y describen los resultados más importantes de este trabajo:

1. Definición del *Modelo Métrico-Temporal* de base de datos incluyendo ambas dimensiones de tiempo: válido y transaccional. Este modelo brinda el soporte teórico para la definición de aquellas consultas que involucran similitud y tiempo a la vez.

2. Caracterización de las situaciones reales en las cuales es conveniente aplicar este modelo.

3. Definición de los tipos de consulta métrico-temporales. Se introdujo la notación *similarity/valid/transaction* que especifica a través de su primer entrada la clase de consulta por similitud ($range$ o NN_k) y los tipos de consulta temporal ($rank$ o $point$) para tiempo válido y transaccional mediante la segunda y tercer entrada, respectivamente.

4. Diseño e implementación del método de acceso métrico-temporal *FHQT-Temporal* para distancias discretas y su variante *FHQT$^+$-Temporal* para distancias continuas.

5. Diseño e implementación del método de acceso métrico-temporal orientado a eventos *Event-FHQT* para distancias discretas y su variante *Event-FHQT*$^+$ para distancias continuas.

6. Verificación de la eficiencia de ambos índices en comparación con la solución trivial y entre sí, mediante la evaluación experimental de los mismos teniendo en cuenta distintas situaciones.

El modelo *Métrico-Temporal* (o *Espacio Métrico-Temporal*) se define a través de un universo de objetos potenciales que poseen dos intervalos de tiempo asociados; el primero corresponde a su período de validez y el restante representa los instantes de inserción y modificación o eliminación del mismo en la base de datos. Además, como parte del modelo se define una función de distancia métrica que determina el grado de disimilaridad entre los objetos del universo. Los objetos pueden ser fotos, huellas digitales, rostros, sonidos, logos, pinturas, cadenas, etc. Este modelo se utilizó luego como base para definir las consultas métrico-temporales y los métodos de acceso.

Posteriormente se definieron las características de las situaciones en las cuales es aplicable el modelo. En estas situaciones no se pueden buscar los objetos por igualdad, sino que las comparaciones deben ser por similitud; los objetos tienen instantes o intervalos de tiempo asociados que deben ser tenidos en cuenta durante la búsqueda, y la cantidad de objetos en la base de datos es suficientemente grande como para que un recorrido secuencial no sea una solución factible.

Para definir los tipos de consultas se utilizó una nueva notación especificada mediante la terna *similarity/valid/transaction*. El primer elemento se refiere al aspecto métrico de la consulta e indica si la búsqueda es por rango, del vecino más cercano o de los k vecinos más cercanos. El segundo y tercer elemento especifican si la consulta incluye los tiempos válido y transaccional, y si se está consultando un instante o un intervalo de tiempo. Por ejemplo, la terna *range/rank/-* indica el tipo de consulta por similitud por rango y considera un intervalo de tiempo válido, mientras que el tiempo transaccional es irrelevante para la misma.

El índice *FHQT-Temporal* es un $FHQT$ modificado, donde a cada nodo del árbol se le agrega el intervalo de tiempo correspondiente al mínimo período de tiempo que incluye temporalmente a todos los objetos del subárbol cuya raíz es dicho nodo. Los nodos hoja contienen además del intervalo, los objetos (o apuntadores a los mismos) y sus intervalos asociados . Ante una consulta, el *FHQT-Temporal* utiliza tanto el aspecto métrico como el temporal para descartar elementos.

El *FHQT*$^+$*-Temporal* es una variante del índice anterior que permite funciones de distancia continuas. Para ello se definen intervalos de distancias asociados a cada rama del árbol en lugar de un valor discreto. Para que la estructura tienda a estar balanceada, luego de establecer la aridad de la estructura, se toma una muestra de la base de datos y se utilizan deciles estadísticos para establecer los valores de los intervalos. Este índice supera ampliamente en eficiencia a la solución trivial planteada en la Sección 4.3. En la evaluación experimental, el costo de las consultas mediante un *FHQT*$^+$*-Temporal* fue de sólo entre el 7,89 % y el 12,6 % del correspondiente a la solución trivial para la base de datos NASA^{MT}, y entre el 40 % y el 60 % en el caso

de COLORS^{MT}.

El segundo índice, el *Event-FHQT*, está orientado a eventos y se compone por una *Línea del Tiempo* de intervalos consecutivos cada uno conteniendo un $FHQT$ con listas de eventos en sus hojas. Este índice incorpora las altas, modificaciones y bajas de los objetos a través del registro de los eventos correspondientes. El *Event-FHQT*$^+$ es una variante de este índice para distancias continuas. El *Event-FHQT*$^+$ muestra un comportamiento más estable que el *FHQT*$^+$*-Temporal* y en la evaluación experimental superó ampliamente la eficiencia de la solución trival con una reducción del costo a sólo entre el 5 % - 9 % de dicha solución para la base de datos NASA^{MT} y a un 8 % en promedio para COLORS^{MT}.

En las comparaciones entre ambos índices pudo comprobarse que el *Event-FHQT*$^+$ posee menor costo de resolución de consultas métrico-temporales que el *FHQT*$^+$*-Temporal* en todos los casos. Esta diferencia se hace más evidente en las pruebas realizadas sobre la base de datos COLORS^{MT}, por lo que se deduce que este índice es más estable ante conjuntos de datos con distintas propiedades. También se nota que el *FHQT*$^+$*-Temporal* disminuye su eficiencia a medida que crece la base de datos. El *Event-FHQT*$^+$ obtuvo mejoras sobre el *FHQT*$^+$*-Temporal* de entre un $2{,}2$ % a un 34 % para la base de datos NASA^{MT} y un $43{,}4$ % al 91 % para COLORS^{MT}. En promedio, el porcentaje de mejora fue el $15{,}5$ % para el primer conjunto y del $72{,}7$ % para el segundo.

6.2. Trabajo futuro

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 y aplicación de los mismos. A continuación se describen brevemente las principales:

- 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.
- Las consultas evaluadas fueron de tipo *range/rank/-* y *range/point/-*. Existen varios otros tipos de consultas para las cuales se requieren diseñar algoritmos y realizar experimentos. Por ejemplo, las consultas métrico-temporales que involucran los k vecinos más cercanos; las que tienen en cuenta ambas dimensiones temporales, y además, todas las consultas métricas o temporales puras.
- El *FHQT-Temporal* tiende a disminuir su eficiencia ante el aumento de la cantidad de elementos debido al problema de los objetos similares distantes en el tiempo. Este índice podría redisen̄se duplicando ramas con las mismas distancias (es decir, correspondientes a las mismas firmas) de tal manera de que cada una contenga un grupo de unicamente objetos cercanos en el tiempo.

- El tamaño de los intervalos de la *Línea del Tiempo* del *Event-FHQT* actualmente es fijo. Si la distribución de los objetos en el tiempo no es uniforme, probablemente sea conveniente utilizar intervalos de amplitud variable para que la cantidad de objetos por cada intervalo no sea muy diferente. Otra variación posible es la modificación la cantidad de pivotes utilizada en cada intervalo, de tal manera de que sea función de la cantidad de objetos o eventos asociados al intervalo.

- Ambas variantes para distancias continuas de los índices presentados necesitan mayor experimentación. Por ejemplo, se requieren criterios para calcular la cantidad óptima de ramas y para decidir si dicha cantidad debe mantenerse en todos los nodos o si es más conveniente que sea particular para cada nodo. Por otro lado, es de notar que esta adaptación para distancias continuas es aplicable también a índices métricos puros como el FHQT, FQT y FQA. Este es un resultado lateral importante.

- Todos los métodos presentados suponen su procesamiento 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.

- Las consultas al *Event-FHQT* podrían resolverse utilizando paralelismo debido a que los FHQTs correspondientes a distintos intervalos son independientes entre sí.

- Existen métodos de selección de pivotes, pero sólo toman en cuenta el aspecto métrico. Es conveniente analizar si es necesario definir nuevas formas para elegir los pivotes, que además del aspecto métrico tengan en cuenta el aspecto temporal, y si existe la necesidad, diseñar nuevas técnicas con este fin.

- Se podría considerar el aspecto temporal agregando dimensiones al vector que describe cada objeto y definiendo una función de distancia que mida ambos aspectos a la vez. Si bien inicialmente tiene algunos inconvenientes, como por ejemplo que no todos los objetos se representan mediante vectores o que usualmente no interesa la cercanía temporal de dos objetos, sino su inclusión, intersección, adyascencia, etc; este enfoque constituye una opción a analizar con mayor profundidad para determinar su validez.

- En la aplicación de métodos y técnicas a la resolución de problemas concretos, usualmente se requiere configurar, adaptar y ajustar los modelos teóricos a las características particulares del problema. En este aspecto, se considera conveniente el estudio de casos de uso donde se utilicen índices métrico-temporales para definir pautas que permitan determinar rangos de valores convenientes para cada uno de los parámetros. Por ejemplo, la cantidad de pivotes, el tamaño de los intervalos del *Event-FHQT*, la cantidad de ramas para las variantes continuas o la función de distancia más adecuada para resolver el problema.

Por último, se describe brevemente una aplicación de los modelos y métodos expuestos en este trabajo –que es de interés para el autor–, para ejemplificar su impacto e importancia en la resolución de problemas de la realidad. Actualmente existe una gran cantidad de cámaras de video públicas funcionando en distintos puntos de ciudades y rutas del país. Cada cámara usualmente registra de 25 a 30 cuadros (fotos) por segundo. La cantidad de imágenes almacenadas

diariamente por cada cámara (en formato de video), es alrededor de $2{,}160{,}000$. Asumiendo que existen 30 cámaras instaladas en las esquinas de una ciudad, por ejemplo, Rosario, obtenemos una base de datos de crece en razón de más de 60 millones de imágenes por día. Por otro lado, ya que la filmación es en tiempo real, cada imágen tiene asociado un instante de tiempo determinado. Este problema claramente se puede modelar a través de un espacio métrico-temporal.

Ante un hecho delictivo, una consulta de interés sobre esta base de datos podría ser *"encontrar una camioneta azul circulando por calles de la ciudad de Rosario entre las 2 y las 6 de la mañana del día 12 de diciembre de 2012"*. Para construir esta consulta se puede seleccionar, de un conjunto de fotos o dibujos de vehículos genéricos, el más adecuado y superponerlo al fondo de la imágen, que siempre es el mismo ya que la cámara está fija (aunque hay que tener en cuenta que la iluminación es distinta de acuerdo al momento del día y las condiciones temporales).

Obviamente el tiempo de respuesta ante esta consulta debería ser el menor posible. Los modelos y métodos de acceso métrico-temporales expuestos en este trabajo son sumamente adecuados para resolver este tipo de problemas con eficiencia.

Bibliografía

[AS88] Ilsoo Ahn and Richard T. Snodgrass. Partitioned storage for temporal databases. *Inf. Syst.*, 13(4):369–391, 1988.

[AV96] Lars Arge and Jeffrey Scott Vitter. Optimal dynamic interval management in external memory (extended abstract). In *FOCS*, pages 560–569, 1996.

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[Ben79] J. L. Bentley. Multidimensional binary search trees in database applications. *IEEE Trans. Softw. Eng.*, 5(4):333–340, 1979.

[BG94] Gabriele Blankenagel and Ralf Hartmut Güting. External segment trees. *Algorithmica*, 12(6):498–532, 1994.

[BGO+96] Bruno Becker, Stephan Gschwind, Thomas Ohler, Bernhard Seeger, and Peter Widmayer. An asymptotically optimal multiversion b-tree. *VLDB J.*, 5(4):264–275, 1996.

[BJSS98] Rasa Bliujute, Christian S. Jensen, Simonas Saltenis, and Giedrius Slivinskas. R-tree based indexing of now-relative bitemporal data. In *VLDB*, pages 345–356, 1998.

[BK73] W. Burkhard and R. Keller. Some approaches to best-match file searching. *Comm. of the ACM*, 16(4):230–236, 1973.

[BKK96] S. Berchtold, D. Keim, and H. Kriegel. The x-tree: An index structure for high-dimensional data. pages 28–39, 1996.

[BNC01] B. Bustos, G. Navarro, and E. Chávez. Pivot selection techniques for proximity searching in metric spaces. In *XXI Conference of the Chilean Computer Science Society*, pages 33–40, 2001.

[BO97] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 357–368, 1997. Sigmod Record 26(2).

[BPGH06] 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.

[BPGH08] A. De Battista, A. Pascal, G. Gutierrez, and N. Herrera. Indices para bases de datos métrico-temporales. In *Actas del X Workshop de Investigadores en Ciencias de la Computación*, La Pampa, Argentina, 2008.

[BPGH09] A. De Battista, A. Pascal, G. Gutierrez, and N. Herrera. Bases de datos métrico-temporales. In *Actas del XI Workshop de Investigadores en Ciencias de la Computación*, San Juan, Argentina, 2009.

[BPHG10] A. De Battista, A. Pascal, N. Herrera, and G. Gutierrez. Metric-temporal access methods. *Journal of Computer Science & Technology,*, 10(2):54–60, 2010.

[Bri95] S. Brin. Near neighbor search in large metric spaces. In *Proc. 21st Conference on Very Large Databases (VLDB'95)*, pages 574–584, 1995.

[BY97] R. Baeza-Yates. *Searching: An algorithmic tour*, volume 37. Allen Kent and James G. Williams, editors, 1997.

[BYCMW94] R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixed-queries trees. In *CPM '94: Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, pages 198–212, London, UK, 1994. Springer-Verlag.

[BZ82] Jacov Ben-Zvi. *The time relational model*. PhD thesis, 1982. AAI8219638.

[Cha94] B. Chazelle. Computational geometry: a retrospective. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 75–94, New York, NY, USA, 1994. ACM.

[Cha07] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.

[CHRV05] E. Chávez, N. Herrera, C. Ruano, and A. Villegas. Una implementación completa del fqtrie. In *VII Workshop de Investigadores de Ciencias de la Computación*, pages 61–65, 2005.

[CMN99] E. Chávez, J. Marroquín, and G. Navarro. Overcoming the curse of dimensionality. *European Workshop on Content-Based Multimedia Indexing (CBMI'99)*, pages 57–64, 1999.

[CMN01] E. Chávez, J. Marroquín, and G. Navarro. Fixed queries array: A fast and economical data structure for proximity searching. *Multimedia Tools and Applications (MTAP)*, 14(2):113–135, 2001.

[CNBYM01] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[CPRZ97] P. Ciaccia, M. Patella, F. Rabitti, and P. Zezula. Indexing metric spaces with m-tree, 1997. Sistemi Evolui per Basi di Dati.

[CU85] J. Clifford and A. Uz Tansel. On an algebra for historical relational databases: two views. *SIGMOD Rec.*, 14(4):247–265, 1985.

[DD02] C. Date and H. Darwen. *Temporal Data and the Relational Model*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[DGK+94] C. Dyreson, F. Grandi, W. Kafer, N. Kline, N. Lorentzos, Y. Mitsopoulos, A. Montanari, D. Nonen, E. Peressi, B. Pernici, J. Roddick, N. Sarda, M. Scalas, A. Segev, R. Snodgrass, M. Soo, A. Tansel, P. Tiberio, and G. Wiederhold. A consensus glossary of temporal database concepts. *SIGMOD Rec.*, 23(1):52–64, 1994.

[Eas86] Malcolm C. Easton. Key-sequence data sets on inedible storage. *IBM Journal of Research and Development*, 30(3):230–241, 1986.

[EWK90] Ramez Elmasri, Gene T. J. Wuu, and Yeong-Joon Kim. The time index: An access structure for temporal data. In *VLDB*, pages 1–12, 1990.

[FLL93] A. Farago, T. Linder, and G. Lugosi. Fast nearest-neighbor search in dissimilarity spaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9):957–962, 1993.

[FNC07] K. Figueroa, G. Navarro, and E. Chávez. Metric spaces library, 2007. Available at `http://www.sisap.org/Metric_Space_Library.html`.

[GS93] Himawan Gunadhi and Arie Segev. Efficient indexing methods for temporal relations. *IEEE Transactions on Knowledge and Data Engineering*, 5:496–509, 1993.

[Gün89] Oliver Günther. The design of the cell tree: An object-oriented index structure for geometric databases. In *ICDE*, pages 598–605, 1989.

[Gut88] A. Guttman. R-trees: a dynamic index structure for spatial searching. pages 599–609, 1988.

[Gut07] Gilberto A. Gutiérrez. Métodos de acceso y procesamiento de consultas espacio-temporales. *Tesis, Doctor en Ciencias mención Ciencias de la Computación, Universidad de Chile, Chile.*, 2007.

[Jen00] C. S. Jensen. *Introduction to temporal databases research - Chapter 1*. PhD thesis, Aalborg University, 2000.

[KM83] I. Kalantari and G. McDonald. A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, 9(5):631–634, 1983.

[KPS00] Hans-Peter Kriegel, Marco Pötke, and Thomas Seidl. Managing intervals efficiently in object-relational databases. In *VLDB*, pages 407–418, 2000.

[KS89] Curtis P. Kolovson and Michael Stonebraker. Indexing techniques for historical databases. In *ICDE*, pages 127–137, 1989.

[KS91] Curtis P. Kolovson and Michael Stonebraker. Segment indexes: Dynamic indexing techniques for multi-dimensional interval data. In *SIGMOD Conference*, pages 138–147, 1991.

[KTF98] Anil Kumar, Vassilis J. Tsotras, and Christos Faloutsos. Designing access methods for bitemporal databases. *IEEE Trans. on Knowl. and Data Eng.*, 10(1):1–20, January 1998.

[LM91] Sitaram Lanka and Eric Mays. Fully persistent b+-trees. In *SIGMOD Conference*, pages 426–435, 1991.

[LS89] David B. Lomet and Betty Salzberg. Access methods for multiversion data. In *SIGMOD Conference*, pages 315–324, 1989.

[McC85] Edward M. McCreight. Priority search trees. *SIAM J. Comput.*, 14(2):257–276, 1985.

[MOV94] L. Micó, J. Oncina, and E. Vidal. A new version of the nearest-neighbor approximating and eliminating search (AESA) with linear preprocessing-time and memory requirements. *Pattern Recognition Letters*, 15:9–17, 1994.

[Nav99] 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.

[ND99] Mario A. Nascimento and Margaret H. Dunham. Indexing valid time databases via b+-trees. *IEEE Trans. on Knowl. and Data Eng.*, 11(6):929–947, November 1999.

[NDE96] Mario A. Nascimento, Margaret H. Dunham, and Ramez Elmasri. M-ivtt: An index for bitemporal databases. In *DEXA*, pages 779–790, 1996.

[NS98] Mario A. Nascimento and Jefferson R. O. Silva. Towards historical r-trees. In *SAC*, pages 235–240, 1998.

[PBGH07] 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.

[PBGH08] 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.

[Ram97] Sridhar Ramaswamy. Efficient indexing for constraint and temporal databases. In *Proc. 6th Int. Conf. on Database Theory (ICDT), LNCS 1186*, pages 419–431. Springer-Verlag, 1997.

[RCH04] C. Ruano, E. Chávez, and N. Herrera. Discretización binaria del fqtrie. In *Actas del X Congreso Argentino de Ciencias de la Computación*, pages 100–111, Buenos Aires, Argentina, 2004.

[RS94] Sridhar Ramaswamy and Sairam Subramanian. Path caching: A technique for optimal external searching. In *PODS*, pages 25–35, 1994.

[SA89] R. Snodgrass and I. Ahn. A taxonomy of time in databases. pages 443–453, 1989.

[Sam84] H. Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260, 1984.

[SG89] Arie Segev and Himawan Gunadhi. Event-join optimization in temporal relational databases. In *In Proceedings of the Conference on Very Large Databases*, pages 205–215, 1989.

[SJ96] R. Snodgrass and C.S. Jensen. private communication. 1996.

[SK86] Arie Shoshani and Kyoji Kawagoe. Temporal data management. In *VLDB*, pages 79–88, 1986.

[Sno00] R. Snodgrass. *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann, 2000.

[SST09] Bela Stantic, Abdul Sattar, and Paolo Terenziani. The point approach to represent now in bitemporal databases. *J. Intell. Inf. Syst.*, 32(3):297–323, June 2009.

[ST99] B. Salzberg and V. Tsotras. Comparison of access methods for time-evolving data. *ACM Comput. Surv.*, 31(2):158–221, 1999.

[Sta05] Bela Stantic. Access methods for temporal databases. *Thesis, Doctor of Philosophy, University of Sarajevo, Bosnia.*, 2005.

[STS03] B. Stantic, J. Thornton, and A. Sattar. A novel approach to model now in temporal databases. *Temporal Representation and Reasoning, 2003 and Fourth International Conference on Temporal Logic. Proceedings*, pages 174–180, 2003.

[STTS10] Bela Stantic, Rodney W. Topor, Justin Terry, and Abdul Sattar. Advanced indexing technique for temporal data. *Comput. Sci. Inf. Syst.*, 7(4):679–703, 2010.

[TK95] Vassilis J. Tsotras and Nickolas Kangerlaris. The snapshot index: An i/o-optimal access method for timeslice queries. *Inf. Syst.*, 20(3):237–260, 1995.

[Uhl91] J. Uhlmann. Implementing metric trees to satisfy general proximity/similarity queries. Manuscrito, 1991.

[Vid86] E. Vidal. An algorithm for finding nearest neighbors in (approximately) constant average time. *Pattern Recognition Letters*, 4:145–157, 1986.

[VV94] Rakesh M. Verma and Peter J. Varman. Efficient archivable time index: A dynamic indexing scheme for temporal data. In *In Proceedings of the International Conference on Computer Systems and Education*, pages 59–72, 1994.

[VV97] Peter J. Varman and Rakesh M. Verma. An efficient multiversion access structure. *IEEE Transactions on Knowledge and Data Engineering*, 9:391–409, 1997.

[WJL93] G. Wiederhold, S. Jajodia, and W. Litwin. *Integrating temporal data in a heterogeneous environment*. Benjamin-Cummings, 1993.

[Yia93] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. 4th ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, pages 311–321, 1993.

[Yia99] P. Yianilos. Excluded middle vantage point forests for nearest neighbor search. In *DIMACS Implementation Challenge, ALENEX'99*, Baltimore, MD, 1999.

Apéndice A

Resultados Experimentales

A.1. Efecto de la amplitud del intervalo de consulta

Gráficos completos correspondientes a los índices *FHQT$^+$-Temporal* y *Event-FHQT$^+$* para los tres radios considerados, bases de datos NASA^{MT} y COLORS^{MT}, variando las cantidades de elementos entre 1.000 y 30.000 objetos (Figuras A.1, A.2 y A.3)

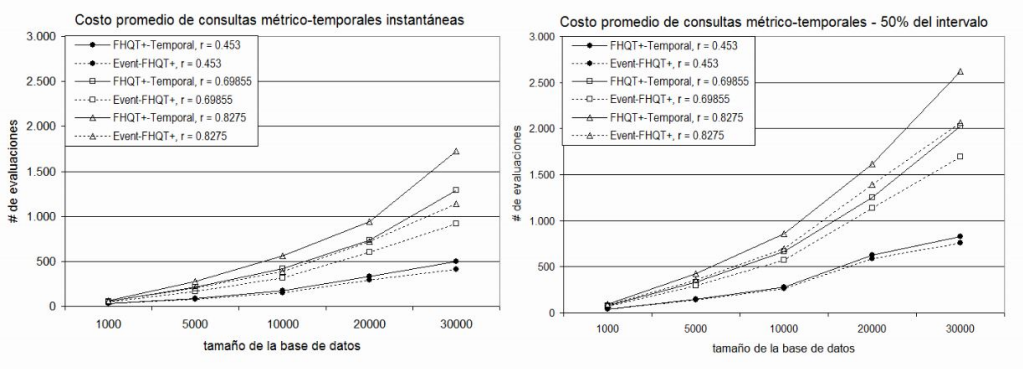

Figura A.1: *Costo promedio consultas métrico-temporales, índices FHQT$^+$-Temporal y Event-FHQT$^+$, base de datos NASA^{MT}, en función del tamaño*

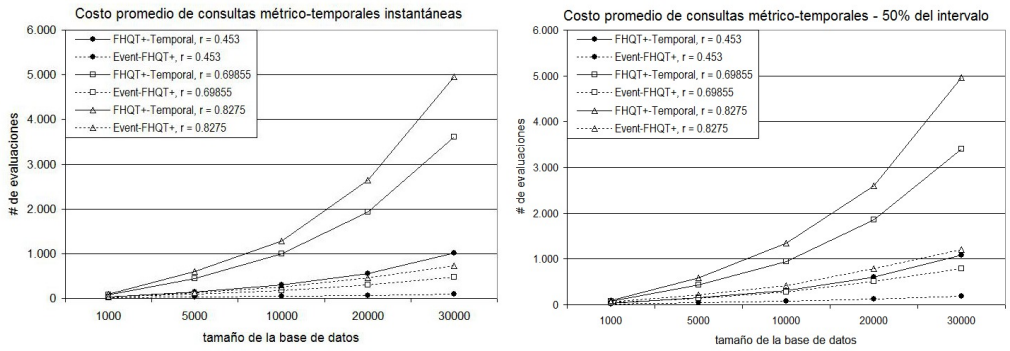

Figura A.2: *Costo promedio consultas métrico-temporales, índices FHQT$^+$-Temporal y Event-FHQT$^+$, base de datos COLORS^{MT}, en función del tamaño*

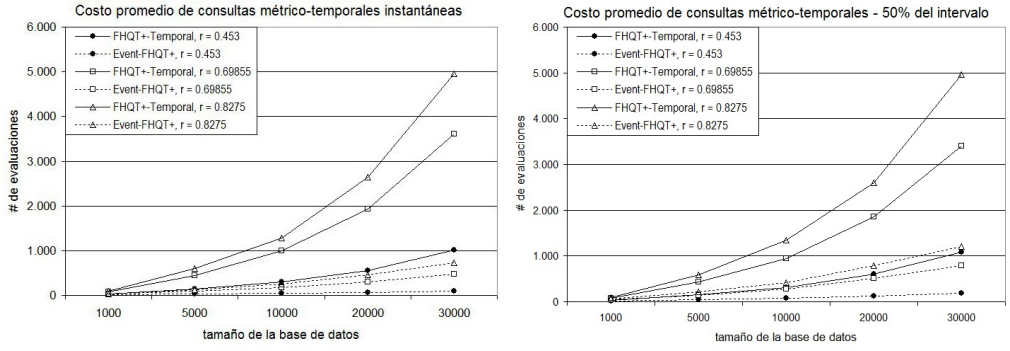

Figura A.3: *Costo promedio consultas métrico-temporales, índices FHQT$^+$-Temporal y Event-FHQT$^+$, bases de datos NASA^{MT} y COLORS^{MT} comparadas, en función del tamaño*