Búsquedas por similitud sobre objetos dinámicos

Anabella C. De Battista

Entre Ríos - Argentina

Noviembre de 2008

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL CONCEPCIÓN DEL URUGUAY
DEPTO. INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Búsquedas por similitud sobre objetos dinámicos

ANABELLA C. DE BATTISTA

ASESORES CIENTÍFICOS: **M. CS. NORMA EDITH HERRERA**
DR. GILBERTO GUTIÉRREZ RETAMAL

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

Este trabajo ha sido parcialmente financiado por el Programa de Posgraduación de Docentes de la Fac. Reg. Concepción del Uruguay (U.T.N.)

A mis padres Liliana y Oscar
y a mi hermano Germán
por su apoyo incondicional.

AGRADECIMIENTOS

En primer lugar quiero agradecer a la Facultad Regional Concepción del Uruguay de la Universidad Tecnológica Nacional por darme la posibilidad y el apoyo necesarios para concretar esta etapa de mi formación académica.

Mi más sincero agradecimiento a mi directora Norma Herrera, en primer lugar por convencerme de que podía tomar este desafío, y además por su paciencia, su comprensión y por estar siempre en el momento indicado con una frase de aliento, por brindarme sus conocimientos y su guía para concretar este trabajo.

Quiero agradecer también al Dr. Gilberto Gutiérrez por su orientación, los conocimientos y el tiempo que me ha dedicado durante la realización de esta tesis.

Especialmente quiero agradecer a mi colega y compañero de tesis Andrés Pascal, ya que sin su ayuda y su paciencia este camino hubiese sido mucho más difícil de recorrer.

A toda mi familia, que siempre me ha acompañado y me ha brindado el apoyo necesario para concretar cada nuevo desafío.

A mis amigos y compañeros de trabajo por su tolerancia, porque me han brindado durante este largo trayecto su compañía y en los momentos complicados me han dado el aliento necesario para continuar. A Giyo, Silvia, Vero, Facu, Jorge, Adri, Paula, Flopi, Andrés, Hernán, Ariel, Pablo, Agustín, Juan, David, Sole, Gabriel, Luciano y Nico.

A mis amigas de toda la vida Gise, July, Belén, Eta y Vivi que siempre están.

A Facu, que tanto me ha acompañado en el último trayecto de esta tesis, que me ha visto caer y me ha ayudado a volver a juntar fuerzas para lograr este objetivo, ya que sin él todo hubiese sido más complicado.

A mi ahijadita Lara por rescatarme de las ciencias y bajarme a tierra con su gracia e inocencia.

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

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Introducción

La búsqueda siempre ha sido un problema muy importante en el campo de las Ciencias de la Computación. Las bases de datos clásicas se organizan basándose en el concepto de búsqueda exacta sobre datos estructurados, capturando sólo un estado de la realidad modelada. Esto significa que la información se organiza en registros cada uno de los cuales contiene campos completamente comparables. Una búsqueda exacta en la base retorna todas aquellas tuplas cuyos campos coinciden con los aportados en la consulta. Una característica importante de las bases de datos clásicas es que capturan sólo un estado de la realidad modelizada, usualmente el más reciente. Una forma trivial de resolver las consultas en una base de datos es examinarla exhaustivamente, es decir, comparar cada elemento de la base datos con el elemento aportado en la consulta. Pero en general, esto es demasiado costoso para aplicaciones reales por lo que se preprocesa la base de datos con el objetivo de construir una estructura de datos o índice, diseñada para resolver eficientemente la búsqueda.

Este modelo tradicional no es aplicable para algunos tipos de datos que pueden almacenar las bases de datos actuales, como por ejemplo imágenes, audio, video, documentos de texto, secuencias de ADN o proteínas, comúnmente denominados datos multimedia. En primer lugar los datos generalmente son no estructurados, esto significa que es imposible organizarlos en registros compuestos por atributos. En segundo lugar, aún cuando tal estructuración fuera posible, la búsqueda exacta carece de interés en este ámbito; a nadie le interesa, por ejemplo, buscar

una imagen que sea exactamente igual a una dada. Y en tercer lugar, en muchas aplicaciones resulta de interés mantener todos los estados de la base de datos y no sólo el más reciente. Es decir, la problemática de almacenamiento y búsqueda difiere de las bases de datos clásicas y los índices consolidados en la actualidad en general consideran atributos cuyo tipo de datos son los clásicos (enteros, character, etc.). Por lo tanto se necesitan nuevos modelos, y en consecuencia nuevos índices, capaces de abordar este nuevo enfoque. Entre estos nuevos modelos se encuentran las bases de datos temporales y los espacios métricos, los que proveen formas de almacenamiento y métodos de recuperación eficiente de la información.

Las bases de datos temporales pueden almacenar y recuperar datos que dependen del tiempo, considerando al mismo como la tercera dimensión. Una verdadera base de datos temporal, es aquella que soporta tiempo válido y tiempo transaccional. El tiempo transaccional indica el tiempo en el que la información aparece reflejada en la base de datos como cierta, es decir, el momento en que se incorpora esa información en la base de datos. El tiempo válido, expresa el tiempo durante el cual la información es cierta en la realidad modelada. Por ejemplo, si se tiene una base de datos con información de todas las personas que han sido alumnos de una universidad, una consulta de interés es *buscar información de las personas que fueron alumnos de la universidad en el año 2001*.

Los espacios métricos permiten resolver búsquedas por similitud o proximidad sobre datos multimedia. Todas las aplicaciones en las que se aplica este modelo tienen como características comunes la existencia de un universo de objetos U y de una función de distancia o métrica d que modela la similitud entre los objetos del universo. Una de las consultas por similitud típicas en espacios métricos es la búsqueda por rango que consiste en recuperar aquellos objetos de la base de datos que están dentro de un determinado radio de tolerancia de un elemento dado. Siguiendo con el ejemplo anterior, si en la base de datos se almacena una foto de cada alumno, una consulta por similitud de interés es *recuperar la información de los alumnos cuya foto es parecida a una foto dada*.

Existen muchas aplicaciones donde resulta de interés realizar búsquedas por similitud pero teniendo en cuenta también la componente temporal. Continuando con el ejemplo, una consulta interesante para resolver es: *recuperar la información de las personas cuya foto es parecida a una foto dada y que fueron alumnos en el año 2001* (considerando que la foto tiene un intervalo de tiempo válido asociado). Esta consulta implica buscar teniendo en cuenta tanto la componente métrica como la componente temporal. Si se usa el modelo de bases de datos temporales, se puede buscar eficientemente teniendo en cuenta el tiempo pero posteriormente se debe realizar una búsqueda exhaustiva para filtrar teniendo en cuenta la compo-

nente métrica. Análogamente, si se usa el modelo de espacios métricos, se puede buscar eficientemente teniendo en cuenta la componente métrica pero posteriormente se debe realizar una búsqueda exhaustiva para filtrar teniendo en cuenta la componente temporal. En consecuencia, se hacen necesarios nuevos enfoques para atacar esta problemática.

En esta tesis se aborda el estudio de las consultas métrico-temporales con el fin de diseñar nuevos modelos que permitan resolverlas eficientemente. El objetivo principal de este trabajo es el diseño y la evaluación de algoritmos orientados a búsquedas en bases de datos métricas que consideran objetos dinámicos, es decir, objetos que pueden cambiar sus atributos en distintos instantes de tiempo. La propuesta es formular nuevos métodos de acceso métricos-temporales y desarrollar algoritmos que utilicen las estructuras de datos subyacentes de dichos métodos para procesar este nuevo tipo de consultas.

1.2. Aportes de la tesis

Los artículos publicados durante el proceso de elaboración de esta tesis son:

- N. Herrera, A. De Battista, A. Pascal. Políticas de Selección de Pivotes. En *Actas del VII Workshop de Investigadores en Ciencias de la Computación*, páginas 273-277, Río Cuarto, Argentina, 2005
- A. Pascal, N. Herrera, A. De Battista. Una propuesta para la selección de pivotes en índices métricos. En *Actas del XI Congreso Argentino de Ciencias de la Computación*, Entre Ríos, Argentina, 2005
- 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*, Buenos Aires, Argentina, 2006
- A. De Battista, A. Pascal, G. Gutierrez, N. Herrera. Un nuevo índice métrico-temporal: el Historical-FHQT. En *Actas del XIII Congreso Argentino de Ciencias de la Computación*, Corrientes, Argentina, 2007
- 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*, La Pampa, Argentina, 2008

1.3. Organización del Informe

Este informe de tesis está organizado en dos partes. La Primera Parte, *"Bases de datos temporales y Espacios Métricos"*, está compuesta por los Capítulos 2 y 3 que conforman el marco teórico de este trabajo.

En el Capítulo 2 se da una introducción a la temática de Bases de Datos Temporales, definiendo el modelo de datos temporal, las distintas dimensionalidades de tiempo con las que trabajan estas bases de datos y una clasificación de las mismas. Por último se presentan los diferentes tipos de consultas y una clasificación de los métodos de acceso.

En el Capítulo 3 se abordan los Espacios Métricos, se explica el concepto de búsquedas por similitud y los distintos enfoques para la construcción de algoritmos en este modelo. Por último se da una breve explicación del índice FHQT, que es un método basado en pivotes, en el que se basa el diseño de las estructuras presentadas en este trabajo.

La Segunda Parte *"Métodos de acceso y procesamiento de consultas métrico-temporales"*, compuesta por los Capítulos 4 y 5, presenta los aportes novedosos de esta tesis.

En el Capítulo 4 se definen los aportes teóricos de este trabajo: el modelo de Bases de Datos Métrico-Temporales y las Consultas Métrico-Temporales, y dos nuevos métodos de acceso que han sido propuestos para este modelo: *H-FHQT* y *New H-FHQT*.

La evaluación experimental de estas nuevas estructuras se realizó con dos bases de datos métricas, que debieron ser adaptadas para su utilización bajo este nuevo modelo. En el Capítulo 5 se explica como se generaron las bases de datos métrico-temporales de prueba y como se configuraron y ejecutaron las evaluaciones experimentales de ambos índices para cada lote de prueba.

Finalmente en el Capítulo 6, se presentan las conclusiones del informe y se propone el trabajo futuro.

Parte I

Bases de Datos Temporales y Espacios Métricos

Capítulo 2

Bases de Datos Temporales

Debido a que los conceptos básicos de bases de datos temporales conforman uno de los pilares fundamentales para el desarrollo de este trabajo, en este capítulo se esboza una introducción a los mismos y se presentan los distintos tipos de métodos de acceso existentes para este modelo.

2.1. Conceptos Básicos

La necesidad de registrar información que varía con el tiempo en las bases de datos ha producido un gran interés en la comunidad de investigadores de bases de datos desde la década del '70. Se trabajó desde entonces en la definición de una semántica del tiempo en el nivel conceptual, en el desarrollo de un modelo para bases de datos que manejan información que varía con el tiempo análogo al modelo relacional para bases de datos estáticas, y en el diseño de lenguajes de consulta temporales. Se vio también que son necesarios dos atributos de tiempo para capturar completamente la variación de información en esta dimensión.

Las bases de datos temporales soportan algún tipo de dominio de tiempo manejado internamente por el sistema administrador de la base de datos. Mantienen información acerca del pasado, el presente y en algunos casos, pueden predecir el futuro más probable. El manejo de datos que varían con el tiempo no está completamente implementado en la mayoría de las aplicaciones ya que muy pocos sistemas manejadores de bases de datos comerciales soportan datos temporales. En la actualidad, cuando se requiere manejo de datos temporales, se deben pro-

gramar esas funcionalidades de las aplicaciones, 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.

Tiempo 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"*. Se considera totalmente ordenado y en constante aumento. 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 vuelva 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*. Por ejemplo, tiempo absoluto es *23 de Febrero de 1979, 11:30 horas*, mientras que tiempo relativo es *11 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 que tiempo 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 aplica para las consultas. Por ejemplo, si se pregunta: *¿Cuál es el puesto de Juan ahora?* se obtendrán diferentes respuestas dependiendo del momento en que se ejecute la consulta. Pero si se pregunta: *¿Cuál era el puesto de Juan en Abril de 2007?* 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].

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. En [DGK+94] se acepta el término *quanta* para hacer referencia a la más pequeña unidad de tiempo indivisible y se define como: *En un modelo de datos, un quanta unidimensional es un intervalo de tiempo indivisible de alguna duración mínima fija. Un quanta n-dimensional es una región indivisible en el tiempo n-dimensional. Algunos tipos especiales de quantas importantes son: de tiempo válido, de tiempo transaccional y bitemporales*. Un *quanta* no es un punto o valor discreto, es 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). En ciencias de la computación, debido al uso de un modelo discreto donde un evento ocurre en un punto de tiempo en particular, como un *quanta* representa un segmento lineal, solo es posible registrar que ese evento ocurrió en un determinado *quanta*. 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 instantáneo, es decir, un hecho que ocurre en un tiempo específico. En el modelo discreto antes mencionado un *quanta* es una granularidad usada en el modelo temporal . Otro concepto a definir es *intervalo de tiempo*, que es el tiempo entre dos instantes. En un sistema que soporta una línea de tiempo compuesta de *quantas*, puede representarse un intervalo por un conjunto de *quantas* contiguas.

Timestamp Es un valor de tiempo asociado con algún objeto, por ejemplo, un valor de un atributo o una tupla. Algunas de las clasificaciones que puede hacerse de este valor son: timestamp válido, transaccional, de intervalo, instantáneo, elemento bitemporal.

2.2. Modelo de datos temporal

Un 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 debe agregarse a cada uno de sus componentes. 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. Además para cada restricción expresable en el modelo de datos no temporal M, hay una homóloga en el modelo de datos temporal M_t.

En los modelos que incorporan al tiempo la comunidad de bases de datos temporales ha propuesto tres modelos básicos para 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 existe un intervalo. En el modelo denso, como en el continuo, los puntos en la línea de tiempo corresponden a números reales pero entre dos puntos existe otro momento. En el modelo discreto cada número natural corresponde a una unidad de tiempo indivisible con una duración predefinida, el *quanta* [CT85]. 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].

2.3. 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 distintos tipos 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 RDBMS (Sistema de Gestión de Bases de Datos Relacional) particular. Los valores para el tiempo definido por el usuario son suministrados por el usuario y, como cualquier otro tipo de datos, pueden insertarse, actualizarse o eliminarse. El mejor ejemplo es el atributo de tipo *date* FechaDeNacimiento, ya que el usuario mantiene e interpreta por si mismo dicha información temporal. El RDBMS trata a este dato temporal como a cualquier otro tipo de atributo.

Tiempo Válido Una de las líneas de tiempo en el modelo de datos temporal se utiliza para capturar el tiempo en que un hecho es verdadero en la realidad modelada. Por ejemplo, se puede querer conocer desde cuando una persona se desempeña en el puesto laboral *A* y hasta qué momento permanece en dicho puesto. También puede ser necesario registrar eventos futuros, tales como boletos de avión para la semana próxima, conociendo la fecha exacta del vuelo. Al registro de información respecto a en qué momento fue, es o será válido un hecho en el mundo real, se le denomina *tiempo válido*, es decir, representa el tiempo en que un hecho es verdadero en la realidad modelada [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 estar asociado con cualquier cantidad de instantes o intervalos de tiempo, son casos especiales los hechos que se asocian a instantes o intervalos únicos. El tiempo válido generalmente es suministrado por el usuario. Las estructuras que se presentan en esta tesis se basan en esta dimensionalidad de tiempo.

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. Es decir, que hay 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 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 una actividad de la base de datos más que 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. Como una consecuencia de la semántica de esta dimensión, capturar este aspecto de las entidades de una base de datos redunda en eliminaciones puramente lógicas. 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.

2.4. Clasificación

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*.

2.4.1. 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 corrigen.

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 que estados de la base a lo largo del tiempo válido serán actualizados por dicha operación.

2.4.2. Bases de Datos Rollback

Estas bases de datos, también llamadas de *Tiempo Transaccional*, registran los cambios que ocurren durante el tiempo de 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, 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. Las características de este tipo de bases de datos es fundamental, por ejemplo, para aplicaciones de auditoría.

2.4.3. 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 que registran los cambios durante el tiempo de transacción, registran un nuevo estado de la base según el tiempo en que fue hecha la modificación en el sistema y además mantiene 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.

2.5. Registro de datos con tiempos

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 de alguna manera

Al registrar datos con tiempos la cuestión principal es qué nivel de unidades de datos se registra y que tipo de marca de tiempo se usa. La unidad de datos puede ser un valor de un atributo, tupla u objeto, colección de tuplas u objetos o inclusive 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 atributo.

Tuplas y objetos con tiempos asociados 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.

Atributos con tiempos asociados 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.

2.6. Métodos de Registro

La elección de que datos registrar y como hacerlo se determina según el modelo de datos subyacente. Generalmente, los datos pueden registrarse con un instante

de tiempo, un intervalo de tiempo o un elemento temporal. En todos los métodos los intervalos de tiempo se toman 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 tiempo en que esos hechos son verdaderos en el mundo real. Hablando teóricamente, en la base de datos se almacenan todos los instantes de tiempo en los cuales el hecho es verdadero para la realidad modelada. Cuando ese conjunto de instantes de tiempo contiene solo instantes continuos, obviamente resulta más conveniente representar ese 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 son el valor más pequeño y más grande, respectivamente, en la escala de tiempo del conjunto de instantes de tiempo continuo.

Registro mediante Instantes de Tiempo Para poder registrar los datos con 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. Las tuplas registradas con instantes de tiempo pueden modelarse extendiendo el esquema con un atributo de tipo *date*. En las tablas de eventos, los datos se registran con el instante de tiempo en que el evento correspondiente ocurre.

Registro mediante Intervalos de Tiempo Como vimos antes, los dos instantes asociados con una tupla representan el intervalo en que un hecho es válido en la realidad modelada. Los límites inferior y superior del intervalo de tiempo se corresponden con dos atributos adicionales 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 despliegue en varias tuplas. Como los intervalos de tiempo no son cerrados para las operaciones de intersección, diferencia y unión, puede ocurrir un problema

similar al calcular el resultado de una consulta. En el caso de registrar intervalos de tiempo para atributos, cada atributo en la relación se extiende agregando dos o cuatro atributos adicionales, dependiendo de cuantas dimensiones de tiempo se manejen en el modelo. Los conjuntos de esos atributos contienen la historia de cada atributo en una tupla. La redundancia se introduce si el mismo valor aparece varias veces durante períodos de tiempo no superpuestos para el atributo en una tupla.

Registro mediante 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. Extender 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 registro de atributos con elementos temporales no introduce redundancia, sin embargo conduce a relaciones muy complejas y es más difícil escribir consultas para asegurar restricciones de integridad y acceder eficientemente a los datos.

2.7. Tipos de consultas

Desde una perspectiva de consulta una base de datos de tiempo válido y una de tiempo transaccional son simplemente una colección de intervalos. Para bases de datos de tiempo simple (válido o transaccional) las consultas son similares. Como la mayoría de los métodos tienen las características del tiempo transaccional se discuten primero las consultas en este dominio, es decir, el intervalo T que sigue corresponde a un intervalo de tiempo transaccional y la *historia* tiene significado en el eje del tiempo transaccional [ST99].

Las consultas pueden categorizarse en las siguientes clases:

(I) Dado un intervalo continuo T, encontrar todos los objetos *vigentes* en ese intervalo.

(II) Dados un rango de claves y un intervalo de tiempo continuo T, hallar los objetos cuyas claves forman parte del rango y que están vigentes dentro del intervalo.

(III) Dado un rango de claves, devolver la historia de todos los objetos cuyas claves están en ese rango.

Un caso especial de (I) es cuando el intervalo T se reduce a un solo instante de tiempo transaccional t. Esta consulta se denomina *transacción pura instantánea*. En el ejemplo del empleado de la compañía esta consulta es *encontrar todos los empleados que trabajan en la compañía en el tiempo* t. Generalmente si un método de acceso resuelve eficientemente la consulta instantánea también es eficiente para el intervalo de consulta más general; por lo tanto se considera la consulta instantánea como una buena representante de las consultas clase (I).

De manera similar para la clase (II), los casos especiales incluyen combinaciones donde el rango clave y/o el intervalo de tiempo transaccional, contienen una clave única y un instante de tiempo único respectivamente. Por simplicidad, se considera el caso representativo en que el intervalo de tiempo se reduce a un único instante de tiempo transaccional; esta es la *consulta instantánea por rango* (por ejemplo, *hallar los empleados que trabajan en la compañía en el tiempo* t *y cuya clave primaria está dentro del rango* K).

Para la clase (III) se eligió el caso especial en que el rango de claves se reduce a una única clave como en: *hallar la cronología de salarios del empleado cuya clave es* k. Esta es la consulta de clave pura. Si el empleado k existió la respuesta deberían ser los salarios del empleado, sino la respuesta es vacía. En algunos métodos, debe ingresarse como parámetro de la consulta una instancia de un objeto empleado para obtener su historia de salarios previos (esto se debe a que esos métodos necesitan incluir un predicado de tiempo en su búsqueda). Esta consulta de clave pura especial (llamada *clave pura con predicado de tiempo*) es de la forma: *encontrar la cronología de salarios del empleado* k *que existió en el tiempo* t.

La consulta de clase (I) puede pensarse como un caso especial de clase (II) donde no se especifica rango de clave y la clase (III) un caso especial de (II) donde no se especifica un intervalo (además, todos los tiempos en la historia son de interés). Como los métodos de acceso en general son más indicados para responder consultas de una clase en particular, se discutirán las tres clases de forma separada.

Para bases de datos de tiempo válido podemos definir de forma similar la consulta instantánea pura de tiempo válido (*hallar todos los contratos válidos en el tiempo* v), instantánea por rango de tiempo válido (*hallar todos los contratos con identificadores dentro del rango* K *y válidos en* v), etc.

Una base de datos bitemporal permite consultas en ambas dimensiones de tiempo: *hallar todos los contratos válidos en v, y que fueron registrados en la base de datos en el tiempo transaccional t*. De todos los contratos en la colección $C(t)$ para t, la consulta recupera solo los que eran válidos en v.

La selección de las clases de consultas anteriores no es completa, solo contiene consultas básicas no triviales. En particular, las clases (I) y (II) se refieren a consultas basadas en intersección, es decir, la respuesta consiste en objetos cuyo intervalo contiene algún punto de tiempo de consulta o en general intersecta un intervalo de consulta. Según la aplicación, puede haber otras consultas de importancia. Por ejemplo, hallar todos los objetos con intervalos anteriores o posteriores a un punto/intervalo de tiempo de consulta, o todos los objetos con intervalos contenidos en un intervalo dado, etc.

Para distinguir entre consultas temporales puede utilizarse una notación de tres entradas llamada: *clave/válido/transacción* [SJ96]. Esta notación especifica qué atributos de objetos están involucrados en la consulta y de que manera. Cada entrada se describe como •, *rango*, $*$, o - (ver Cuadro 2.1). Un • para la entrada clave significa que el usuario especificó un valor único para que coincida la clave del objeto; un • para la entrada válida o transaccional implica que se especificó un instante de tiempo único para el dominio de tiempo válido o transaccional. El valor *rango* indica un rango de valores de la clave del objeto para la entrada clave, o un intervalo para las entradas válida o transaccional. Un $*$ significa que se acepta cualquier valor en esta entrada, mientras que - significa que la entrada no se aplica en la consulta. Por ejemplo, */-/• denota la consulta instantánea pura, *rango*/•/- es la consulta por rango e instante válido y •/-/* es la consulta de clave pura. En un ambiente bitemporal la consulta *hallar todos los contratos de la compañía que eran válidos en v, y que fueron grabados en la base de datos durante el intervalo de tiempo transaccional T* es un ejemplo de una consulta */•/*rango*. Como se ve, la notación de tres entradas trata con consultas de intersección pero puede extenderse fácilmente a la incorporación de descripciones de entradas extra para acomodar consultas temporales del tipo antes/después o algún otro tipo.

Entrada	**Clave**	**T. Válido**	**T. Transaccional**
•	Clave única	Instante	Instante
Rango	Rango de claves	Intervalo	Intervalo
$*$	Cualquier valor	Cualquier valor	Cualquier valor
-	No aplica	No aplica	No aplica

Cuadro 2.1: *Notación de tres entradas para consultas temporales*

2.8. Métodos de Acceso

Previamente se ha dado una clasificación de las dimensionalidades de tiempo con las que se puede trabajar en bases de datos temporales. A continuación se detallan algunas cuestiones a tener en cuenta en el diseño de métodos de acceso que soporten cada una de esas dimensiones [ST99].

Un método de acceso para una base de datos de tiempo transaccional necesita:

- Almacenar sus estados lógicos pasados
- Soportar inserción/modificación del estado lógico actual de los objetos
- Acceder eficientemente y consultar los objetos en cualquiera de sus estados

En general, un hecho puede ingresarse en la base de datos en un tiempo diferente al que ha sucedido en la realidad. Esto implica que el intervalo del tiempo de transacción asociado con un registro se relaciona con el proceso de actualización de la base de datos y, por lo tanto, puede no representar exactamente el período en que el objeto correspondiente estuvo vigente en la realidad.

Una base de datos de tiempo válido tiene una abstracción diferente. Para visualizarla, consideremos una colección dinámica de *objetos-intervalo*. Con este término se hace hincapié en que el objeto tiene un intervalo de tiempo válido que representa el período de vigencia de alguna propiedad del objeto (para enfatizar que el tiempo transaccional representa la actividad de la base de datos más que la realidad, se denomina a los objetos en dicha abstracción como *objetos-simples*). Los cambios admitidos son inserción/eliminación/modificación de un *objeto-intervalo*, pero la evolución de la colección (estados pasados) no se mantiene.

La noción de tiempo en este caso se relaciona al eje de tiempo válido. Dado un punto de tiempo, los *objetos-intervalos* pueden clasificarse como pasados, futuros o actuales (vigentes) en relación a ese punto, si su intervalo de tiempo válido 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* en la colección pueden modificarse, independientemente de su posición en el eje de tiempo.

Un método de acceso para una base de datos de tiempo válido debería:

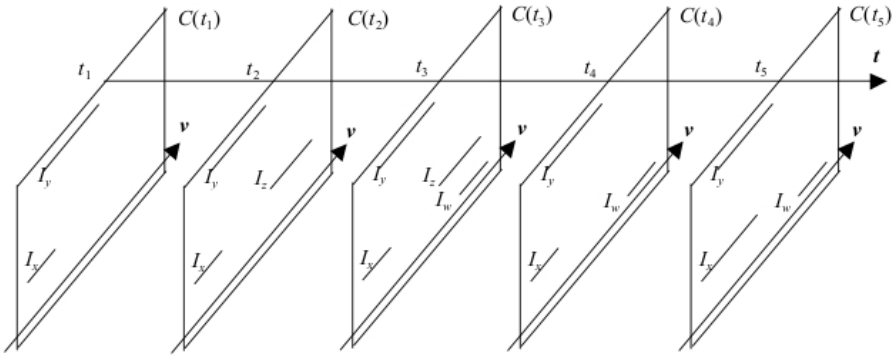

Figura 2.1: *Vista conceptual de una base de datos bitemporal*

- almacenar la última colección de objetos-intervalos,
- soportar inserción/eliminación/modificación sobre dicha colección,
- consultar eficientemente los *objetos-intervalo* que contiene la colección.

La realidad se representa de un modo más completo si se soportan ambas dimensiones. La abstracción de una base de datos bitemporal puede verse como el mantenimiento de la evolución (con el soporte de tiempo transaccional) de una colección dinámica de *objetos-intervalos* (tiempo válido). La Figura 2.1 ofrece 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].

Un método de acceso bitemporal debería:

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

La Figura 2.1 es de ayuda al resumir las diferencias entre los problemas subyacentes de varios tipos de bases de datos. Una base de datos de tiempo transaccional difiere de una base de datos bitemporal en que mantiene la historia de un conjunto

evolución de *objetos-simples* en lugar de *objetos-intervalo*. Una base de datos de tiempo válido difiere de una bitemporal debido a que mantiene solo una colección de *objetos-intervalo* (la última). Cada colección $C(t_i)$ puede verse como una base de datos de tiempo válido separada. Una base de datos de tiempo transaccional difiere de una base de datos instantánea (tradicional) en que además mantiene los estados pasados en lugar de solo el último estado. Finalmente, la diferencia entre una base de datos de tiempo válido y una instantánea es que la primera mantiene *objetos-intervalo* (y dichos intervalos pueden consultarse).

2.8.1. Métodos de tiempo transaccional

Estos métodos tratan con cambios que ocurren en orden de tiempo incremental, una característica propia del tiempo transaccional. Esta propiedad afecta enormemente el proceso de actualización del método. Si se soportan cambios *fuera de orden* el costo de actualización se vuelve mucho mayor (prácticamente prohibitivo).

Métodos Key-Only

La característica básica de estos métodos es la organización de la evolución de los datos por clave, es decir, todas las versiones de una clave son agrupadas juntas lógica o físicamente. Esa organización hace a estos métodos más eficientes para transacciones en las que se consulta por clave pura. En el Cuadro 2.2 se presentan algunas características de estos métodos.

Métodos de Tiempo Transaccional		
Métodos Key-Only		
Método	**Autores**	**Características**
Reverse chaining	Ben Zvi	Agrupa las versiones previas de una clave dada en orden cronológico inverso
Time Sequence Arrays	Shoshani y Kawagoe	Conceptualmente es un arreglo bidimensional con una fila por cada clave, cada columna representa un instante de tiempo
Accession lists	Ahn y Snodgrass	Mejora al Reverse Chaining agrupando todas las versiones de una clave dada juntas. Es eficiente para responder consultas de clave pura, pero no así para instantáneas puras o por rango
C-Lists	Varman y Verma	Es la solución teórica óptima para la consulta transaccional de clave pura con predicado de tiempo

Cuadro 2.2: *Características de los Métodos Key-Only*

Métodos Time-Only

La mayoría de estos métodos registran los cambios (inserciones, eliminaciones, etc.) con el tiempo de transacción en que estos ocurren y los agregan en un *history log*. Como no se hace agrupamiento de datos por clave, estos métodos optimizan las consultas instantáneas y por rango (en base al tiempo transaccional). Como los cambios ocurren en orden cronológico, pueden proveer procesamiento constante de actualizaciones (ya que el cambio consiste simplemente en insertar un registro al final del *history log*); esta es una ventaja importante en aplicaciones donde hay cambios frecuentes y la base de datos debe seguir esos cambios de manera *online*. Para lograr un tiempo de consulta eficiente, la mayoría de los métodos utiliza algún índice por sobre los *history logs* que indexa los instantes en los que se producen los cambios. En el Cuadro 2.3 se presentan algunas características de algunos de estos métodos.

Métodos de Tiempo Transaccional		
Métodos Time-Only		
Método	**Autores**	**Características**
Append-only Tree	Gunadhi y Segev	Optimiza event-joins
Time Index	Elmasri y otros	Propuesto inicialmente para almacenar tiempo válido, pero como incorpora características de tiempo transaccional
Two-Level Attribute/Time Index	Elmasri y otros	Responde consultas instantáneas por rango más eficientemente que el Time Index
ST-Tree	Gunadhi y Segev	Responde eficientemente consultas de clave pura e instantáneas por rango
Archivable Time Index	Verma y Varman	Puede responder consultas de clave pura e instantáneas por rango si cuenta con otro índice para regiones clave
Time Index$^+$	Kourmajian y otros	Mejora los altos requerimientos de almacenamiento y actualización
Snapshot Index	Tsotras y Kangelaris	Logra la solución óptima al problema de E/S para transacciones instantáneas puras
Windows Method	Ramaswamy	Resuelve de manera óptima las consultas instantáneas puras en memoria principal

Cuadro 2.3: *Características de los Métodos Time-Only*

Métodos Time-Key

Para responder una consulta instantánea por rango de claves eficientemente resulta una buena estrategia agrupar los datos por tiempo de transacción y por clave en páginas; así los datos lógicamente relacionados para este tipo de consultas están almacenados cerca y se minimiza el número de páginas a las que se accede.

Estos métodos se basan en árboles balanceados cuyas hojas corresponden dinámicamente a regiones en el espacio de transacción bidimensional *time-key*. Mientras los cambios ocurran en orden de tiempo creciente, las claves en las que se aplican los cambios están desordenadas. Por lo tanto, hay un proceso de actualización logarítmico por cada cambio y la información se registra según los valores de las claves en el mencionado espacio *time-key*.

Métodos de Tiempo Transaccional		
Métodos Time-Key		
Método	**Autores**	**Características**
Overlapping B^+-Tree; Exodus Large Storage Object	Manolopoulos y otros; Burton y otros	Responden consultas de clave pura como el resto de los métodos basados en B^+-Tree agrupando registros de datos.
Time Split B-Tree	Lomet y Salzberg	Resuelve consultas de clave pura con predicados de tiempo
Segment R-Tree	Kolovson y Stonebraker	Reduce la superposición en los nodos hojas cuando se trata con grandes intervalos
Multiversion B-Tree	Becker y otros	Otra opción para los B+-trees parcialmente persistentes
MVAS	Varman y Verma	Resuelve consultas de clave pura con predicados de tiempo en un tiempo óptimo

Cuadro 2.4: *Características de los Métodos Time-Key*

Hay dos enfoques principales para el diseño de estos métodos: los basados en variantes de *R-Trees* y los basados en variantes de B^+*-Trees* [ST99]. Una importante ventaja de usar métodos basados en *R-Trees* es que éstos pueden representar dimensiones adicionales sobre el mismo índice (por ejemplo, podrían soportar ambas dimensiones de tiempo). Una desventaja de estos métodos es que no pueden garantizar una buena performance en el tiempo de consulta y actualización para el peor caso. Sin embargo, esos peores casos son generalmente patológicos (es decir, que no suceden con frecuencia). En la práctica, los *R-Trees* han mostrado una buena performance en el caso promedio. Otra característica de los métodos basados en *R-Trees* es que el t_f de un intervalo se asume conocido cuando el registro se inserta en el método, que no es una propiedad del tiempo transaccional. Se presenta un resumen de algunas características de métodos de esta clase en el Cuadro 2.4.

2.8.2. Métodos de Tiempo Válido

Un índice de tiempo válido es un índice temporal que facilita el acceso a conjuntos de datos de tiempo válido. En una base de datos tradicional los índices se utilizan para consultas de selección. Al acceder a bases de datos de tiempo válido dichas selecciones involucran la dimensión de tiempo válido (el tiempo en que

un hecho es válido en la realidad modelada). Las características de esta dimensión implica varias propiedades que el índice temporal debería cumplir para ser eficiente.

El problema de indexar bases de datos de tiempo válido se puede reducir a indexar colecciones dinámicas de intervalos, donde un intervalo representa la validez temporal de un registro. En este caso se utiliza el término "intervalo" para indicar un subconjunto convexo del dominio 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. En cambio hay varios enfoques propuestos para el problema asociado de manejar intervalos en almacenamiento secundario. Dada la dificultad del problema, la mayoría de esos enfoques intentan lograr un buen comportamiento para el peor caso; como resultado, son principalmente de importancia teórica.

En el Cuadro 2.5 se presentan ciertas características de algunos de estos métodos de acceso.

Métodos de Tiempo Válido		
Método	**Autores**	**Características**
Segment Tree	Bentley	Estructura de datos en memoria principal que resuelve la consulta instantánea válida
Priority Search Tree	McCreight	Provee la solución óptima en memoria principal para la consulta instantánea válida
Cell-tree	Gunther	Para intervalos representados como segmentos lineales en un espacio bidimensional clave-tiempo
Metablock Tree	Kanellakis y otros	Resuelve el problema de la gestión de intervalos dinámicos
External Segment Tree	Blankenagel y Guting	Versión paginada del Segment Tree. Con modificaciones puede responder consultas con predicados clave (como la consulta rango/punto/-)
Patch Caching	Ramaswamy y Subramanian	Técnica que resuelve consultas por rango bidimensionales
MAP21	Nascimento y otros	Otro enfoque para indexar bases de datos de tiempo válido
External Interval Tree	Arge y Vitter	Solución óptima de E/S para la consulta instantánea válida

Cuadro 2.5: *Características de los Métodos de Acceso de Tiempo Válido*

2.8.3. Métodos Bitemporales

Un índice bitemporal es una estructura de datos que soporta ambas dimensiones de tiempo, es decir, el tiempo transaccional y el tiempo válido. A pesar de que los datos relativos a *now* son parte natural de las bases de datos bitemporales,

solo unos pocos métodos de acceso reconocen su existencia. Una tupla se considera relativa a *now* si la validez del hecho se corresponde con el tiempo actual o si la tupla es parte del estado actual de la base de datos, sin eliminaciones lógicas, es decir, el final del tiempo válido y/o transaccional sigue el incremento del tiempo actual. Algunas estructuras asumen que el tiempo válido no puede ser relativo a *now*, y por lo tanto no dependen del tiempo actual y que tv_i y tv_f son conocidos cuando la información se registra en la base. Esta restricción hace que dichos métodos de acceso no resulten aptos para aplicaciones temporales prácticas.

En el Cuadro 2.6 se presentan algunas descripciones de estas estructuras.

Métodos Bitemporales		
Método	***Autores***	***Características***
Interval Tree	Edelsbrunner	Trabaja en memoria principal y resuelve las consultas */punto/punto y */rango/punto
R-Tree	Guttman	Visualiza los intervalos asociados con un objeto bitemporal como un rectángulo acotado.
Dual-Root Mixed Media R-Tree	Kolovson y Stonebraker	Almacena intervalos y consta de dos R-Trees
Bitemporal Interval Tree	Kumar y otros	Creado a partir de hacer un Interval Tree parcialmente persistente y bien paginado. Diseñado para las consultas */punto/punto y */rango/punto
M-IVTT	Nascimento y otros	Es una extensión del Time-Index a un entorno bitemporal
Bitemporal R-Tree	Kumar y otros	Creado a partir de hacer un R-Tree parcialmente persistente. Diseñado para responder las consultas bitemporales más generales rango/rango/punto y rango/punto/punto
2R-Tree	Kumar y otros	Resuelve el problema de superposición que presenta el R-Tree

Cuadro 2.6: *Características de los Métodos de Acceso Bitemporales*

Capítulo 3

Espacios Métricos

3.1. Introducción

La mayoría de las aplicaciones de computación requieren realizar operaciones de búsquedas sobre bases de datos. En el campo de las bases de datos tradicionales se trabaja con el concepto de búsquedas exactas: la base de datos se divide en registros, teniendo cada registro campos totalmente comparables; una consulta a la base retorna todos aquellos registros cuyos campos coincidan con los aportados en la búsqueda. Actualmente las bases de datos han incluido la capacidad de almacenar datos no estructurados tales como imágenes, sonido, texto, video, datos geométricos, etc. Estructurar este tipo de datos en registros para adecuarlos al concepto tradicional de búsqueda exacta es difícil en muchos casos y hasta imposible si la base de datos cambia más rápido de lo que se puede estructurar (por ejemplo, la web). Aún cuando esto pudiera hacerse, las consultas que se pueden satisfacer con la tecnología tradicional están limitadas a variaciones de búsquedas exactas. En este ámbito las búsquedas que son de interés son aquellas en las que se pueden recuperar objetos similares a uno dado. Este tipo de búsqueda se conoce con el nombre de *búsqueda por proximidad* o *búsqueda por similitud* y surge en diversas áreas. Algunas de ellas son:

Consultas por contenido en objetos multimedia Los nuevos tipos de datos tales como imágenes, huellas digitales, audio y video (llamados tipos de datos multimedia) no pueden consultarse en el sentido clásico, ya que no pueden ordenarse ni compararse por igualdad. En general en ninguna aplicación resulta de

interés, por ejemplo, buscar un segmento de audio exactamente igual a otro dado. En aplicaciones multimedia todas las consultas buscan objetos *similares* a uno dado.

Recuperación de información Aunque no se considera un tipo de dato multimedia, la recuperación de texto no estructurado posee problemas similares a la recuperación multimedia. Esto se debe a que los documentos de texto en general no están estructurados para proveer fácilmente la información deseada. En ellos pueden buscarse cadenas que estén presentes o no, pero en la mayoría de los casos se buscan conceptos semánticos. El problema se resuelve básicamente recuperando documentos similares a uno dado. El usuario puede presentar un documento como una consulta, para que el sistema encuentre documentos que se asemejen.

Recuperación de texto Otro problema relacionado a la recuperación de texto es la ortografía. A partir del surgimiento de grandes bases de datos en las que existe un bajo control de la calidad del contenido (por ejemplo, la web) y de la escritura, los errores de ortografía son muy comunes en el texto y en la consulta; por lo tanto ocurre que los documentos que contienen palabras mal escritas no son recuperables por una consulta que esté correctamente escrita, del mismo modo, si la consulta está mal escrita, no podrán recuperarse aquellos documentos en los que las palabras que se están buscando están bien escritas. Existen modelos de similitud entre palabras que capturan ese tipo de errores.

Biología computacional El ADN y las secuencias de proteínas son los objetos básicos del estudio de la biología molecular. Como pueden modelarse como texto, se presenta el problema de hallar una secuencia de caracteres dada dentro de una secuencia más larga. Sin embargo, es improbable una correspondencia exacta, y los biólogos computacionales están más interesados en encontrar partes de una secuencia larga que resulten similares a una secuencia más corta dada. El hecho de que la búsqueda no sea exacta se debe a diferencias menores en las tramas genéticas que describen las secuencias de funcionalidades similares, evolución, errores experimentales, etc.

Todas estas aplicaciones que se han mencionado tienen como característica común la existencia de un universo de objetos y de una función de distancia que modela la similitud entre los objetos del universo. Esto da origen a un nuevo modelo de bases de datos conocido con el nombre de *espacio métrico*.

Formalmente, un *Espacio Métrico* se define como un par (U, d) donde U es el universo de objetos válidos del espacio y $d : U \times U \longrightarrow R^+$ es una función de distancia definida entre los elementos de U que mide su similitud; esto significa que a menor distancia más cercanos o similares son los objetos. Llamaremos base de datos a cualquier subconjunto finito $X \subseteq U$ cuya cardinalidad es $|X| = n$. La función d cumple con las propiedades características de una función 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 en la mayoría de los casos se verifica también:

- $\forall x, y \in U,\ x \neq y \Rightarrow d(x, y) > 0$ (positividad estricta)

Las propiedades anteriores aseguran una definición consistente de la función, pero no hacen que d pueda usarse para evitar comparaciones durante una búsqueda por similitud. Para que d sea realmente una función métrica debe verificar además:

- $\forall x, y, z \in U,\ d(x, y) \leq d(x, z) + d(z, y)$ (desigualdad triangular)

Si d no satisface la positividad estricta, se habla de un espacio *pseudo métrico*. En general en estos casos se adaptan las técnicas de espacios métricos para identificar a todos los objetos a distancia 0 como un único objeto. Esto es posible dado que $d(x, y) = 0 \Rightarrow \forall z, d(x, z) = d(y, z)$ (puede demostrarse fácilmente usando la desigualdad triangular).

En algunos casos se tienen *cuasi métricas* si no se cumple la propiedad de simetría. En estos casos, se puede definir a partir de d una nueva función d', que sea simétrica: $d'(x, y) = d(x, y) + d(y, x)$. Puede permitirse en algunos casos relajar la desigualdad triangular haciendo $d(x, y) \leq \alpha d(x, z) + \beta d(z, y) + \gamma$. En estos casos es posible seguir usando los mismos algoritmos de espacios métricos, previo escalamiento.

3.2. Consultas por similitud

Se pueden mencionar tres tipos de consultas por similitud que normalmente se utilizan en espacios métricos [CNBYM01a].

Búsqueda por rango $(q, r)_d$: Esta es una de las consultas típicas en este nuevo modelo de bases de datos. Dado un elemento $q \in X$ al que se denomina *query*, y un radio de tolerancia r, una búsqueda por rango consiste en recuperar los objetos de la base de datos que estén a distancia a lo sumo r de q, es decir:

$$(q, r)_d = \{x \in X : d(q, X) \leq r\} \tag{3.1}$$

Búsqueda del vecino más cercano NN(q) : Consiste en recuperar el (o los) elemento(s) más cercano(s) a un elemento q dado. En símbolos:

$$NN(q) = \{x \in X : \forall v \in X, d(q, x) \leq d(q, v)\} \tag{3.2}$$

Búsqueda de los k-vecinos más cercanos NNk(q) : Se busca recuperar los k elementos más cercanos a q en X. Esto significa encontrar un conjunto $A \subseteq X$ tal que:

$$|A| = k \wedge \forall x \in A, v \in (X - A) : d(q, x) \leq d(q, v) \tag{3.3}$$

Hay tres factores que afectan el tiempo T necesario para resolver una consulta: la cantidad de evaluaciones de la función de distancia hechas, la cantidad de operaciones adicionales realizadas que implican un tiempo extra de CPU y la cantidad de accesos a disco que implican un tiempo extra de E/S. En consecuencia, se puede calcular T de la siguiente manera:

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

En muchas ocasiones debido al alto costo de evaluación de la función d, las demás componentes de la fórmula del cálculo de tiempo pueden descartarse. Este es el modelo de complejidad usado en esta tesis, por consiguiente, la eficiencia de un algoritmo dependerá de la cantidad de evaluaciones de la función de distancia d realizadas para resolver la consulta. Es claro que una búsqueda por similitud puede resolverse de forma ineficiente examinando exhaustivamente la base de datos X, lo que implica $O(n)$ evaluaciones de distancia. Para evitar esto, se preprocesa X usando algún algoritmo de indexación (también conocidos como métodos de acceso) que construye de antemano una estructura de datos denominada *índice*, diseñada para ahorrar cálculos al momento de resolver la consulta.

3.3. Un ejemplo de Espacios Métricos: los Espacios Vectoriales

Con el fin de ilustrar los conceptos dados hasta el momento, en esta sección veremos un ejemplo concreto de espacios métricos, los espacios vectoriales k-dimensionales. Las métricas comúnmente usadas para espacios vectoriales son las de Minkowsky (familia L_p). Las distancias se definen sobre vectores de números reales, según la siguiente definición:

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

Algunos casos particulares de esta familia de distancias son:

- L_1, conocida como *distancia de Manhattan*.

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

- 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 cuando p tiende a infinito

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

La Figura 3.1 da un ejemplo de una 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, más conocida como *distancia euclideana*, se corresponde con nuestra noción de distancia espacial. Las búsquedas que utilizan L_∞ se corresponden con la búsqueda por rango clásica, donde el rango es un hiperrectángulo k-dimensional.

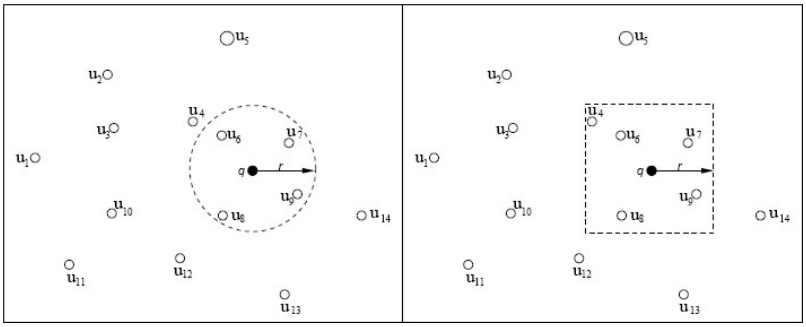

Figura 3.1: *Ejemplos de búsquedas por rango* $(q, r)_d$*, con* d*=*L_2 *(izquierda) y* $d = L_\infty$ *(derecha)*

Para espacios vectoriales existen implementaciones eficientes para búsquedas por similitud. Por ejemplo, *KD-Tree* [Ben75, Ben79], *R-Tree* [Gut88], *Quad-Tree* [Sam84], *X-Tree* [BKpK96]. Estas técnicas usan información sobre coordenadas para clasificar y agrupar puntos en el espacio. Desafortunadamente, las técnicas existentes se ven afectadas por la dimensión del espacio vectorial. La complejidad de las búsquedas por rango depende exponencialmente de la dimensión del espacio [Cha94]. Los espacios vectoriales pueden presentar grandes diferencias entre su dimensión representacional y su dimensión intrínseca (es decir la dimensión real en la que se pueden embeber los puntos manteniendo las distancias entre ellos). Por ejemplo, un plano embebido en un espacio de dimensión 50 tiene dimensión representacional 50 y dimensión intrínseca 2. Dado que existen técnicas eficientes para tratar espacios vectoriales, algunos diseñadores tratan de dar a su problema una estructura de espacio vectorial. Sin embargo, no siempre es fácil o posible hacer esto; por ejemplo esta conversión no es posible cuando se usa la función de similitud entre cadenas para comparar secuencias de ADN. Por esta razón varios autores recurren a espacios métricos generales, aún sabiendo que el problema de búsqueda es más difícil. Obviamente, también es posible tratar un espacio vectorial como un espacio métrico general usando sólo las distancias entre los puntos. Una ventaja inmediata de esto es que toma peso la dimensión intrínseca del espacio, independiente de cualquier dimensión representacional. En [CPRZ97] se presentan resultados que muestran que una estructura para espacios métricos (el *M-tree*) puede superar a una estructura para espacios vectoriales (el *R*-tree*) cuando se aplica a un espacio vectorial.

3.4. Métodos de acceso

3.4.1. Métodos de acceso basados en Particiones Compactas o tipo Voronoi

Para búsquedas por similitud en espacios vectoriales se utilizan los Diagramas de Voronoi. Esta es una importante estructura en Geometría Computacional para abordar problemas relacionados al *punto más cercano*. El Diagrama de Voronoi de un conjunto de objetos de un espacio es la división que éstos originan según la influencia de cada uno de ellos. Cada punto del espacio se asigna al objeto que tiene más influencia sobre el (el más cercano).

Resulta complicada la generalización de esta noción a espacios métricos generales, debido a que los algoritmos para su contrucción se basan en información de coordenadas. No obstante, el concepto en sí mismo ha inspirado varios enfoques para construir una aproximación al *grafo de Voronoi* o a su concepto dual, la *triangulación de Delaunay*, que es un grafo cuyos nodos son los objetos a partir de los cuales se generó el Diagrama de Voronoi, y cuyas aristas conectan aquellos nodos que comparten un borde en el correspondiente Diagrama de Voronoi (Figura 3.2). La Triangulación de Delaunay es la base para algoritmos eficientes de búsquedas por similitud.

Los métodos de acceso basados en este concepto definen una relación de equivalencia basándose en la proximidad a un conjunto de elementos que llamaremos *centros*. Sea $\{c_1, c_2, ..., c_k\}$ el conjunto de centros, la relación de equivalencia se define del siguiente modo:

$$x \sim_{\{c_i\}} y \Leftrightarrow NN_{\{c_i\}}(x) = NN_{\{c_i\}}(y)$$

$$\text{donde } NN_S(z) = \{w \in S \,/\, \forall w' \in S : d(z, w) \leq d(z, w')\}$$

La partición que esta relación genera en el espacio se conoce con el nombre de *partición compacta* y consiste en dividir el espacio asociando a cada c_i la partición formada por el conjunto de puntos que tiene a c_i como su centro más cercano. En espacios vectoriales se asume que la partición usa X como el conjunto de centros.

En espacios métricos generales la distancia $D([x], [y])$ en el espacio $\Pi[X]$ de clases de Voronoi será la menor distancia entre cualquier par de puntos $x \in [X]$ e $y \in [y]$. Para hallar $[q]$ se necesitan encontrar los vecinos más cercanos a q

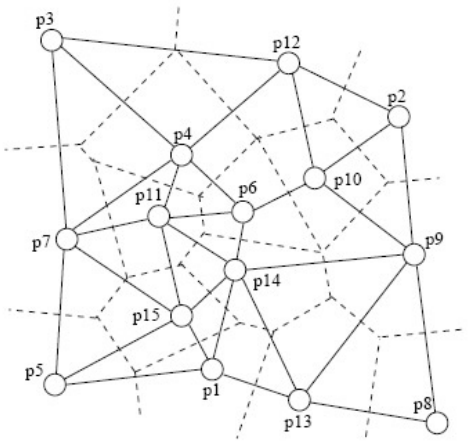

Figura 3.2: *Diagrama de Voronoi (líneas punteadas) para un conjunto de puntos en R^2 con distancia L_2; y su concepto dual, la Triangulación de Delaunay*

en el conjunto de centros. La respuesta a la consulta $([q], r)_D$ es el conjunto de clases intersectadas por la bola con centro en q (se denomina *bola de la query q* al conjunto de puntos que se encuentran a distancia a lo sumo r de q).

No es fácil determinar en espacios métricos generales si una clase es intersectada o no por una bola, para esto existen varios criterios, pero los dos más populares son:

- **Criterio del hiperplano:** es el más básico y el que mejor expresa la idea de partición de Voronoi. Básicamente, si c es el centro de la clase $[q]$ entonces la bola con centro q no intersecta $[c_i]$ si $d(q, c) + r < d(q, c_i) - r$. Es decir, si la bola asociada a q no intersecta el hiperplano que divide su centro más cercano c y el centro c_i, entonces cae fuera de la clase de c_i.
- **Criterio del radio de cobertura:** en este caso se trata de limitar la clase $[c_i]$ considerando la bola centrada en c_i que contiene todos los elementos de X que caen en la clase. Definimos el radio de cobertura de c para X de la siguiente manera:

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

  Por lo tanto, se ve que se puede descartar $[c_i]$ si $d(q, c_i) - r > cr(c_i)$.

Algunos ejemplos de este tipo de métodos de acceso son: Generalized-Hyperplane

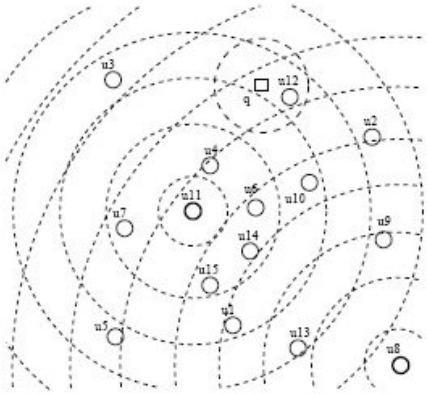

Figura 3.3: *Ejemplo de la relación de equivalencia inducida por la intersección de anillos centrados en los pivotes u_8 y u_{11}*

.

Tree (GHT) [Uhl91], Geometric Near-neighbor Acces Tree (GNAT) [Bri95], Spatial Approximation Tree (SAT) [Nav99], Bisector Tree (BST) [KM83].

3.4.2. Métodos de acceso basados en Pivotes

Estos algoritmos definen una relación de equivalencia basados en la distancia de los objetos de la base de datos a un conjunto de elementos preseleccionados a los que se llama *pivotes*. Sea $p_1, p_2, ..., p_k$ el conjunto de pivotes, dos elementos son equivalentes si y solo si están a la misma distancia de todos los pivotes.

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

Gráficamente, cada clase de equivalencia está definida por la intersección de varias capas de esferas centradas en los puntos p_i. En la Figura 3.3 se puede ver una relación de equivalencia utilizando solo dos pivotes, u_8 y u_{11}.

Durante la indexación, estos algoritmos asignan a cada elemento a de la base de datos el vector o firma $\phi(a) = (d(a, p_1), d(a, p_2), ..., d(a, p_k))$. Durante la búsqueda usan la desigualdad triangular junto con la firma de cada elemento para filtrar objetos de la base de datos sin medir su distancia a q.

Dada $(q, r)_d$, se computa la firma de la query q,

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

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 $d(q, p_i) \leq d(q, a) + d(a, p_i)$, si se reemplaza 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 puede demostrar que si $d(a, p_i) - d(q, p_i) > r$ entonces $d(a, q) > r$. Resumiendo, durante la búsqueda se pueden descartar todos aquellos elementos a tales que para algún pivote p_i se cumple que

$$|d(q, p_i) - d(a, p_i)| > r \tag{3.4}$$

Los elementos no descartados forman parte de una lista de candidatos, que posteriormente se comparan directamente con la query q.

Si el espacio E posee n elementos, se construye un índice con las nk distancias $d(x, p_i)$, y por lo tanto al momento de realizar la consulta (q, r) solo es necesario calcular las k distancias $d(q, p_i)$. A esto se le denomina complejidad interna de la consulta (q, r). 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 Ecuación 3.4 forman una lista de candidatos que deben ser comparados directamente con q y comprobar si verdaderamente se cumple la condición descrita en la Ecuación 3.1. A este cálculo de distancias adicionales se le denomina complejidad externa de la consulta (q, r).

También puede visualizarse una relación de equivalencia definida por un conjunto de k pivotes como la proyección al espacio vectorial R^k (Figura 3.4). En este último caso la i-ésima coordenada de un elemento es la distancia al i-ésimo pivote. Luego, a cada elemento x del espacio le corresponde el vector $\delta(x) = (d(x, p_1), d(x, p_2), ..., d(x, p_k)) \in R^k$, y ante una búsqueda $(q, r)_d$ se debe encontrar el conjunto $([q], r)_D$. En este caso particular, significa encontrar los elementos x tales que:

$$max_{1 \leq i \leq k} |d(x, p_i) - d(q, p_i)| = L_\infty(\delta(x), \delta(y)) \leq r$$

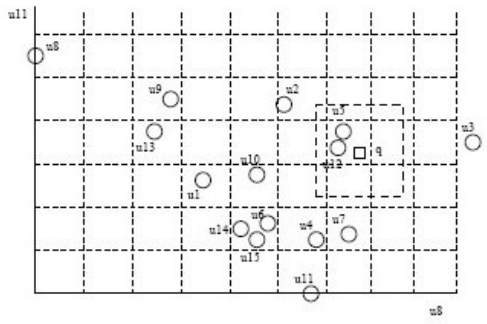

Figura 3.4: *Ejemplo de la transformación de la relación de equivalencia de la Figura 3.3 en un espacio vectorial de dimensión 2. También se ilustra la transformación de una búsqueda* $d(q, r)_d$

.

Esto significa que se ha proyectado el espacio métrico original (X, d) en el espacio vectorial R^k con la función de distancia L_∞.

Algunos algoritmos basados en pivotes son: Burkhard-Keller Tree [BK73], Approximating Eliminating Search Algorithm [Vid86], Vantage Point Tree [Yia93], Linear AESA [MOV94], Multi Vantage Point Tree [BO97], Excluded Middle Vantage Point Forest [Yia99], Spaghettis [CMBY99], Fixed Queries Array [CMN01].

Fixed Height Queries Tree (FHQT)

De los métodos basados en pivotes, en este trabajo se seleccionó el FHQT como base para el diseño de un índice métrico-temporal. Este índice admite dinamismo, una característica que no todos los índices métricos poseen. A continuación se explica su funcionamiento.

En [BY97] se presenta el FHQT, que es una variante del Fixed Queries Tree (FQT) [BYCMW94] en donde todas las hojas se encuentran a la misma altura. Originalmente estas estructuras fueron propuestas para funciones de distancias discretas, pero se pueden adaptar a distancias continuas discretizando los valores de las mismas [RCH04, CHCR05].

El árbol se construye a partir de un elemento p (pivote) que puede ser elegido arbitrariamente, o mediante algún procedimiento de selección de pivotes [BNC01], del universo U. Para cada distancia i se crea el conjunto C_i formado por todos

aquellos elementos de la base de datos que están a distancia i de p. Luego, para cada C_i no vacío se crea un hijo del nodo correspondiente a p, con rótulo i, y se construye recursivamente un FHQT teniendo en cuenta que todos los subárboles del mismo nivel usarán el mismo pivote como raíz. Este proceso recursivo, en el caso del FQT se repite hasta que queden menos de b elementos, los cuales se almacenan en una hoja. En el caso del FHQT se continúa hasta lograr que todas las hojas tengan menos de b elementos y estén en un mismo nivel. La Figura 3.5 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.

Ejemplo de Búsqueda

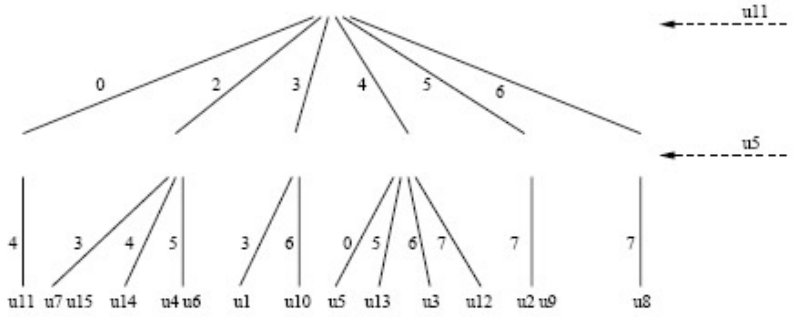

Figura 3.5: *FHQT con pivotes u11 y u5*

Para la consulta $(q, 2)_d$, siendo $(1, 2)$ la firma de q, el FHQT de la Figura 3.5 se comporta de la siguiente manera: como el radio de búsqueda es 2 y la distancia de q al primer pivote es 1, aplicando la condición de exclusión 3.6, se seleccionan las ramas etiquetadas con las distancias 0, 2 y 3 y se descartan las ramas 4, 5 y 6. En el siguiente nivel, se verifica qué ramas cumplen con la condición de similitud: para el primer subárbol, la rama 4, para el segundo subárbol, las ramas 3 y 4 y para el tercer subárbol, la rama etiquetada con 3. Por lo tanto el conjunto resultante está formado por los siguientes objetos: $u_{11}, u_7, u_{15}, u_{14}, u_1$. En la Figura 3.6 se muestra el pseudocódigo del algoritmo de consulta.

Algoritmo de consulta

A continuación se da el pseudocódigo del algoritmo de consulta del FHQT.

HFHQT $(q, r)_d$
1. calcular la firma f de q - distancias a cada uno de los pivotes de l_p
2. candidatos:= $\emptyset$
3. for all (ramas del FHQT)
4. candidatos := candidatos $\cup$ $h(q, r, f)_d$
5. resultado := x $\in$ candidatos | d(q, x) $\leq$ r
6. return resultado

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

Figura 3.6: *Pseudocódigo de consulta del FHQT*

3.5. Modelo Unificado

Las búsquedas por similitud en espacios métricos en general se resuelven en dos partes: en una primera instancia se preprocesa la base de datos y luego, se consulta. Durante la primera etapa se construye el índice que permitirá reducir la cantidad necesaria de evaluaciones de la función de distancia. La segunda etapa consiste propiamente en la realización de la búsqueda.

Si se analiza cada índice por separado parecería que cada uno de ellos sigue un enfoque totalmente diferente, sin embargo, existe un marco común que permite englobar todas las técnicas de indexación. Todos los métodos de acceso particionan la base de datos X en subconjuntos X_i. El índice permitirá hallar una lista de X_i que contienen elementos potencialmente relevantes para la consulta [CNBYM01a]. La búsqueda se divide en dos partes:

1. Recorrer el índice para obtener los X_i candidatos. El costo de este proceso se denomina *complejidad interna*.
2. Examinar exhaustivamente los X_i para hallar los elementos que forman parte del resultado de la búsqueda. El costo de este proceso se denomina *complejidad externa*.

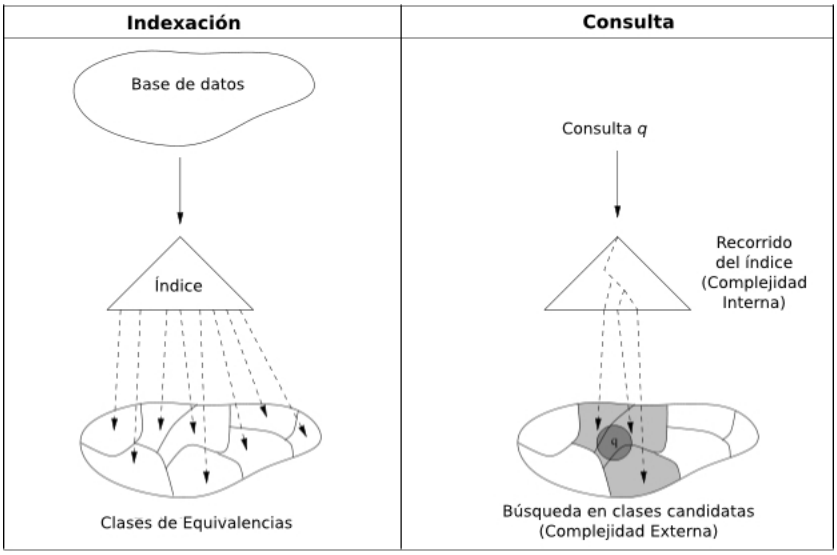

Figura 3.7: *Modelo general de métodos de acceso*

En la Figura 3.7 se puede apreciar un modelo general de los métodos de acceso.

Toda partición $\pi(X) = \pi_1, \pi_2, ..., \pi_n$ de un conjunto X induce una relación de equivalencia (que se denota con $\sim$) y toda relación de equivalencia induce una partición. La relación de equivalencia inducida por una partición $\pi(X)$ se define como:

$$\forall x, y \in X : x \sim y \Leftrightarrow x \text{ está en la misma parte que } y$$

Las clases de equivalencia $[x]$ de esta relación se corresponden con las partes π_i de la partición π, es decir, $\pi(X) = X/\sim$. En consecuencia, los métodos de acceso existentes definen una relación de equivalencia sobre el espacio métrico X. Las clases de equivalencia de X en el conjunto cociente $\pi(X)$ pueden considerarse como elementos de un nuevo espacio métrico, bajo alguna función de distancia $D : \pi(X) \times \pi(X) \rightarrow R^+$. A continuación se da la definición de la función D_0, que será de utilidad para encontrar la función D:

$$D_0 : \pi(X) \times \pi(X) \rightarrow R^+$$

$$D_0([x], [y]) = \text{mínimo}_{x \in [x], y \in [y]} d(x, y)$$

Se verifica que $\forall x, y \in X : D_0([x], [y]) \leq d(x, y)$, es decir, D_0 es contractiva. D_0 no es una métrica, dado que no cumple la desigualdad triangular, pero da el mínimo valor posible para que una función D sea contractiva. Esto significa que, cualquier función D que cumpla la desigualdad triangular y que limite inferiormente a D_0, sirve para definir un espacio métrico $\pi(X, D)$ que sea de utilidad para resolver nuestro problema de búsqueda. Luego, se puede convertir un problema de búsqueda en otro, que se espera que sea más sencillo. Para una búsqueda $(q, r)_d$:

- Se busca en el espacio $\pi(X, D)$ obteniendo como resultado $([q], r)_D = x \in X,\ D([x], [q]) \leq r$. Como D es contractiva, se puede asegurar que $(q, r)_d \subseteq ([q]; r)_D$.
- Se realiza una búsqueda exhaustiva en $([q], r)_D$ a fin de determinar los elementos que conforman la respuesta a la consulta $(q, r)_d$.

Existen dos enfoques generales para construir estas relaciones de equivalencia: uno está basado en pivotes y el otro está basado en particiones compactas o tipo Voronoi. A continuación se describen cada uno de ellos.

3.6. 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, CNBYM01a, CNBYM01b]. Si se considera el conjunto de elementos como puntos en un espacio vectorial, que es un caso particular de un espacio métrico, su dimensión corresponde al número de coordenadas que tienen los puntos que lo componen. Todos los algoritmos de búsqueda por similitud empeoran su rendimiento cuando aumenta la dimensión del conjunto, hecho conocido como *maldición de la dimensionalidad*. En [CNBYM01a] 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 se mantiene en dichos espacios.

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

$$\rho = \frac{\mu^2}{2\sigma^2} \tag{3.5}$$

Los parámetros μ y σ de la Ecuación 3.5 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 (E, d) es fácil ver que según la condición de exclusión de elementos (Ecuación 3.4), 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 Ecuación 3.6:

$$d(p_i, x) \notin [d(p_i, q) - r, d(p_i, q) + r] \tag{3.6}$$

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

Cuando la varianza del histograma disminuye a medida que aumenta la dimensión del espacio, la cantidad de elementos que se encuentran dentro del rango $[d(p_i, q) - r, d(p_i, q) + r]$ también aumenta, es decir, cada vez son menos los elementos que se pueden descartar. Alternativamente, si la media crece entonces r crece con la dimensión del espacio si se desea obtener un porcentaje fijo de elementos del conjunto. En espacios de alta dimensionalidad, prácticamente todos los elementos se transforman en candidatos a la verificación exhaustiva (complejidad externa), por lo que deben ser comparados directamente con la consulta. 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 3.8 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

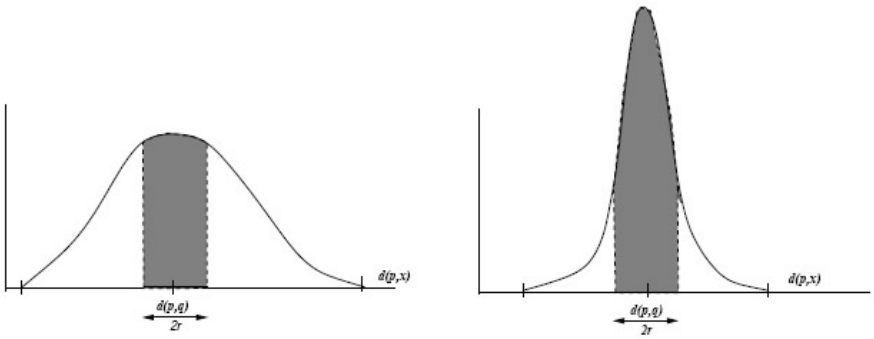

Figura 3.8: *Histogramas de distancias de baja dimensionalidad (izquierda) y de alta dimensionalidad (derecha)*

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 [CNBYM01a] 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, con 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 permitiría usar menos memoria, y por lo tanto se podrían abordar nuevas aplicaciones en donde los espacios métricos posean una alta dimensionalidad. En general, casi todos los algoritmos basados en pivotes los eligen de manera aleatoria dentro del conjunto de elementos. A continuación se presentan una serie de ejemplos de aplicaciones de Espacios Métricos, y posteriormente se introduce el modelo formal correspondiente.

Parte II

Métodos de acceso y procesamiento de consultas métrico-temporales

Capítulo 4

Bases de Datos Métrico-Temporales

4.1. Introducción

En este capítulo se presenta un nuevo modelo de bases de datos que se basa en conceptos de Espacios Métricos y de Bases de Datos Temporales. Este nuevo modelo permite el abordaje de aquellas situaciones en las que resulta necesario realizar búsquedas por similitud teniendo en cuenta además, la componente temporal. Por ejemplo, si tomamos una base de fotografías, algunas búsquedas que tienen significado son:

1. Hallar todas las fotografías similares a una dada cuya vigencia se corresponde a un instante de tiempo.
2. Hallar todas las fotografías similares a una dada cuya vigencia se corresponde con un intervalo de tiempo.

Ninguna de estas búsquedas puede ser resuelta eficientemente con índices temporales ya que los mismos están organizados para realizar búsquedas exactas sobre los objetos. Tampoco podríamos resolverlas con un índice métrico dado que, si bien estos índices permiten realizar búsquedas por similitud, sólo capturan un instante determinado de tiempo. En consecuencia, se hacen necesarios nuevos enfoques para atacar esta problemática.

Las Bases de Datos Métrico-Temporales permiten realizar búsquedas sobre objetos que tienen un intervalo de vigencia asociado y que no poseen un identi-

ficador que se pueda utilizar como clave de búsqueda, por lo cual tiene sentido consultar por similitud considerando también el aspecto temporal.

4.2. Espacio Métrico-Temporal

Sea O el universo de objetos válidos, un Espacio Métrico-Temporal se define formalmente como un par (U, d), con $U = O \times N \times N$ y la función $d : O \times O \rightarrow R^+$. Cada elemento $u \in U$ es una 3-upla (obj, t_i, t_f), donde obj es un objeto (por ejemplo, una imagen, sonido, cadena, etc) y $[t_i, t_f]$ es el intervalo de vigencia de obj definido sobre el conjunto N de los números naturales. La función de distancia d, que mide la similitud entre dos objetos, cumple con las propiedades de una métrica (positividad, simetría, reflexividad y desigualdad triangular).

En general las aplicaciones donde tienen sentido las consultas métrico-temporales tienen las siguientes características:

- No se pueden realizar búsquedas exactas sobre los objetos: los elementos de la base de datos no tienen un identificador (o un grupo de atributos) que se pueda utilizar como clave de búsqueda, o si existe, no se conoce en el momento de la consulta. Esto sucede por ejemplo, cuando los objetos se incorporan a la base de datos en tiempo real y no hay tiempo suficiente para clasificarlos en el momento.
- Los objetos tienen un intervalo de vigencia asociado. Este intervalo representa el período en el cual el objeto tiene significado para la realidad, por ejemplo el intervalo de tiempo en el que una persona es alumno de una universidad. En algunos casos, dicho intervalo se puede reducir a un instante de tiempo, por lo que se habla de objetos instantáneos.
- En las consultas se necesita combinar las búsquedas por similitud con el aspecto temporal.
- 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. Consulta Métrico-Temporal

Sea $X \in U$ el conjunto finito sobre el que se realizan las búsquedas, una consulta por rango métrico-temporal se denota mediante la 4-upla $(q, r, t_{iq}, t_{fq})_d$, y se define formalmente de la siguiente manera:

$$(q, r, t_{iq}, t_{fq})_d = \{o/(o, t_{io}, t_{fo}) \in X \wedge d(q, o) \leq r \wedge (t_{io} \leq t_{fq}) \wedge (t_{iq} \leq t_{fo})\} \tag{4.1}$$

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

Una forma trivial de resolver una consulta métrico-temporal es construir un índice métrico agregándole a cada objeto el intervalo de tiempo de vigencia del mismo. Luego, ante una consulta $(q, r, t_{iq}, t_{fq})_d$, se procede de la siguiente manera:

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

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

4.4. Métodos de Acceso Métrico-Temporales

4.4.1. Historical-FHQT

En esta Sección se presenta una estructura de acceso métrico-temporal denominada *Historical-FHQT (H-FHQT)*, que utiliza el índice métrico *FHQT* [BY97]

para tratar el aspecto métrico, e ideas de índices temporales mediante las cuales aborda el aspecto temporal.

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

Estructura

Formalmente, un H-FHQT es un par (l_i, l_p) donde:

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

Los árboles pueden tener distintas profundidades, y esto depende de la cantidad de elementos que se deban indexar. La cantidad de pivotes utilizada en un árbol se calcula como $\lceil log_2(|o_i|) \rceil$, donde $|o_i|$ es la cantidad de objetos vigentes en el instante i. De esta manera se evita que haya árboles demasiado profundos cuando la cantidad de objetos es baja, a fin de que la estructura no tenga un costo en almacenamiento excesivo.

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

La estructura es dinámica, permitiendo altas de dos tipos: históricas o de nuevos instantes. Un alta es histórica cuando se incorpora un objeto a un instante ya existente. El costo de esta inserción es el costo de calcular la firma del nuevo objeto, siempre que no haya que reestructurar el árbol. Un alta de un nuevo instante, toma como entrada el conjunto de objetos vigentes para ese instante, construye el *FHQT* correspondiente a dichos objetos, y lo agrega al final de la lista l_i como nuevo instante. Los instantes en los que no hay objetos vigentes, se

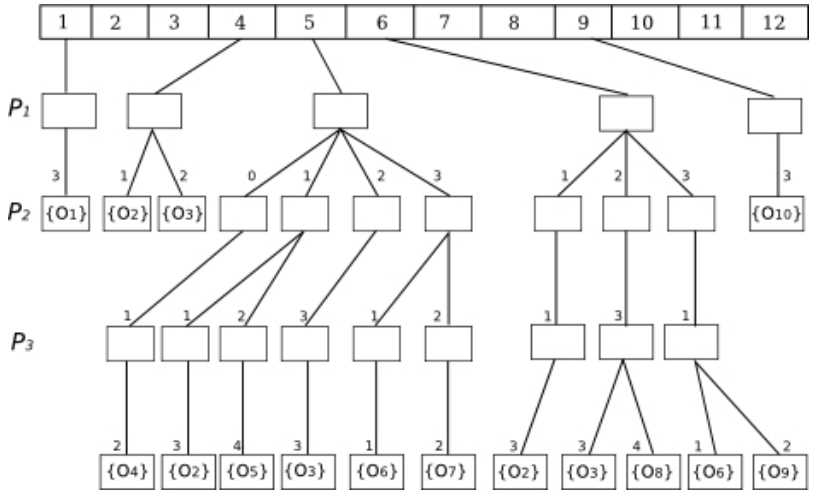

Figura 4.1: *Historical FHQT*

agregan a l_i como conjuntos vacíos, y se representan dentro de la lista con el valor *nil*.

Consulta

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

En la Figura 4.1 se muestra un ejemplo del *H-FHQT*. El intervalo total representado es $[1, 12)$. La estructura contiene *FHQT* sólo en los instantes 1, 4, 5, 6 y 9, ya que en el resto no hay objetos vigentes registrados. Como se ve, los árboles tienen diferente profundidad: los correspondientes a los instantes 1, 4 y 9 utilizan sólo un pivote; los correspondientes a los instantes 5 y 6 contienen FHQTs de 3 pivotes cada uno. En el ejemplo $l_p = (p_1, p_2, p_3)$ y $max = 3$.

Para la consulta $(q, 1, 3, 7)_d$, siendo (1, 2, 4) la firma de q, el *H-FHQT* se comporta de la siguiente manera: se acceden en forma directa los instantes 3, 4, 5, 6 y 7. Los instantes 3 y 7 se ignoran ya que no poseen un *FHQT* asociado.

En el instante 4, como la firma de q contiene como primer elemento a 1, y el radio de la búsqueda es 1, se toman las ramas 1 (1-1 $\leq$ 1 $\leq$ 1+1) y 2 (1-2 $\leq$ 2 $\leq$ 1+2) y pasan a formar parte de la lista de candidatos los objetos $\{O_2, O_3\}$

Para el instante 5 se toman las ramas 0 (1-0 $\leq$ 1 $\leq$ 1+0), 1 (1-1 $\leq$ 1 $\leq$ 1+1) y 2 (1-2 $\leq$ 2 $\leq$ 1+2), y se descarta la rama 3.

En el siguiente nivel del FHQT asociado con $t = 5$, las ramas que cumplen la restricción son 1, 1, 2 y 3, y en el último nivel del *FHQT* las ramas que resultan seleccionadas son 3, 4 y 3, por lo tanto los objetos candidatos para este instante son $\{O_2, O_3, O_5\}$. El mismo proceso se ejecuta para $t = 6$, dando como candidatos a $\{O_2, O_3, O_8\}$. Por último se unen los conjuntos de candidatos y se obtiene el conjunto resultante: $\{O_2, O_3, O_5, O_8\}$ y estos objetos se comparan directamente con la consulta q para obtener el resultado final.

En la Figura 4.2, se muestra el pseudocódigo del algoritmo de consulta al *H-FHQT*. La función toma como entrada el objeto de consulta q, el radio de búsqueda r, y el intervalo que se consulta $[t_{iq}, t_{fq}]$. Como primer paso se calcula la firma de q para todos los pivotes utilizados en al menos un árbol del *H-FHQT*; luego se consultan los *FHQT* correspondientes a cada uno de los instantes que conforman el intervalo consultado y se unen los conjuntos resultantes para obtener el resultado final.

HFHQT $(q, r, t_{iq}, t_{fq})_d$
```
1. calcular la firma f de q - distancias a cada uno de los pivotes de l_p
2. candidatos:= ∅
3. for all (fhqt h correspondiente a los instantes del intervalo [t_iq, t_fq]
del HFHQT)
4.    candidatos := candidatos ∪ h(q, r, f)_d
5. resultado := x ∈ candidatos | d(q, x) ≤ r
6. return resultado

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

Figura 4.2: *Pseudocódigo de consulta del H-FHQT*

Algoritmo de inserción

En la Figura 4.3 se presenta el algoritmo de inserción que utiliza el *H-FHQT*. El mismo toma como parámetros *unaListaDeObjetos* y el instante para el que

se realiza la inserción de objetos vigentes. Agrega el instante correspondiente a la línea del tiempo válido, y posteriormente procede a indexar los objetos que pertenecen al parámetro unaListaDeObjetos construyendo el FHQT que se asocia al instante dado.

Insercion H-FHQT (unaListaDeObjetos, instante)

```
1. AgregarInstante (lineaDelTiempo, instante).
2. cantPivotes = log(size(unaListaDeObjetos),2).
3. IF cantPivotes <1 THEN: [cantPivotes:= 1 ].
4. CrearFHQT.
5. 1 to: cantPivotes do [agregarPivote(FHQT,pivote)]
6. 1 to: size(unaListaDeObjetos) do [agregarObjeto(FHQT,objeto)]
```

Figura 4.3: *Pseudocódigo de inserción del H-FHQT*

H-FHQT versus solución trivial

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

El H-FHQT tiene menor costo espacial que el FHQT de la solución trivial, si se sigue la estrategia de utilizar cantidades logarítmicas de pivotes para los árboles correspondientes a cada instante. Esto se debe a que varios árboles pequeños ocupan menos espacio que un solo árbol de mayor profundidad, ya que normalmente el incremento en nodos de un nivel al siguiente es exponencial. En caso de utilizar una cantidad fija de pivotes para todos los árboles, igual a la de la solución trivial, el costo espacial del HFHQT alcanza a ser hasta el doble que el de un único FHQT.

En [DPGH07] se muestran resultados experimentales que avalan lo anteriormente expuesto. En dicho trabajo se observan reducciones drásticas en la cantidad de evaluaciones de distancias del H-FHQT respecto de una solución trivial para

una base de datos pequeña. El H-FHQT realiza entre un 45 % y 65 % menos de evaluaciones de distancia que la solucion trivial para consultas por intervalo y entre un 92 % y 98 % menos para consultas instantáneas. Por esta razón la solución trivial se descarta en los experimentos que se presentan en el próximo capítulo.

4.4.2. New Historical-FHQT

En esta Sección se presenta otro método de acceso métrico-temporal propuesto en este trabajo denominado *New Historical-FHQT* (New H-FHQT). Este índice está basado también en el uso del FHQT como estructura métrica y el enfoque temporal se ha abordado mediante el uso de una línea de tiempo, del mismo modo que en el H-FHQT.

El New H-FHQT consiste en una lista compuesta por los instantes válidos de tiempo. Para cada instante de la lista que posee objetos vigentes, se construye un FHQT para indexar los objetos.

La principal diferencia con el índice antes propuesto se da en la etapa de construcción de los FHQTs. En este caso se van tomando los primeros pivotes disponibles de la lista, que se considera una "lista circular", de tal manera que el FHQT del instante i, esté construido con pivotes diferentes a los de los instantes $i-1$ e $i+1$. Cuando en la construcción de un FHQT se utiliza el último pivote disponible de la lista, se vuelve al inicio de la misma y se comienzan a utilizar nuevamente los pivotes comenzando por el primero. La construcción de FHQTs consecutivos con diferentes grupos de pivotes da a la estructura mayor poder de filtrado de elementos desde el punto de vista métrico. Esta idea se plantea debido a que los objetos a indexar tienen un intervalo de vigencia asociado, por lo que pueden estar presentes en varios FHQTs consecutivos. Con este enfoque se logra que los objetos pasen por varios filtros, se eliminen la mayor cantidad de objetos mediante la firma y la desigualdad triangular y se reduzcan así la cantidad necesaria de evaluaciones de la función de distancia al momento de la ejecución de la consulta.

Estructura

Formalmente, un New H-FHQT es un par (l_i, l_p) donde:

- l_i es una lista $(f_1, f_2, ..., f_n)$ en la cual $[1, n)$ es el intervalo válido de tiempo, y cada f_i es un FHQT correspondiente al instante i, o *nil* si no hay ningún

objeto vigente en dicho instante.

- $l_p = (p_1, p_2, ..., p_{max})$ contiene todos los pivotes utilizados en todos los árboles.

Los árboles pueden tener distintas profundidades de acuerdo a la cantidad de elementos que se deban indexar. La cantidad de pivotes disponibles para la construcción de la estructura (max) será como mínimo un valor igual a la cantidad de pivotes que utiliza el árbol más profundo de la estructura y como máximo, el doble de dicha cantidad de pivotes. Se establece como cota superior ese valor ya que agregar una cantidad mayor de pivotes carece de sentido debido a que no produce un incremento significativo del filtrado y aumenta la cantidad de evaluaciones de la función de distancia, por lo que hace más costosas las búsquedas.

Cuando se calcula la firma de la consulta $f(q)$ se almacenan para cada pivote las distancias en un vector como el siguiente:

$d(q, p_1)$	$d(q, p_2)$	...	$d(q, p_{max})$	...	$d(q, p_k)$
3	7	...	3	...	...

Así, cuando la consulta involucra varios instantes de tiempo, se utiliza este vector para comparar si la distancia de la consulta a un pivote ya está calculada, y si no lo está, se calcula en el momento en que sea necesario. Este mecanismo evita recalcular distancias a pivotes que ya hayan sido evaluadas cuando se consultó por similitud en algunos de los FHQT de instantes anteriores.

Consulta

Para resolver las consultas métrico-temporales el New H-FHQT procede de la siguente manera: se seleccionan los instantes t_i involucrados en la consulta. Se ingresan en el vector antes mencionado los pivotes correspondientes a los FHQTs que indexan los objetos vigentes en dichos instantes. Cuando se presenta una consulta q se proceder a calcular su firma $f(q)$. Posteriormente se consultan los FHQTs involucrados. Por cada uno se obtienen dos listas: la lista de candidatos y la lista de objetos descartados. Finalmente se realiza la unión de dichas listas y al conjunto resultante se le quitan aquellos objetos que pertenecen a la lista de descartados, obteniendo de este modo el conjunto de objetos que 'sobrevivieron' a todos los filtros. Por último, se realiza la comparación de cada elemento de la lista definitiva de candidatos con la query q.

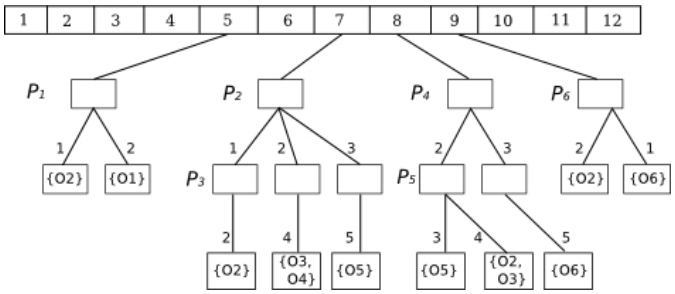

Figura 4.4: *New Historical-FHQT*

En la Figura 4.4 se presenta un ejemplo del New H-FHQT. En este caso se representa el intervalo de tiempo $[1, 12]$. Se graficaron objetos vigentes solo en los instantes 5, 7, 8 y 9. Se puede apreciar que los FHQTs tienen diferentes profundidades, esto depende de la cantidad de objetos que indexa cada FHQT.

Para la consulta $(q, 2, 4, 7)_d$, siendo (1, 2, 4, 3, 1, 2) la firma de q, el *New H-FHQT* se comporta de la siguiente manera: se acceden en forma directa los instantes 4, 5, 6 y 7. Los instantes 4 y 6 se ignoran ya que no poseen *FHQTs* asociados.

En el instante 5, como la firma de q contiene como primer elemento a 1, y el radio de la búsqueda es 2, se toman las ramas 1 (1-1 $\leq 2 \leq$ 1+1) y 2 (1-2 $\leq 2 \leq$ 1+2); los objetos resultantes en este paso son $\{O_1; O_2\}$.

En el instante 7, como la firma de q contiene como segundo y tercer elementos a 2 y 4, se toman las ramas 1 (2-1 $\leq 2 \leq$ 2+1), 2 (2-2 $\leq 2 \leq$ 2+2) y 3 (2-3 $\leq 2 \leq$ 2+3). En el siguiente nivel, se toman las ramas 2 (4-2 $\leq 2 \leq$ 4+2), 4 (4-4 $\leq 2 \leq$ 4+4) y 5 (4-5 $\leq 2 \leq$ 4+5). Los objetos resultantes son $\{O_2; O_3; O_4; O_5\}$.

En este caso la lista de descartados es vacía, por lo que se realiza la consulta por similitud sobre la lista de candidatos $\{O_1; O_2; O_3; O_4; O_5\}$.

En la Figura 4.5, se muestra el pseudocódigo del algoritmo de consulta al *New H-FHQT*. La función toma como entrada el objeto de consulta q, el radio de búsqueda r, y el intervalo que se consulta $[t_{iq}, t_{fq}]$. Como primer paso se calcula la firma de q para todos los pivotes utilizados en al menos un árbol del *New H-FHQT*; luego se consultan los *FHQT* correspondientes a cada uno de los instantes que conforman el intervalo consultado, se obtienen las listas de candidatos y descartados, se realiza la unión de las mismas y al resultado se le quitan los objetos que pertenecen a la lista de descartados. Finalmente se compara cada elemento de la lista definitiva de candidatos con la query q para obtener el conjunto resultado.

HFHQT $(q, r, t_{iq}, t_{fq})_d$
1. calcular la firma f de q - distancias a cada uno de los pivotes de l_p
2. candidatos:= $\emptyset$
3. descartados:= $\emptyset$
4. for all (fhqt h correspondiente a los instantes del intervalo $[t_{iq}, t_{fq}]$ del New H-FHQT)
5. candidatos := candidatos $\cup$ $h(q, r, f)_d$
6. descartados := descartados $\cup$ $no_h(q, r, f)_d$
7. candidatosfinal := candidatos $-$ descartados
8. resultado := x $\in$ candidatosfinal | d(q, x) $\leq$ r
9. return resultado

Observación 1: $h(q, r, f)_d$ devuelve el conjunto de objetos candidatos a ser similares a q con radio r a través de una consulta al *FHQT* h, donde f es la firma de q.
Observación 2: $no_h(q, r, f)_d$ devuelve el conjunto de objetos descartados por los pivotes del *FHQT* h.

Figura 4.5: *Pseudocódigo de consulta del New H-FHQT*

Algoritmo de inserción

En la Figura 4.6, se muestra el pseudocódigo del algoritmo de inserción del *New H-FHQT*. Para insertar un nuevo instante en el árbol, en primer lugar se calcula la cantidad de pivotes a utilizar. Luego se toma la cantidad de pivotes necesaria para indexar los objetos vigentes en el instante que se quiere insertar y se genera el FHQT correspondiente. Si en la construcción del FHQT se utiliza el último pivote de la lista, se debe ir al inicio de la misma y tomar el primer pivote para continuar la construcción del árbol. Una vez construído el FHQT se agrega el instante con su árbol asociado a la línea de tiempo válido.

Insercion New H-FHQT (unaListaDeObjetos, instante)

```
 1. cantPivotes = truncar(log(size(unaListaDeObjetos),2)) + 1.
 2. cantPivotes := cantPivotes/2.
 3. IF cantPivotes <1 THEN: [cantPivotes := 1 ].
 4. generarInstante(instante).
 5. pivotesAUsar:= nueva lista.
 6. 1 to: cantPivotes do:
 7.    IF ultimo(unPivote) THEN agregarPivote(pivotesAUsar at: 1).
 8.              ELSE pivotesAUsar add: (pivotes at: pivoteActual).
 9. fhqt := FHQT new()
10. 1 to: cantPivotes do: agregarPivote(pivotesAUsar at: unPivote).
11. unaListaDeObjetos do: agregarObjeto(unObjeto).
12. agregarInstante(lineaDelTiempo, instante).
```

Figura 4.6: *Pseudocódigo de inserción del New H-FHQT*

Capítulo 5

Evaluación Experimental

En este capítulo se presenta la evaluación experimental de los índices diseñados. Inicialmente se da una descripción de los experimentos para luego realizar el análisis de resultados. Todos los experimentos aquí presentados fueron obtenidos en un Athlon 64 X2 Dual Core Processor 4200, 2.21 GHz y 1 GB de RAM.

5.1. Descripción de los experimentos

Bases de datos utilizadas

Debido a que los espacios métrico-temporales son un nuevo modelo de bases de datos, el primer obstáculo que surgió al momento de realizar las evaluaciones experimentales fue la falta de lotes de prueba testigos adecuados para los índices propuestos. Por esta razón el primer paso consistió en generar estos lotes. Para ello se utilizaron dos bases de datos de imágenes vectorizadas ampliamente empleadas por la comunidad de espacios métricos: $COLORS$, que contiene vectores de 761 componentes, y $NASA$, que contiene vectores de 20 componentes, las que se encuentran disponibles en el sitio:

http://www.sisap.org/library/dbs/vectors/

A partir de estas bases métricas se generaron bases de datos métrico-temporales de la siguiente manera:

- a cada objeto se le asignó un identificador y un intervalo de vigencia que

indica el período de validez del objeto en la línea de tiempo válido (no transaccional). El intervalo total considerado fue [1, 1000].

- se generaron aleatoriamente lotes de tamaños 5.000, 10.000 y 15.000 tanto para $COLORS$ como para $NASA$.

Para cada base de datos se utilizó como función de distancia la distancia euclidiana. De aquí en adelante se hará referencia a las bases de datos métrico-temporales generadas como $ColorsMT$ y $NasaMT$.

Consultas realizadas

Para cada una de las 6 bases de datos creadas, se generaron 100 consultas por rango métrico-temporales que fueron obtenidas tomando aleatoriamente 100 elementos de la base considerada ($COLORS$ o $NASA$ según corresponda) y asignándole a cada uno de ellos los siguientes valores para intervalos de validez:

instantáneas, $10\,\%$, $25\,\%$ y $50\,\%$ (del intervalo$[1, 1000]$)

Esto originó 4 lotes de consulta para cada una de las bases de datos. Para cada uno de esos lotes de consulta se realizaron búsquedas métrico-temporales utilizando los siguientes valores para el radio de búsqueda r:

$$\begin{aligned} r &= 5, 9, 11 \text{ para las bases generadas a partir de } COLORS \\ r &= 7, 9, 11 \text{ para las bases generadas a partir de } NASA \end{aligned}$$

Estos radios de búsquedas retornan aproximadamente el $1\,\%$, $5\,\%$ y $10\,\%$ de objetos de las bases de datos y fueron establecidos experimentalmente realizando búsquedas exhaustivas sobre muestras aleatorias de cada base de datos.

Para cada consulta se tomó como función de costo la cantidad de evaluaciones de la función de distancia, dado que se está suponiendo que el índice completo se mantiene en memoria principal.

Creación de los índices

Los FHQTs correspondientes a cada instante de tiempo se construyeron tomando pivotes de una lista construida con elementos de la base de datos elegidos al

azar [BNC01]. En el caso del *H-FHQT* la cantidad de pivotes de cada árbol se calculó como $\lceil log_2(|O_i|) \rceil$, donde $|O_i|$ es la cantidad de objetos vigentes en el instante i. Para el *New H-FHQT* se generó una lista de pivotes cuya cardinalidad es un número natural que varía entre $\lceil log_2(|O_i|) \rceil$ y $\lceil log_2(|O_i|) \rceil * 2$.

5.2. Análisis de resultados para ColorsMT

Todas las figuras que se presentan a continuación fueron obtenidas promediando los resultados obtenidos con cada una de las 100 consultas realizadas. Por cuestiones de simplicidad sólo se muestran las gráficas que se consideran más representativas. Las gráficas de los experimentos restantes pueden consultarse en el apéndice.

Efecto del radio de búsqueda

La Figura 5.1 muestra los resultados obtenidos con la base de datos ColorsMT de 5.000 objetos con los índices *H-FHQT* y *New H-FHQT* para consultas instantáneas (arriba izquierda) y por intervalos. En el eje X de las gráficas se representan los radios de búsqueda y en el eje Y el número promedio de evaluaciones de distancia.

Como puede observarse, el índice *New H-FHQT* tiene un mejor desempeño que el *H-FHQT* logrando reducciones en la cantidad de evaluaciones de distancias de hasta el 12 % para la consulta instantánea. Esta mejora se acentúa más en el caso de las consultas por intervalos llegando a realizar hasta un 26 % menos de evaluaciones de distancia. En todos los casos los porcentajes de mejora disminuyen a medida que aumentamos el radio de búsqueda llegando a un 7 % en el caso de la consulta instantánea y a un 10 % en las consultas por intervalo.

Esta observación se puede apreciar más claramente en la Figura 5.2, en la misma cada barra representa el porcentaje de mejora del *New H-FHQT* respecto del *H-FHQT*, para cada radio y cada tipo de consulta considerado en los experimentos. Puede visualizarse claramente que a medida que se aumenta el radio de búsqueda el porcentaje de mejora disminuye, independientemente del intervalo consultado.

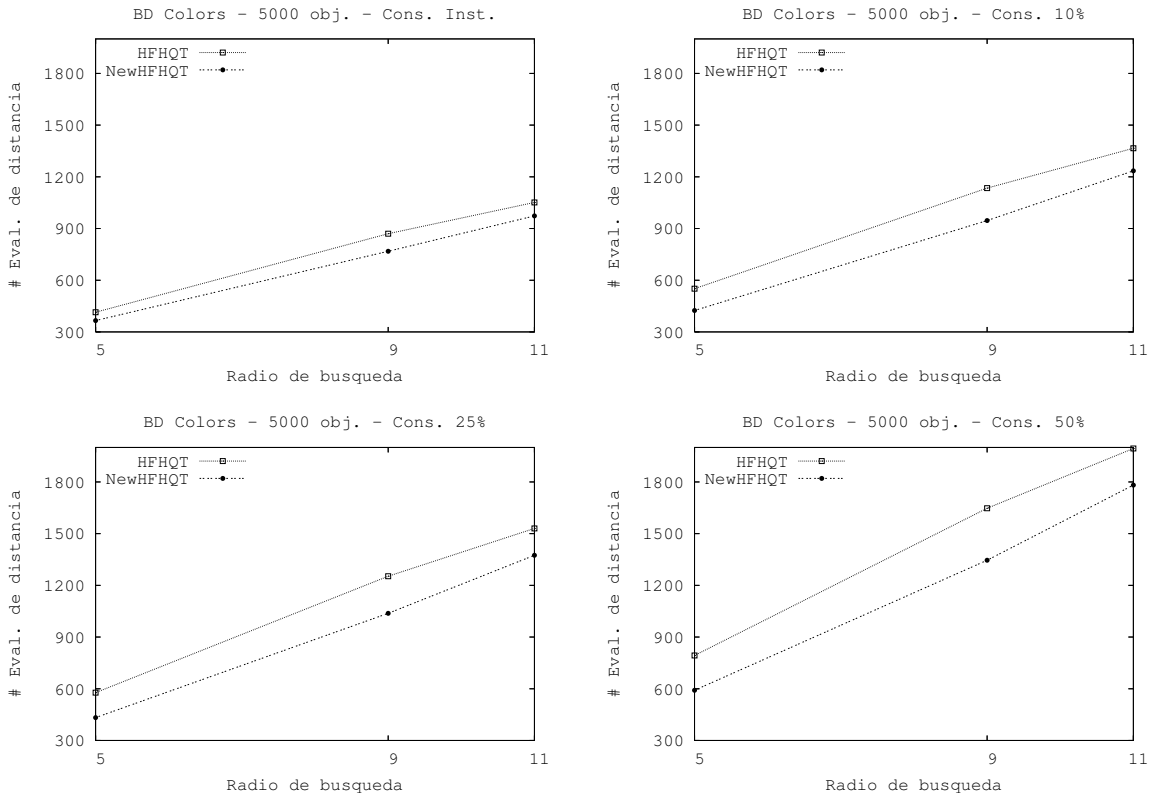

Figura 5.1: *Comparación de la cantidad promedio de evaluaciones de distancia para los índices H-FHQT y New H-FHQT para r = 5, 9 y 11 (BD ColorsMT de 5.000 objetos)*

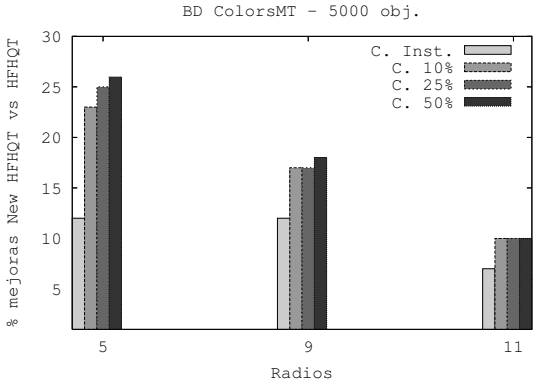

Figura 5.2: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para r = 5, 9 y 11 (BD ColorsMT de 5.000 objetos)*

Efecto de la amplitud del intervalo de consulta

Para mostrar gráficamente como influyen las variaciones de la amplitud del intervalo temporal de consulta sobre la cantidad de evaluaciones de la función de distancia se presenta la Figura 5.3. En el eje X de la misma se graficaron los diferentes intervalos temporales que se toman para realizar las consultas a los índices. En el eje Y se graficaron la cantidad promedio de evaluaciones de la función de distancia resultantes de la ejecución de 100 consultas a cada índice.

En esta gráfica se compara el desempeño de los dos índices en función de la variación de los intervalos temporales de consulta para los radios 5, 9 y 11, sobre la base ColorsMT con 5.000 elementos. En el caso de las consultas que toman el 10 % del intervalo temporal total se observan mejoras del New H-FHQT con respecto al H-FHQT de entre el 9 % y 23 %; para las consultas que toman el 25 % del intervalo total las mejoras del New H-FHQT respecto al otro índice se dan entre el 10 % y el 25 %; y para consultas por el intervalo 50 %, las mejoras varían entre el 10 % y el 26 %.

Para mostrar un panorama más general se presenta la Figura 5.4, en la que puede observase claramente la regularidad de las gráficas para todos los intervalos, lo que indica que la variación de los mismos no afecta los porcentajes de competitividad del *New H-FHQT* respecto al otro índice.

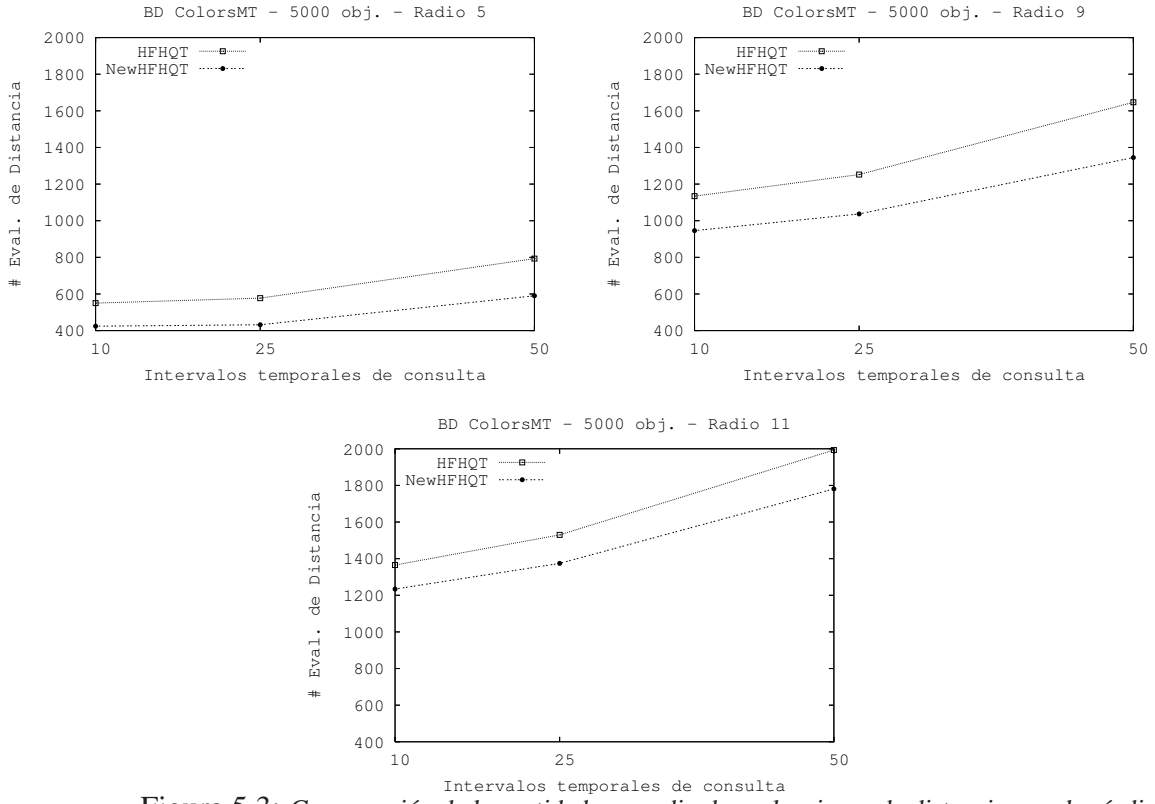

Figura 5.3: *Comparación de la cantidad promedio de evaluaciones de distancia para los índices H-FHQT y New H-FHQT para intervalos temporales de consulta 10 %, 25 % y 50 % (BD ColorsMT de 5.000 objetos)*

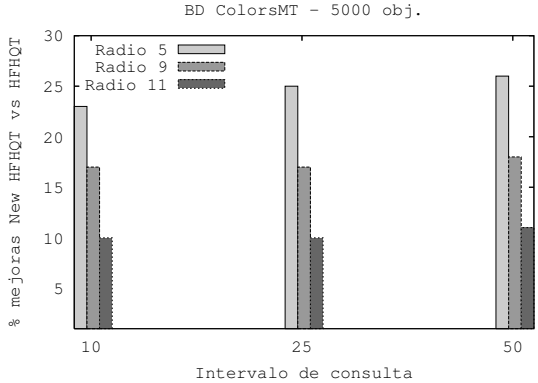

Figura 5.4: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para Consultas por Intervalo de 10 %, 25 % y 50 % (BD ColorsMT de 5.000 objetos)*

Efecto del tamaño de la base de datos

A continuación se presentan las Figuras 5.5 y 5.6 que muestran claramente los porcentajes de mejoras del *New H-FHQT* respecto al *H-FHQT* para consultas instantáneas y por intervalo del 10 % respectivamente, realizadas sobre la BD ColorsMT de 5.000 elementos. En todos los casos el *New H-FHQT* supera al otro índice, pero se observa que para consultas instantáneas las mejoras se dan en porcentajes menores que para las consultas por intervalo. En el primer caso los porcentajes varían entre 7 % y 12 % y en el segundo caso la variación se da entre el 10 % y el 23 %.

Las Figuras A.1 y A.2 (ver Apéndice) presentan los resultados obtenidos para las bases de datos ColorsMT de 10.000 y 15.000 objetos respectivamente. Para las consultas por intervalo en todos los casos las mejoras varían entre un 9 % y un 25 %, independientemente del tamaño de base de datos consultado. También puede observarse que los porcentajes de mejoras disminuyen en la medida en que se incrementa el radio de búsqueda. Esto indica que el comportamiento de los índices se mantiene a pesar del aumento del tamaño de la bases de datos.

Cuando se realiza el análisis de las consultas instantáneas se ve que el incremento en el tamaño de las bases de datos hace que, si bien persiste la competitividad del *New H-FHQT*, los porcentajes de mejoras con respecto al *H-FHQT* son menores. Pero se ve que para ambos tamaños de la base de datos (10.000 y 15.000) los porcentajes se mantienen invariables: 10 % para $r = 7$, 9 % para $r = 9$ y el 6 % para $r = 11$.

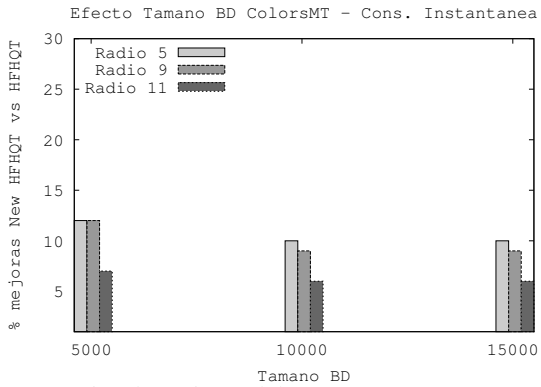

Figura 5.5: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para Consultas Instantáneas para distintos tamaños de la base de datos ColorsMT*

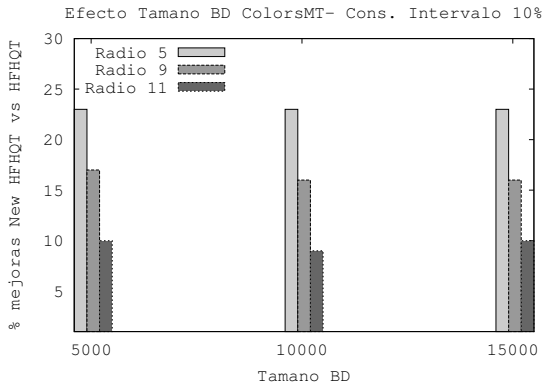

Figura 5.6: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para Consultas por Intervalo 10 % para distintos tamaños de la base de datos ColorsMT*

5.3. Análisis de resultados para NasaMT

De igual modo que en la Sección anterior, las figuras que se presentan a continuación fueron obtenidas promediando los resultados obtenidos con cada una de las 100 consultas realizadas. Por cuestiones de simplicidad sólo se muestran las gráficas que se consideran más representativas. Las gráficas de los experimentos restantes pueden consultarse en el apéndice.

Efecto del radio de búsqueda

La Figura 5.7 muestra los resultados obtenidos con la base de datos NasaMT de 5.000 objetos con los índices *H-FHQT* y *New H-FHQT* para consultas instantáneas (arriba izquierda) y por intervalos. En el eje X de las gráficas se representan los radios de búsqueda y en el eje Y el número promedio de evaluaciones de distancia.

En este caso se puede apreciar que para consultas instantáneas el *New H-FHQT* solo supera al *H-FHQT* cuando $r = 7$, logrando una mejora del 1 %. Cuando $r = 9$ y $r = 11$ en cambio, el *H-FHQT* es mas competitivo que el *New H-FHQT*, consiguiendo mejoras del 2 % y 3 % respectivamente. Se observa para estos dos casos que los porcentajes de mejoras se incrementan a medida que aumenta también el radio de búsqueda.

En cambio cuando se analizan los resultados para las consultas por intervalo se ve que el índice *New H-FHQT* es el más eficiente en todos los casos. Las mejoras varían entre el 2 % y el 14 %, decreciendo a medida que se incrementa el radio de búsqueda.

El comportamiento de los índices para consultas por intervalo se evidencia más claramente en la Figura 5.8; en la misma cada barra representa el porcentaje de mejora del *New H-FHQT* respecto del *H-FHQT*, para cada radio de búsqueda y cada tipo de consulta considerados en los experimentos. Aquí se ve también que a medida que aumentamos el radio de búsqueda el porcentaje de mejora disminuye, independientemente del intervalo consultado.

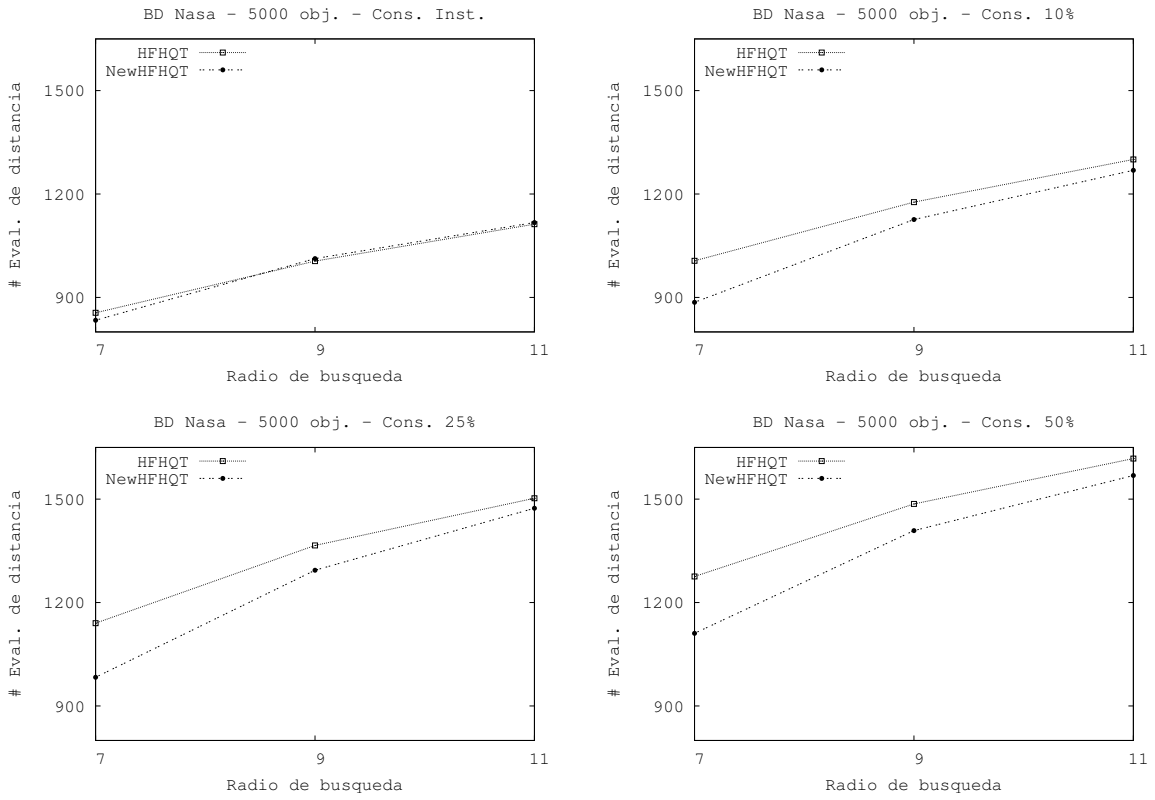

Figura 5.7: *Comparación de la cantidad promedio de evaluaciones de distancia para los índices H-FHQT y New H-FHQT para r = 7, 9 y 11 (BD NasaMT de 5.000 obj.)*

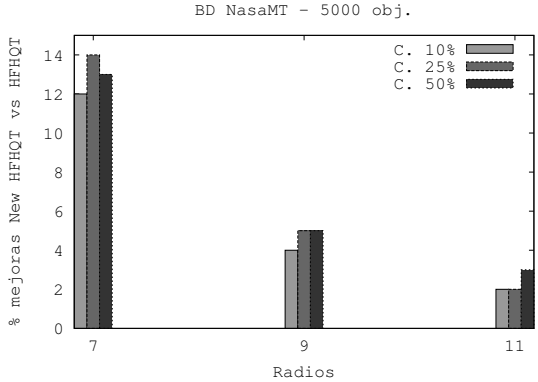

Figura 5.8: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para r = 7, 9 y 11 (BD NasaMT de 5.000 objetos)*

Efecto de la amplitud del intervalo de consulta

En la Figura 5.9 se compara el desempeño de los dos índices en función de la variación de los intervalos temporales de consulta para los radios 7, 9 y 11, sobre la base NasaMT con 5.000 elementos. En el eje X de la misma se graficaron los diferentes intervalos temporales que se toman para realizar las consultas y en el eje Y la cantidad promedio de evaluaciones de la función de distancia.

En el caso de las consultas que toman el 10 % del intervalo temporal total se observan mejoras del *New H-FHQT* con respecto al *H-FHQT* de entre el 2 % y el 12 %; para las consultas que toman el 25 % del intervalo total el *New H-FHQT* mejora al otro índice entre el 2 % y el 14 %; y para consultas por el intervalo 50 %, las mejoras varían entre el 3 % y el 13 %. Los porcentajes mayores corresponden a las consultas realizadas cuando $r = 7$ y los menores a $r = 11$.

Para mostrar un panorama más general se presenta la Figura 5.10, en la que puede observase claramente la regularidad de las gráficas para todos los intervalos, lo que indica que la variación de los mismos no afecta los porcentajes de competitividad del *New H-FHQT* respecto al otro índice.

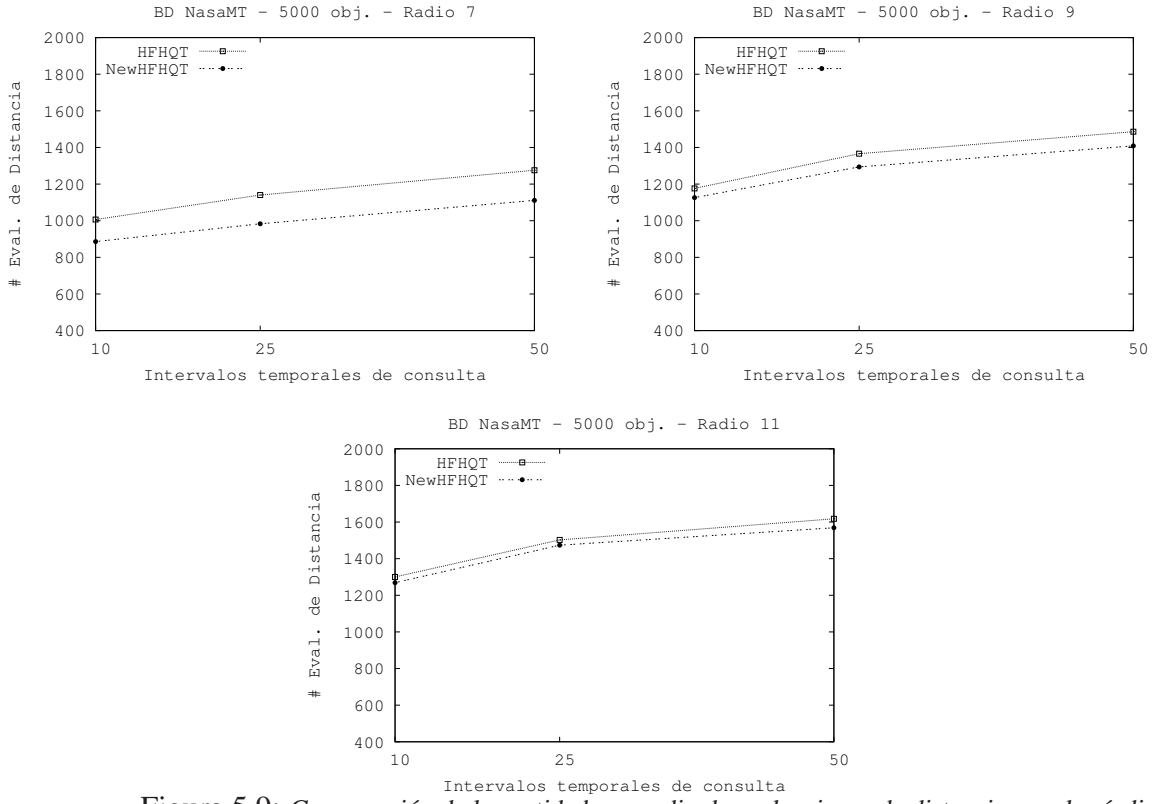

Figura 5.9: *Comparación de la cantidad promedio de evaluaciones de distancia para los índices H-FHQT y New H-FHQT para intervalos de consulta 10 %, 25 % y 50 % (BD NasaMT de 5.000 objetos)*

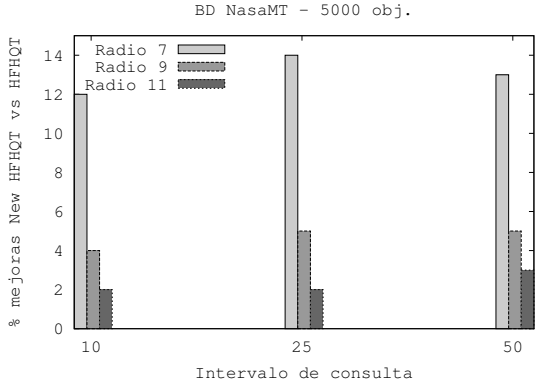

Figura 5.10: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para Consultas por Intervalo de 10 %, 25 % y 50 % (BD NasaMT de 5.000 objetos)*

Efecto del tamaño de la base de datos

A continuación se presenta la Figura 5.11 que muestra el comportamiento de los índices para las consultas instantáneas cuando se varía el tamaño de la base de datos. En la gráfica se puede apreciar que el índice *New H-FHQT* solo mejora al *H-FHQT* cuando $r = 7$, alcanzando mejoras de entre el 1 % y 2 %. Para los radios mayores el *H-FHQT* resulta ser más competitivo, alcanzando mejoras que varían entre 1 % y 3 %.

La Figura 5.12 presenta los porcentajes de mejoras del *New H-FHQT* respecto al otro índice para las consultas por intervalo de 10 %. Para todos los tamaños de la base de datos NasaMT se observa que se mantienen los porcentajes de mejoras prácticamente invariables. Dichos porcentajes varían entre el 1 % y el 12 %, decreciendo a medida que se incremente el radio de consulta. Los gráficos para los restantes intervalos de consulta pueden encontrarse en el Apéndice.

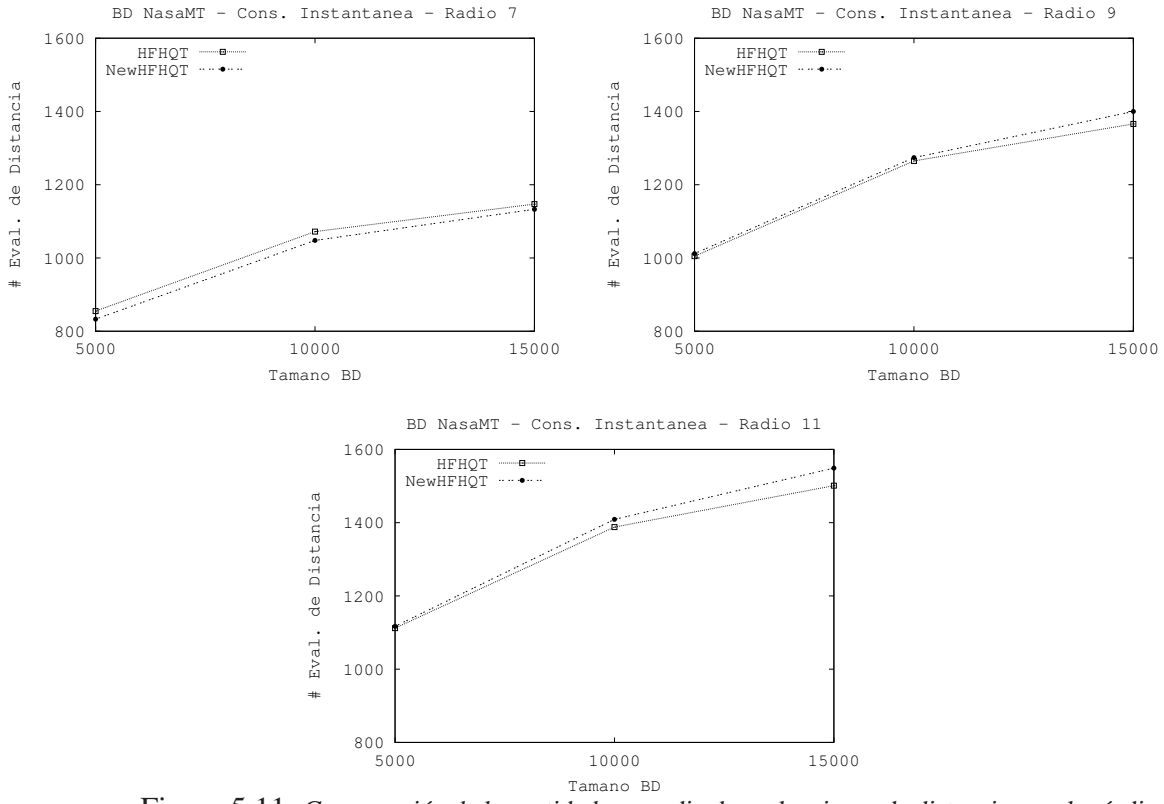

Figura 5.11: *Comparación de la cantidad promedio de evaluaciones de distancia para los índices H-FHQT y New H-FHQT para Consultas Instantáneas variando los tamaños de la BD NasaMT*

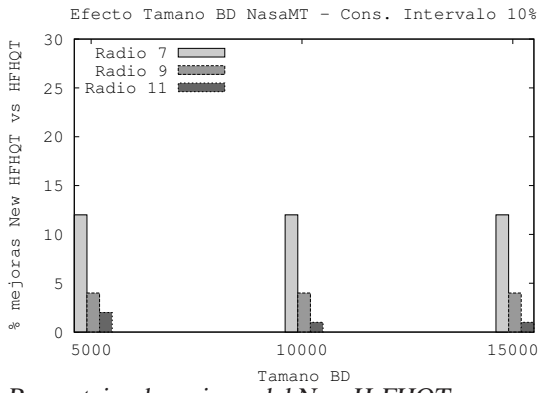

Figura 5.12: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para Consultas por Intervalo 10 % (BD NasaMT)*

5.4. Conclusiones

En este Capítulo se presentó la evaluación experimental de los índices métrico-temporales *H-FHQT* y *New H-FHQT* sobre las bases de datos ColorsMT y NasaMT de 5.000, 10.000 y 15.000 elementos. A partir de las evaluaciones se pudo apreciar que en general el índice *New H-FHQT* resultó ser el más competitivo.

En primer lugar se analizó el efecto que el radio de búsqueda tiene sobre las mejoras que presenta dicho índice respecto al *H-FHQT*. En este sentido se observó que para las consultas por intervalo los porcentajes de mejoras varían entre el 9 % y el 26 % para las pruebas realizadas con la base de datos ColorsMT y entre el 1 % y el 14 % para la base NasaMT.

En el caso de las consultas instantáneas, para la base ColorsMT el índice *New H-FHQT* es más eficiente para todos los radios de búsquedas y las mejoras oscilan entre el 6 % y el 12 %. En cambio, en las pruebas realizadas con la base de datos NasaMT el *New H-FHQT* solo fue superior cuando $r = 7$ alcanzando mejoras del 1 % y 2 %. Cuando $r = 9$ y $r = 11$ el *H-FHQT* realizó menor cantidad de evaluaciones de la función de distancia, mejorando al otro índice entre un 1 % y 3 %. Además se pudo observar que los porcentajes de mejoras tienden a disminuir a medida que se producen incrementos en el radio de búsqueda.

Cuando se analizó la influencia de la variación de la amplitud del intervalo temporal de consulta sobre el comportamiento de los índices, se vió que los porcentajes de mejoras del *New H-FHQT* sobre el otro índice se mantienen prácticamente invariables para cada uno de los radios y de los tamaños de cada bases de datos considerada. En el caso de la base ColorsMT los porcentajes de mejoras están entre un 9 % y un 26 % y para la base NasaMT las mejoras varían entre el 1 % y el 14 %.

Al analizar los resultados de las evaluaciones experimentales teniendo en cuenta las variaciones en el tamaño de la base de datos ColorsMT no se detectaron modificaciones en los porcentajes de mejoras para las consultas por intervalos, que se encuentran entre el 23 % y 25 % para el radio menor y entre 9 % y 11 % para el radio de búsqueda mayor; en cambio, en las consultas instantáneas, se vió que dichos porcentajes, que varían entre el 6 % y el 12 %, presentan una tendencia decreciente a medida que aumenta el tamaño de la base.

En el caso de la base NasaMT para las consultas instantáneas el índice *New H-FHQT* sólo superó al *H-FHQT* para el menor radio de búsqueda, obteniendo mejoras de entre el 1 % y 2 %. Para los otros radios de búsqueda el *H-FHQT* logró

mejoras entre el 1 % y 3 %. En las consultas por intervalo el índice *New H-FHQT* resultó ser más competitivo, alcanzando porcentajes de mejoras entre el 12 % y 14 % para el menor radio y entre el 1 % y 3 % para el radio mayor.

Los porcentajes reales en los que el índice *New H-FHQT* mejoró al otro método son: para el caso de las consultas instantáneas en que el *New H-FHQT* fue más eficiente se pudo observar que se requirió un 6 % menos de evaluaciones de distancia que con el otro índice.

Para las consultas por intervalo se vió que para el radio menor el *New H-FHQT* realizó un 19 % menos de evaluaciones de distancia, para el radio medio el 11 % menos y para el radio mayor, fueron necesarias un 6 % menos de evaluaciones de la función de distancia respecto al índice *H-FHQT*.

Por lo tanto se puede concluir que para las consultas por intervalo en la totalidad de los casos evaluados el *New H-FHQT* resultó ser el índice más eficiente y para las consultas instantáneas lo fue en el 66 % de los casos.

Capítulo 6

Conclusiones y Trabajo Futuro

En este trabajo se abordó el estudio de aquellas consultas a bases de datos de objetos no estructurados que involucran tanto aspectos métricos como temporales. Como hasta la actualidad no existían herramientas teóricas que permitieran el análisis y resolución de este tipo de búsquedas, la primer labor de esta tesis consistió en generar un marco teórico para sustentar el abordaje de la temática. Los aportes fundamentales de esta tesis son:

1. Definición de un nuevo modelo de bases de datos: los Espacios Métrico-Temporales. Con esto se generó el marco necesario para realizar el análisis de aquellas consultas que involucran a la vez aspectos métricos y temporales.

2. Definición del concepto de Consulta Métrico-Temporal. En particular se definieron las búsquedas por rango en el campo de los Espacios Métrico-Temporales.

3. Definición de dos métodos de acceso Métrico-Temporales: *Historical-FHQT (H-FHQT)* y *New Historical-FHQT (New H-FHQT)*.

Con respecto al primer punto se definió el modelo formal y se presentaron las características que poseen aquellas aplicaciones que requieren de este modelo para ser evaluadas. Luego se definieron las consultas por rango métrico-temporales, en las que se basó principalmente el estudio y la definición de estructuras en esta tesis. Posteriormente se plantearon la estructura y la modalidad de indexación y consulta para los métodos de acceso métrico-temporales antes mencionados.

El primer índice definido, el H-FHQT, consiste en una lista de instantes de tiempo válido donde cada elemento de dicha lista tiene un FHQT asociado que indexa los objetos vigentes para ese instante. Este índice se comparó con la solución trivial y pudo comprobarse de manera experimental que logró disminuir significativamente la cantidad de evaluaciones de la función de distancia (realiza entre un 45 % y 65 % menos de evaluaciones de distancia que la solucion trivial para consultas por intervalo y entre un 92 % y 98 % menos para consultas instantáneas), factor que se tomó en todos los casos como función de costo de los experimentos realizados.

El segundo índice, el *New H-FHQT*, consiste en una lista de instantes de tiempo válidos, y cada uno de esos instantes tiene asociado un FHQT que indexa los objetos vigentes correspondientes. A diferencia de su antecesor, en este caso los FHQTs se construyen utilizando una lista con mayor cantidad de pivotes, asegurando que al menos tres árboles consecutivos se conformen con diferentes grupos de pivotes, incrementando así la posibilidad de filtrar elementos que no sean candidatos a formar parte del conjunto resultante.

Al analizar los resultados de las evaluaciones experimentales pudo comprobarse que el índice *New H-FHQT* es más competitivo que su antecesor en la totalidad de las consultas por intervalo ejecutadas sobre las bases ColorsMT y NasaMT. Para las consultas instantáneas se vió que para la base ColorsMT dicho índice superó en todos los casos al *H-FHQT* y para la base NasaMT el índice *New H-FHQT* fue más competitivo en el 66 % de las pruebas . Las mejoras observadas en este índice se deben al mayor poder de filtrado que se logró generando los FHQTs consecutivos con diferentes grupos de pivotes, así un objeto que tiene un período de vigencia asociado recorre más filtros que en el índice *H-FHQT* y por lo tanto hay mayor probabilidad de que sea eliminado de la lista de candidatos que se comparan directamente con la consulta, eliminando así la cantidad de evaluaciones de la función de distancia.

A partir de este trabajo de tesis han surgido algunas líneas de investigación que se consideran interesantes para continuar en futuros trabajos de investigación. Las mismas se describen a continuación:

- Se sabe que la dimensionalidad de un espacio métrico afecta el desempeño de los índices. En bases de datos métrico-temporales podría suceder que la dimensión de un conjunto de elementos en el instante i sea distinta a la dimensión del conjunto de elementos en otro instante j y en ese caso las decisiones tomadas con respecto a la construcción del índice deberían variar de un instante a otro. Por esta razón, un aspecto interesante a estudiar es el

concepto de dimensionalidad aplicado a bases de datos métrico-temporales con el fin de encontrar una definición que se adecue a este nuevo modelo de bases de datos y que permita comprender mejor el desempeño de los índices.

- En base al punto anterior, se puede diseñar un índice híbrido que permita tener distintos índices métricos en distintos instantes de tiempo, según sea la dimensionalidad del conjunto de elementos almacenados en cada instante.

- Este trabajo se desarrolló bajo el supuesto de que la memoria principal tiene capacidad suficiente como para mantener tanto el índice como la base de datos. Habiendo diseñado índices competitivos en memoria principal, el próximo paso es adecuarlos para que los mismos también resulten eficiente en memoria secundaria. En este sentido nos proponemos encontrar un método de paginación para el *New H-FHQT* para que el mismo siga siendo competitivo en memoria secundaria. En este caso la función de costo utilizada será no sólo la cantidad de evaluaciones de distancias sino también la cantidad de accesos a disco realizados para resolver una búsqueda.

- Tanto el *H-FHQT* como el *New-FHQT* son paralelizables dado que la búsqueda en el *FHQT* del instante i es independiente de la búsqueda en el *FHQT* del instante j. Esto es una característica importante principalmente para las consultas por intervalos y que exploraremos en el futuro.

- Otro aspecto interesante a estudiar es el referido al espacio necesario para mantener el índice, dado que esto decide si el índice se mantendrá en memoria principal o en memoria secundaria. Una forma de reducir el espacio utilizado por el *New H-FHQT* es tratar de reutilizar subárboles: si un subárbol del instante i está también en el instante j (con $j > i$), entoces el instante j debería reutilizar el subárbol del instante i en lugar de crearlo de nuevo. Esto implica diseñar un algoritmo que permita detectar subárboles isomorfos.

- Posiblemente la reutilización de subárboles afecta la paralelización del índice, por lo tanto se hará necesario estudiar si el beneficio que se obtiene al reutilizar subárboles se compensa frente a la posible pérdida de paralelización de las consultas.

- Finalmente, en las aplicaciones en las que el modelo métrico-temporal tiene interés, existen otros tipos de consultas que resultan interesantes, tales como:
  - la búsqueda del vecino más cercano (*encontrar la fotografía más similar a una dada cuya vigencia corresponda a un intervalo de tiempo*)
  - la búsqueda de los k-vecinos más cercanos (*encontrar las k fotografías*

más similares a una dada cuya vigencia corresponda a un intervalo de tiempo),
- consultas instantáneas puras (*encontrar todas las fotografías vigentes en un instante de tiempo*),
- o consultas por clave (*encontrar las diferentes fotografías de una persona a lo largo del tiempo*).

Para cada una de ellas se hará necesario estudiar si los índices propuestos en este trabajo las pueden resolver o si son necesarios cambios en el diseño de los mismos.

Bibliografía

[Ben75] Jon Louis 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.

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

[BKpK96] Stefan Berchtold, Daniel A. Keim, and Hans peter 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).

[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] Ricardo A. Baeza-Yates, Walter Cunto, Udi Manber, and Sun 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.

[Cha94] Bernard 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.

[CHCR05] E. Chávez, N. Herrera, and A. Villegas C. Ruano. Una implementación completa del fqtrie. In *VII Workshop de Investigadores de Ciencias de la Computación*, pages 61–65, 2005.

[CMBY99] E. Chávez, J. Marroquín, and R. Baeza-Yates. Spaghettis: an array based algorithm for similarity queries in metric spaces. In *Proc. String Processing and Information Retrieval (SPIRE'99)*, pages 38–46. IEEE CS Press, 1999.

[CMN99] E. Chávez, J. Marroquín, and G.Ñavarro. 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.

[CNBYM01a] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, 2001.

[CNBYM01b] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Proximity searching in metric spaces. *ACM Computing Surveys*, 2001.

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

[CT85] James Clifford and Abdullah Uz Tansel. On an algebra for historical relational databases: two views. *SIGMOD Rec.*, 14(4):247–265, 1985.

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

[DGK+94] Curtis Dyreson, Fabio Grandi, Wolfgang Käfer, Nick Kline, Nikos Lorentzos, Yannis Mitsopoulos, Angelo Montanari, Daniel

Nonen, Elisa Peressi, Barbara Pernici, John F. Roddick, Nandlal L. Sarda, Maria Rita Scalas, Arie Segev, Richard Thomas Snodgrass, Mike D. Soo, Abdullah Tansel, Paolo Tiberio, and Gio Wiederhold. A consensus glossary of temporal database concepts. *SIGMOD Rec.*, 23(1):52–64, 1994.

[DPGH07] A. De Battista, A. Pascal, G. Gutierrez, and N. Herrera. Un nuevo índice métrico-temporal: el historical fhqt. In *Actas del XIII Congreso Argentino de Ciencias de la Computación*, Corrientes, Agentina, 2007.

[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.

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

[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.

[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.

[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.

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

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

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

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

[ST99] Betty Salzberg and Vassilis J. 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. 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.

[Uhl91] J. Uhlmann. Implementing metric trees to satisfy general proximity/similarity queries. *Information Processing Letters*, 40:175–179, 1991.

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

[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 del radio de búsqueda

En esta sección se muestra el análisis de la influencia de las variaciones del radio de búsqueda sobre las consultas instantáneas y por intervalo para las bases de datos ColorsMT de 10.000 y 15.000 objetos (Figuras A.1 y A.2) y también para NasaMT de 10.000 y 15.000 objetos (Figuras A.3 y A.4)

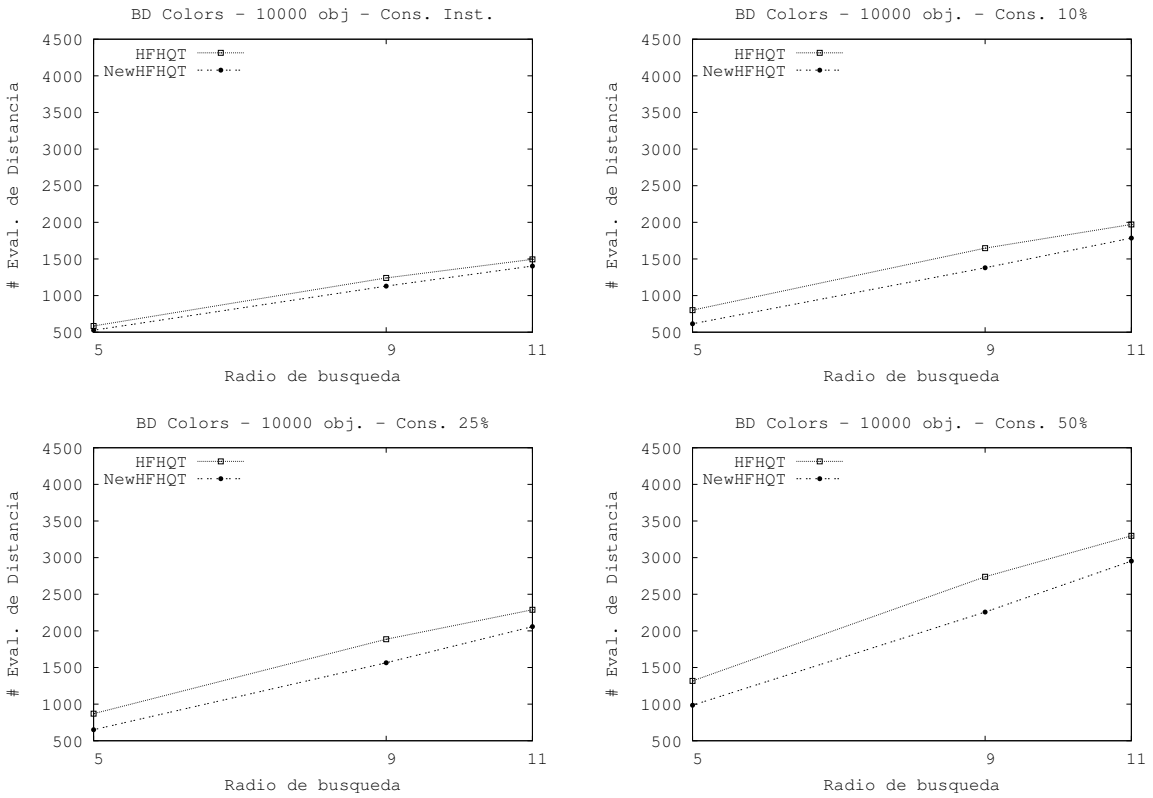

Figura A.1: *Cantidad promedio de evaluaciones de la función de distancia sobre la Base de Datos ColorsMT con 10.000 objetos.*

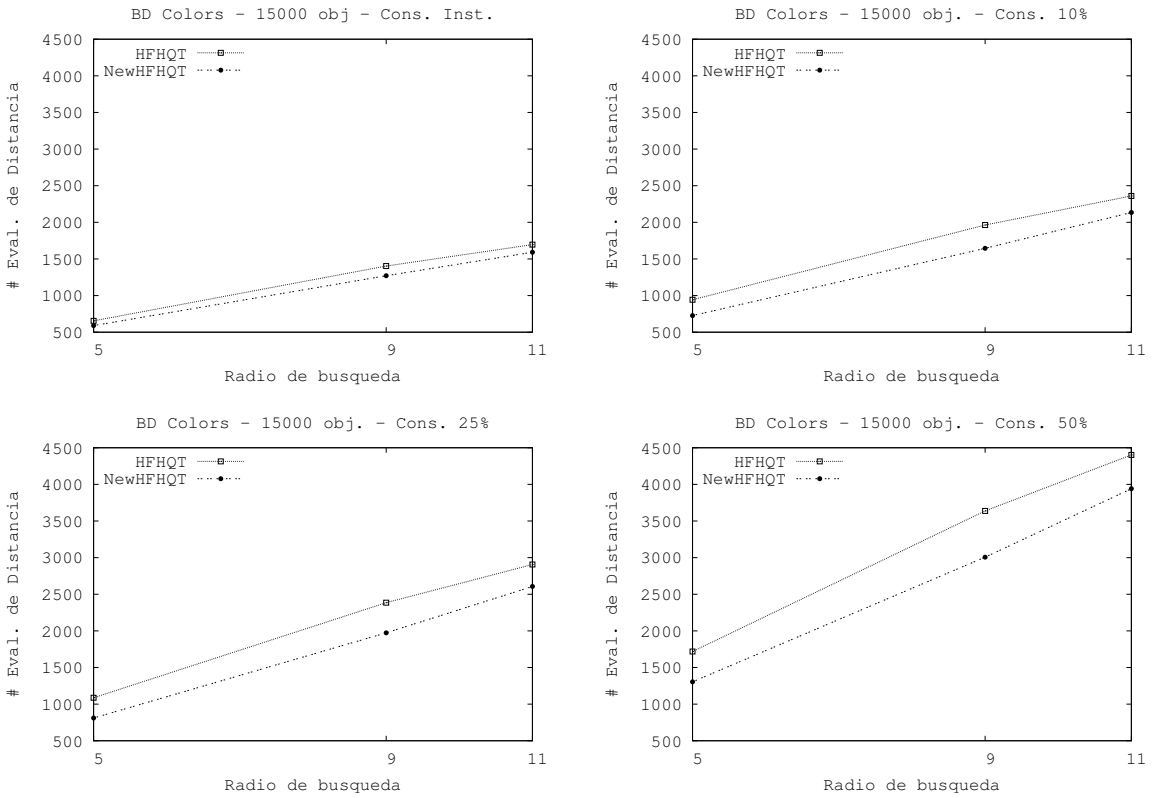

Figura A.2: *Cantidad promedio de evaluaciones de la función de distancia sobre la Base de Datos ColorsMT con 15.000 objetos.*

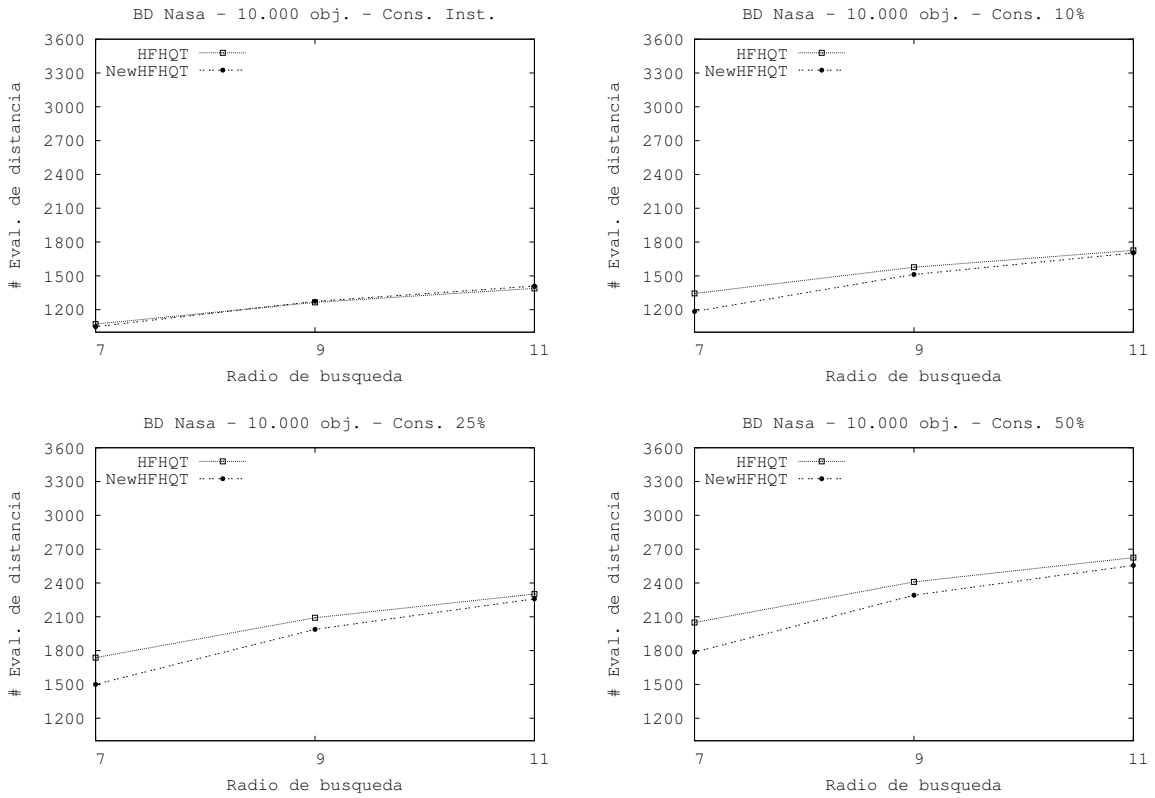

Figura A.3: *Cantidad promedio de evaluaciones de la función de distancia sobre la Base de Datos NasaMT con 10.000 objetos.*

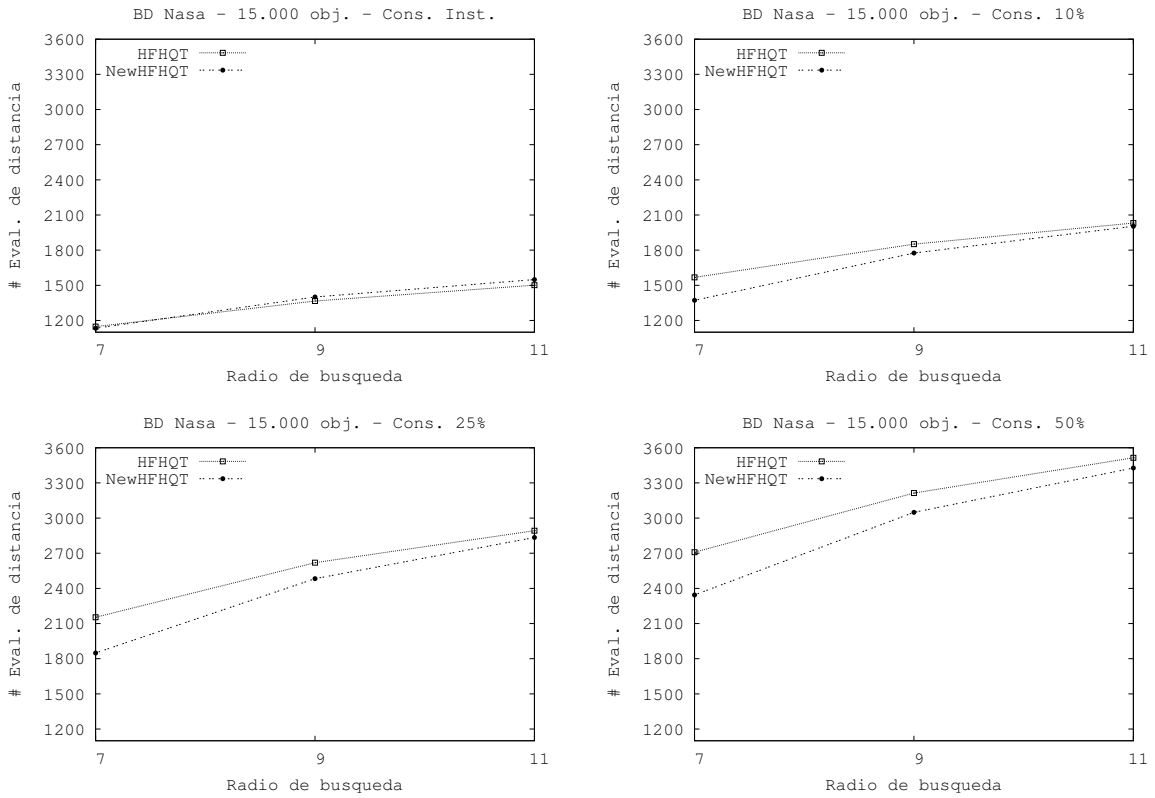

Figura A.4: *Cantidad promedio de evaluaciones de la función de distancia sobre la Base de Datos NasaMT con 15.000 objetos.*

A.2. Efecto de la amplitud del intervalo de consulta

En esta sección se presentan los resultados experimentales realizados para verificar la influencia de modificar el intervalo de consulta para las bases de datos ColorsMT de 10.000 y 15.000 objetos (Figuras A.5 y A.6) y NasaMT de 10.000 y 15.000 objetos (Figuras A.7 y A.8)

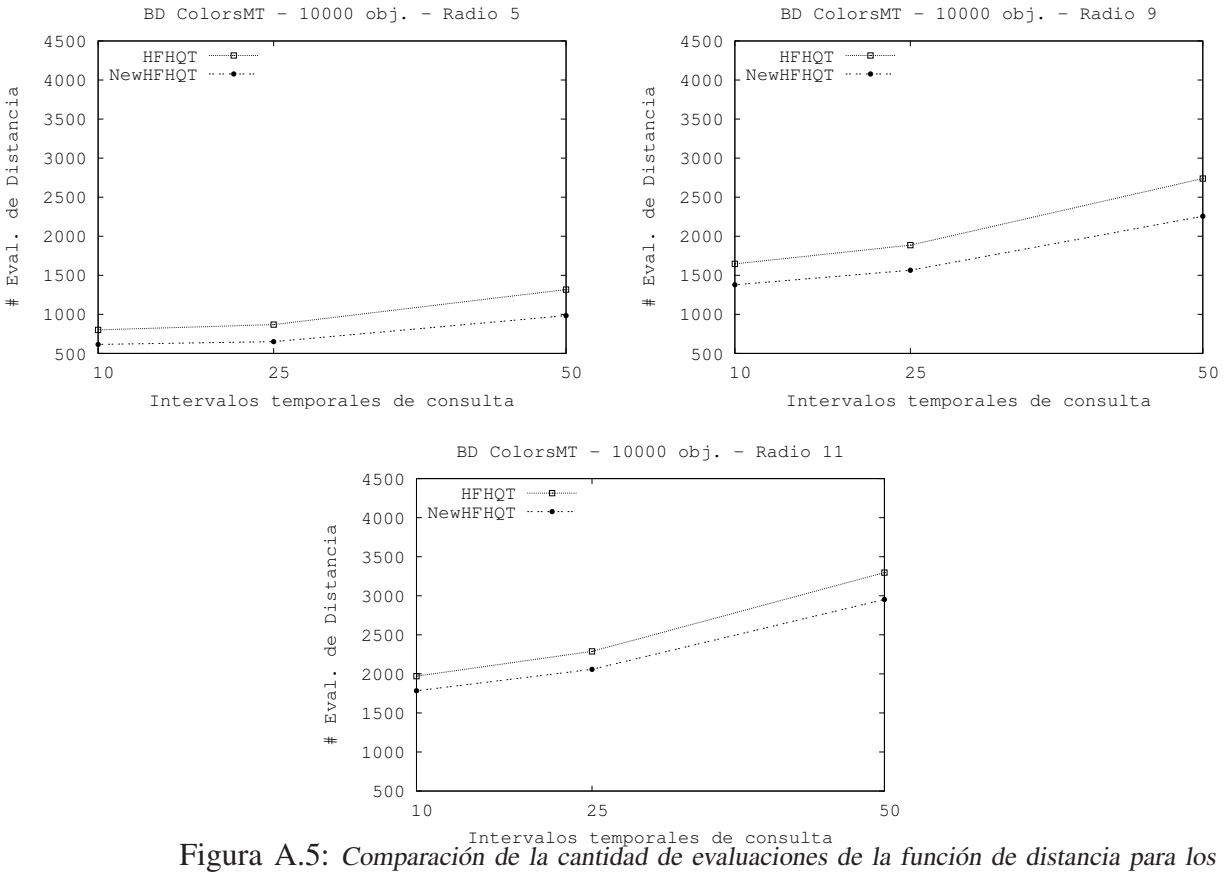

Figura A.5: *Comparación de la cantidad de evaluaciones de la función de distancia para los índices H-FHQT y New H-FHQT sobre la BD ColorsMT de 10.000 objetos con intervalos temporales de consulta 10 %, 25 % y 50 %*

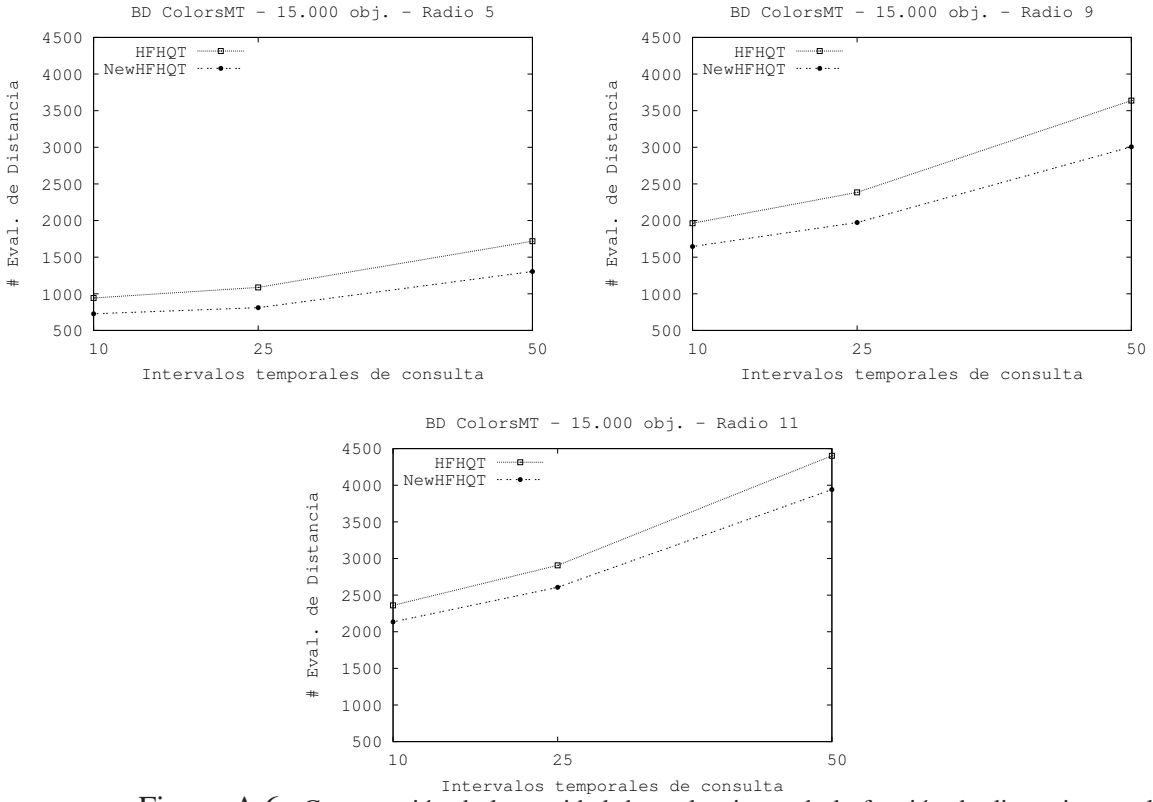

Figura A.6: *Comparación de la cantidad de evaluaciones de la función de distancia para los índices H-FHQT y New H-FHQT sobre la BD ColorsMT de 15.000 objetos con intervalos temporales de consulta 10 %, 25 % y 50 %*

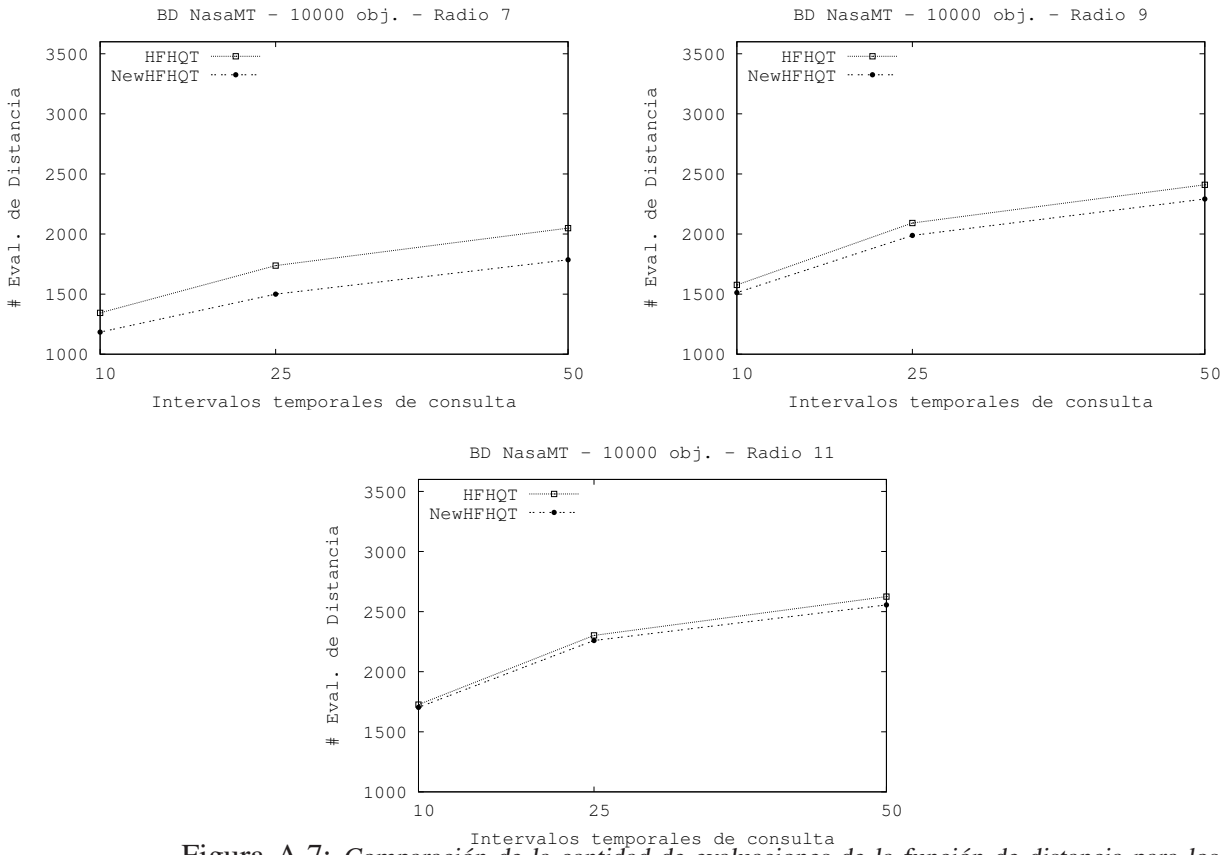

Figura A.7: *Comparación de la cantidad de evaluaciones de la función de distancia para los índices H-FHQT y New H-FHQT sobre la BD NasaMT de 10.000 objetos con intervalos temporales de consulta 10 %, 25 % y 50 %*

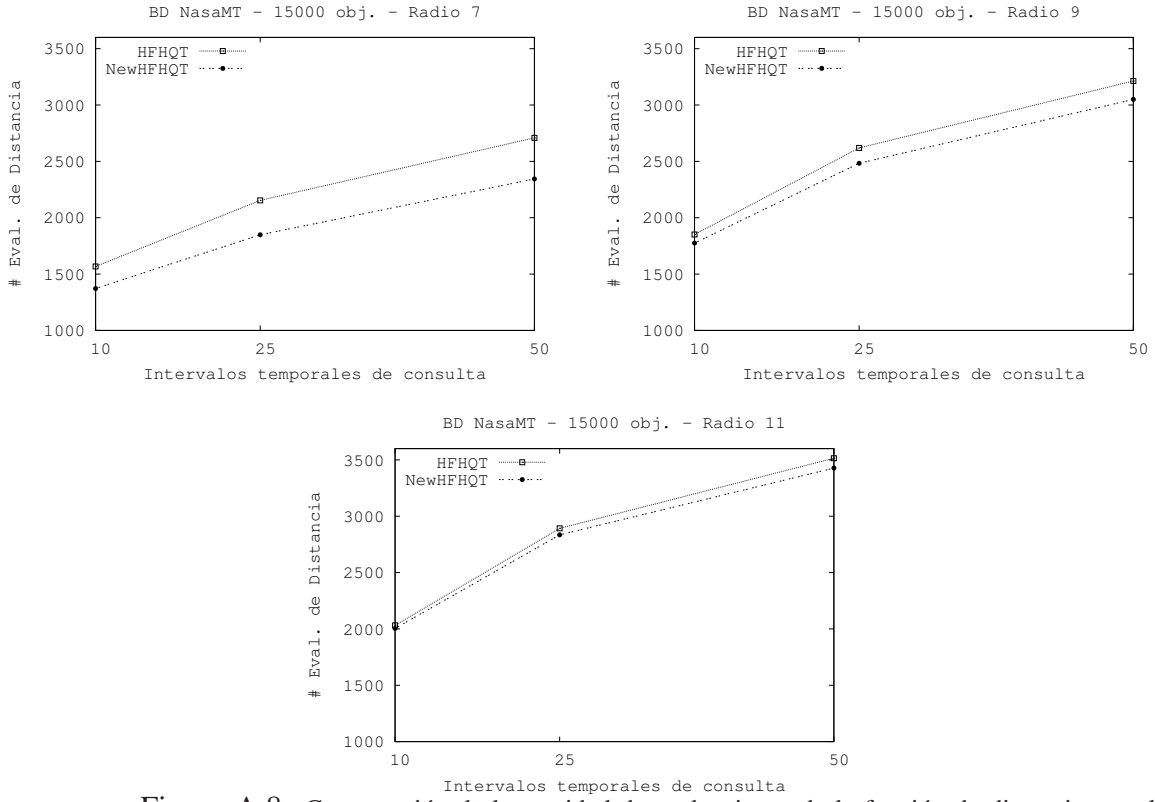

Figura A.8: *Comparación de la cantidad de evaluaciones de la función de distancia para los índices H-FHQT y New H-FHQT sobre la BD NasaMT de 15.000 objetos con intervalos temporales de consulta 10 %, 25 % y 50 %*

A.3. Efecto del tamaño de la base de datos

Las Figuras A.9 y A.10 reflejan los porcentajes de mejoras del *New H-FHQT* comparado con el *H-FHQT* para consultas por los intervalos 25 % y 50 %, para los radios 5, 9 y 11 sobre la base de datos ColorsMT. Como puede apreciarse la modificación en el tamaño de la base de datos no hace variar los porcentajes de mejoras alcanzados por el *New H-FHQT*.

Las Figuras A.11 y A.12 reflejan los porcentajes de mejoras del *New H-FHQT* comparado con el *H-FHQT* para consultas por los intervalos 25 % y 50 %, para los radios 7, 9 y 11 sobre la base de datos NasaMT. En este caso tampoco produce modificaciones en el comportamiento de los índices la variación del tamaño de la base de datos.

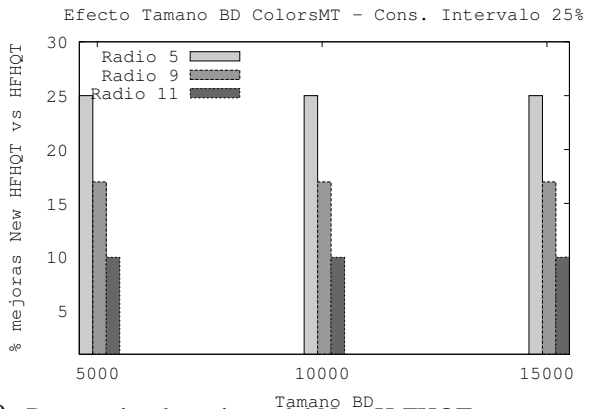

Figura A.9: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para Consultas por Intervalo 25 % para distintos tamaños de la base de datos ColorsMT*

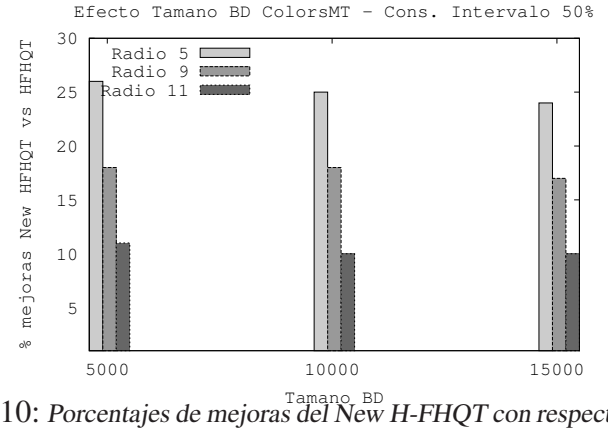

Figura A.10: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para Consultas por Intervalo 50 % para distintos tamaños de la base de datos ColorsMT*

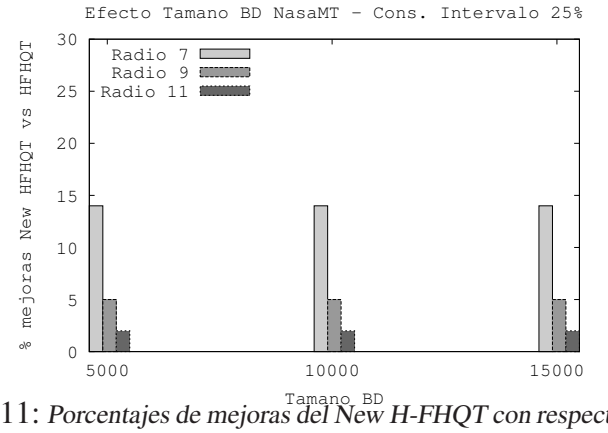

Figura A.11: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para Consultas por Intervalo 25 % para distintos tamaños de la base de datos NasaMT*

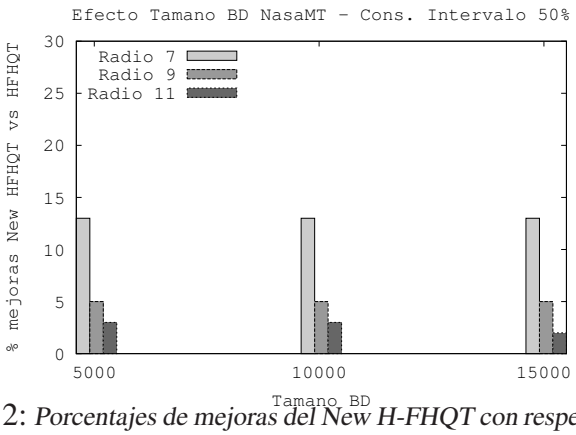

Figura A.12: *Porcentajes de mejoras del New H-FHQT con respecto al H-FHQT para Consultas por Intervalo 50 % para distintos tamaños de la base de datos NasaMT*