Indexando Datos no Estructurados

Gonzalo Navarro
Departamento de Ciencias de la Computación
Universidad de Chile, Chile
gnavarro@dcc.uchile

Nieves Rodríguez Brisaboa
Facultad de Informática
Universidad de A Coruña, España
brisaboa@udc.es

Norma Herrera, Carina Ruano, Darío Ruano, Ana Villegas, Susana Esquivel
Departamento de Informática
Universidad Nacional de San Luis, Argentina
{nherrera, cmruano, dmruano, anaville, esquivel}@unsl.edu.ar

Resumen

La próxima generación de administradores de bases de datos deberá ser capaz de indexar datos no estructurados (datos multimedia) y responder consultas sobre estos datos con tanta eficiencia como actualmente responden consultas de búsqueda exacta sobre bases de datos relacionales. Si bien existen numerosas técnicas de indexación diseñadas para esta problemática, mejorar la eficiencia de las mismas es de vital importancia. Nuestro ámbito de investigación es el estudio de índices eficientes para datos no estructurados.

1. Contexto

El presente trabajo se desarrolla en el ámbito de la línea Técnicas de Indexación para Datos no Estructurados del Proyecto Tecnologías Avanzadas de Bases de Datos (22/F014), cuyo objetivo es realizar investigación básica en problemas relacionados al manejo y recuperación eficiente de información no tradicional.

2. Introducción

Las bases de datos clásicas se organizan bajo el concepto de búsqueda exacta sobre datos estructurados. Esto significa que la información se organiza en registros divididos en campos que contienen valores completamente comparables. Una búsqueda en la base de datos retorna todos aquellos registros cuyos campos coinciden con los aportados en la consulta (búsqueda exacta). Actualmente las bases de datos han incluido la capacidad de almacenar datos no estructurados tales como imágenes, sonido, texto, video, datos geométricos, etc. La problemática de almacenamiento y búsqueda en estos tipos de base de datos difiere de las bases de datos clásicas, dado que no es posible organizar los datos en registros y campos. Aun cuando pudiera hacerse, la búsqueda exacta carece de interés.

Es en este contexto donde surgen nuevos modelos de bases de datos capaces de cubrir las necesidades de almacenamiento y búsqueda de estas aplicaciones. Nuestro interés se basa en el diseño de índices para estas nuevas bases de datos, centrándonos en bases de datos textuales y en espacios métricos.

Bases de Datos Textuales

Un base de datos de texto es un sistema que mantiene una colección grande de texto, y provee acceso rápido y seguro al mismo. Sin pérdida de generalidad, asumiremos que la base de datos de texto es un único texto T posiblemente almacenado en varios archivos. Las búsquedas en la que el usuario ingresa un *patrón de búsqueda* y el sistema retorna todas las posiciones del texto donde el patrón ocurre, es una de las búsquedas más comunes en este tipo de bases de datos. Si el texto es pequeño, la búsqueda de patrones puede resolverse eficientemente sin indexar el texto. Si el texto es demasiado grande se debe preprocesar el texto para construir un índice.

Mientras que en bases de datos tradicionales los índices ocupan menos espacio que el conjunto de datos indexado, en las bases de datos de texto el índice ocupa más espacio que el texto, pudiendo necesitar de 4 a 20 veces el tamaño del mismo [4, 8]. Por lo tanto construir un índice tiene sentido cuando el texto es grande, cuando las búsquedas son más frecuente que las modificaciones (de manera tal que los costos de construcción se vean amortizados) y cuando hay suficiente espacio como para contener el índice. Una alternativa para reducir el espacio ocupado por el índice es buscar una representación compacta del mismo, manteniendo las facilidades de navegación sobre la estructura. Pero en grandes colecciones de texto, el índice aún comprimido suele ser demasiado grande como para residir en memoria principal. [5, 6]. Por esta razón, el estudio de índices comprimidos y en memoria secundaria para búsquedas en texto es un tema de creciente interés en la comunidad de bases de datos.

Espacios Métricos

El modelo de espacios métricos permite formalizar el concepto de búsqueda por similitud en bases de datos no tradicionales [1].

Un espacio métrico está formado por un conjunto de objetos $\mathcal{X}$ y una función de distancia d definida entre ellos que mide cuan diferentes son. La base de datos será un subconjunto finito $\mathcal{U} \subseteq \mathcal{X}$.

Una de las consultas más comunes en este modelo de bases de datos es la *búsqueda por rango*. En esta búsqueda dado un elemento $q \in \mathcal{X}$, al que llamaremos *query* y un radio de tolerancia r, la búsqueda por rango consiste en recuperar los objetos de la base de datos cuya distancia a q no sea mayor que r. Para evitar examinar exhaustivamente la base de datos, se preprocesa la misma por medio de un *algoritmo de indexación* con el objetivo de construir una *índice*, diseñado para ahorrar cálculos en el momento de la búsqueda. En [1] se presenta un desarrollo unificador de las soluciones existentes en la temática. Básicamente se pueden distinguir dos grupos de algoritmos: *basados en pivotes* y *basados en particiones compactas*.

3. Líneas de Investigación

Nuestra principal línea de trabajo es el estudio de algoritmos de indexación para el procesamiento de consultas sobre bases de datos no estructurados, centrándonos principalmente en el diseño de índices para bases de datos textuales y para espacios métricos Describimos a continuación las líneas de investigación que actualmente estamos desarrollando.

3.1. Indexación en Bases de Datos Textuales

Dado un texto $T = t_1, \ldots, t_n$ sobre un alfabeto Σ de tamaño σ, donde $t_n = \$ \notin \Sigma$ es un símbolo menor en orden lexicográfico que cualquier otro símbolo de Σ, denotaremos con $T_{i,j}$ a la secuencia $t_i, \ldots, t_j$, con $1 \leq i \leq j \leq n$. Un sufijo de T es cualquier string de la forma $T_{i,n}$ y un prefijo de T es cualquier string

de la forma $T_{1,i}$ con $i = 1..n$. Un patrón de búsqueda P es cualquier string sobre el alfabeto Σ.

Si el texto es pequeño, la búsqueda de patrones puede resolverse eficientemente sin indexar el texto. Si el texto es demasiado grande se debe preprocesar el texto para construir un índice. Entre los índices más populares para resolver búsqueda de patrones encontramos el *arreglo de sufijos* [8], el *trie de sufijos* [11] y el *árbol de sufijos* [11]. Estos índices se construyen basándose en la observación de que un patrón P ocurre en el texto si es prefijo de algún sufijo del texto.

El objetivo principal en esta línea de investigación es el diseño de índices comprimidos en memoria secundaria, que resulten eficientes para la búsqueda de patrones. A continuación resumimos el trabajo en curso.

Trie de Sufijos

El trabajo sobre este índice apunta a lograr una representación en memoria secundaria de un trie de sufijos, de manera tal que el mismo resulte competitivo en cantidad de accesos a disco. Para ello, hemos extendido la técnica de paginación de un árbol binario, el Compact Pat Tree [2], a árboles r-arios.

Al igual que el algoritmo presentado en [2], nuestra técnica de paginado también consiste en particionar el árbol en componentes conexas, denominadas *partes*, cada una de las cuales se almacena en una página de disco.

El algoritmo procede en forma bottom-up tratando de condensar en una única parte un nodo con uno o más subárboles que dependen de él. En este proceso de particionado las decisiones se toman en base a la profundidad de cada nodo involucrado, donde por profundidad se entiende la cantidad de accesos a disco que deberá realizar el proceso de búsqueda para llegar desde esa parte a una hoja del árbol.

Para particionar un árbol, el algoritmo comienza asignando cada hoja a una parte con profundidad 1 y luego, en forma bottom-up, procesa cada uno de los nodos de este árbol r-ario según las siguientes reglas:

Paso 1: se ordenan los hijos del nodo corriente de mayor a menor según su profundidad.

Paso 2: si el nodo corriente y el hijo de mayor profundidad d entran en una página de disco, colocamos en una misma página el nodo corriente y tantos hijos como entren en una página. Estos hijos se toman de acuerdo al orden establecido en el paso a). Se establece la profundidad de la nueva parte creada en d, donde d es el máximo de las profundidades de los hijos, y se cierran las partes de aquellos hijos que no conforman la nueva parte creada.

Paso 3: si el padre no puede unirse con el hijo de mayor profundidad, se cierran todas las partes hijas y se crea una nueva parte para el nodo corriente con profundidad $d + 1$, donde d es el máximo de las profundidades de los hijos.

Los primeros resultados obtenidos con esta técnica son alentadores, por lo que nos encontramos en la etapa de ajuste de parámetros de la misma para mejorar la eficiencia.

String B-Tree

El String B-Tree(SBT) [3] es un índice dinámico para búsquedas de patrones en memoria secundaria. Básicamente consiste en una combinación de dos estructuras: el B-Tree y el Pat-Tree [4]. No es un índice comprimido y su versión estática requiere en espacio de 5 a 6 veces el tamaño del texto.

Sobre el SBT el objetivo principal es lograr una reducción en el espacio utilizado por el mismo manteniendo los costos de búsquedas de la versión original. Para ello, se han diseñado e implementado dos variantes que consisten en modificar la representación de cada nodo del árbol B subyacente. Una de las variantes consiste en usar un Pat-Tree como

originalmente proponen los autores para los nodos pero usando representación de paréntesis para el mismo [9]. La otra variante consiste en representar cada nodo con la representación de arreglos propuesta en [10] que ofrece las mismas funcionalidades que un Pat-Tree pero que tienen las características necesarias como para permitir una posterior compresión de los mismos. Estamos trabajando en la implementación de una variante de este índice que consiste en comprimir la representación de los Pat-Tree involucrados. Para ello se utiliza la representación de paréntesis [9] para mantener la topología del árbol y los códigos DAC para la representación de los valores de salto del Pat-Tree. La implementación ha sido realizada, encontrándonos en la etapa de evaluación experimental y ajuste de parámetros.

3.2. Espacios Métricos

El procesamiento de consultas en espacios métricos es un tema de investigación emergente tanto desde el punto de vista de los algoritmos que las implementan como de los índices que las soportan.

Uno de los principales obstáculos en el diseño de buenas técnicas de indexación en espacios métricos es lo que se conoce con el nombre de *maldición de la dimensionalidad*. El concepto de dimensionalidad está relacionado a la dificultad o facilidad de buscar en un determinado espacio métrico. En [1] se define el concepto de dimensión intrínseca de un espacio métrico y se muestra que a medida que la dimensionalidad intrínseca crece, la media crece y su varianza del histgramas de distancias se reduce. Esto significa que el histograma de distancia se concentra más alrededor de su media, lo que influye negativamente en los algoritmos de indexación.

En el modelo de espacios métricos los algoritmos de indexación se clasifican en dos grandes grupos: basados en pivotes y basados en particiones compactas. En este trabajo estamos interesados en los algoritmos basados en particiones compactas. Estos algoritmos basan la construcción del índice en un grupo de elementos preseleccionados denominados *centros*. Los centros seleccionados durante la construcción del índice no afectan la efectividad del mismo pero son cruciales para su eficiencia.

Se sabe que la forma en que se seleccionan los centros afecta en gran medida el desempeño del índice creado. La selección trivial es la random, pero la experiencia marca que aquellas tareas realizadas aleatoriamente pueden mejorarse incorporando alguna política específica. El grupo de centros seleccionados durante la construcción del índice no afecta en absoluto la efectividad del mismo pero es crucial para su eficiencia.

En esta línea nos proponemos el estudio de nuevas técnicas de selección de centros para índices basados en particiones compactas. Las técnicas diseñadas se basan en la información que brindan los histogramas de distancia como una forma de identificar la concentración de elementos en distintas zonas del espacio.

Hemos diseñado hasta el momento dos nuevas políticas de selección, las que resultaron competitivas en los casos de prueba utilizados [7]. En la actualidad estamos evaluando el comportamiento de estas técnicas sobre otros tipos de espacios métricos.

En otra línea de trabajo, avanzamos en el diseño de un prototipo de aplicación de índices métricos al comercio electrónico.

El objetivo de este trabajo es agilizar las búsquedas por similitud sobre un conjunto de productos clasificados en distintas categorias. Para ello se utilizan diferentes índices métricos los que dada la descripción de un producto permiten identificar eficientemente todos aquellos productos con descripción similar al dado.

Actualmente nos encontramos en la etapa de diseño del prototipo analizando las características de los índices métricos exitentes para identificar cuál de ellos se adapta mejor a este

caso de estudio.

4. Resultados Esperados

Se espera obtener índices eficientes, tanto en espacio como en tiempo, para el procesamiento de consultas en bases de datos textuales y en espacios métrico. Los mismos serán evaluados tanto analíticamente como empíricamente. Para esto último se cuenta con un conjunto de lotes de prueba usados y aceptados por la comunidad científica del área de estudio. Los mismos se encuentran disponibles en los sitios `http://pizzachili.dcc.uchile.cl` y `http://sisap.org/Home.html`.

5. Recursos Humanos

El trabajo desarrollado en esta línea forma parte del desarrollo de tres Trabajos Finales de la Licenciatura, dos Tesis de Maestría y una Tesis de Doctorado, todas ellas en el ámbito de Ciencias de la Computación en la Universidad Nacional de San Luis. Se cuenta con el asesoramiento del Dr. Gonzalo Navarro de la Universidad de Chile y de la Dra. Nieves Brisaboa de la Universidad de Coruña, España.

Referencias

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

[2] D. Clark and I. Munro. Efficient suffix tree on secondary storage. In *Proc. 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 383–391, 1996.

[3] P. Ferragina and R. Grossi. The string B-tree: a new data structure for string search in external memory and its applications. *Journal of the ACM*, 46(2):236–280, 1999.

[4] G. H. Gonnet, R. Baeza-Yates, and T. Snider. *New indices for text: PAT trees and PAT arrays*, pages 66–82. Prentice Hall, New Jersey, 1992.

[5] R. González and G. Navarro. A compressed text index on secondary memory. In *Proc. 18th International Workshop on Combinatorial Algorithms (IWOCA)*, pages 80–91. College Publications, UK, 2007.

[6] R. González and G. Navarro. Compressed text indexes with fast locate. In *Proc. 18th Annual Symposium on Combinatorial Pattern Matching (CPM)*, LNCS 4580, pages 216–227, 2007.

[7] A. Lucero, C. Ruano, and N. Herrera. Técnicas de selección de centros basadas en histogramas de distancia. In *Actas del XVII Congreso Argentino de Ciencias de la Computación*, Argentina, 2011.

[8] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. *SIAM Journal of Computing*, 22(5):935–948, 1993.

[9] J. Ian Munro and Venkatesh Raman. Succinct representation of balanced parentheses and static trees. *SIAM J. Comput.*, 31(3):762–776, 2001.

[10] J. Na and K. Park. Simple implementation of string b-trees. In Alberto Apostolico and Massimo Melucci, editors, *SPIRE*, volume 3246 of *Lecture Notes in Computer Science*, pages 214–215. Springer, 2004.

[11] P. Weiner. Linear pattern matching algorithm. In *Proc. 14th IEEE Symposium Switching Theory and Automata Theory*, pages 1–11, 1973.