Indexación y Búsqueda sobre Datos no Estructurados

Norma Herrera, Darío Ruano, Paola Azar, Susana Esquivel
Departamento de Informática
Universidad Nacional de San Luis, Argentina
{nherrera, dmruano, epazar, esquivel}@unsl.edu.ar

Anabella De Battista
Departamento Ingeniería en Sistemas de Información
FRCU, Universidad Tecnológica Nacional
Entre Ríos, Argentina
debattistaa@frcu.utn.edu.ar

Abstract

Las bases de datos actuales han incluido la capacidad de almacenar datos no estructurados tales como imágenes, sonido, texto, video, 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 organizarlos en registros y campos, y 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 eficientes para estas nuevas bases de datos.

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

La mayoría de los administradores de bases de datos actuales están basados en el modelo relacional, presentado por Edgard F. Codd en 1970. Bajo el modelo relacional, cada elemento de la base de datos puede ser almacenado como un registro (tupla) y cada registro a su vez dividido en campos (atributos). La mayoría de las consultas que se realizan a una base de datos relacional (conocidas también como bases de datos tradicionales) se corresponden con *búsquedas exactas*, esto significa obtener todos los registros cuyos campos coinciden exactamente con los campos aportados durante la búsqueda. También se pueden realizar búsquedas por rango sobre valores numéricos, y búsquedas de sub-cadenas sobre campos alfabéticos; en estos casos debe existir una relación de orden sobre los campos consultados.

La información disponible en formato digital aumenta día a día su tamaño de manera ex-

ponencial. Gran parte de esta información involucra el uso de datos no estructurados tales como imágenes, sonido, texto, video, etc. Debido a que no es posible organizar estos tipos de datos en registros y campos, las tecnologías tradicionales de bases de datos para almacenamiento y búsqueda de información no son adecuadas en este ámbito.

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.

2.0.1 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 [5, 11]. 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 [6, 7]. 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.

2.0.2 Espacios Métricos

El modelo de espacios métricos permite formalizar el concepto de búsqueda por similitud en bases de datos no tradicionales [2].

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 [2] 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*.

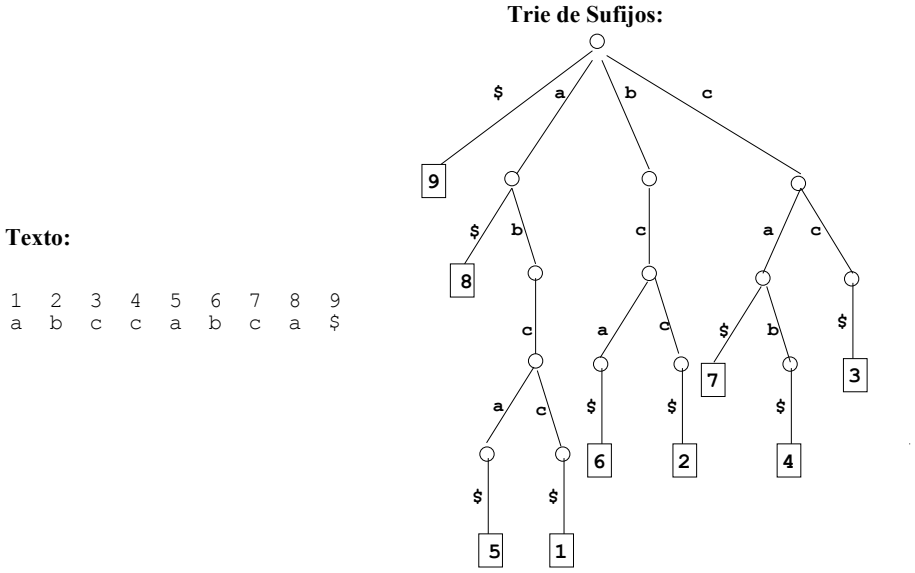

Figure 1: *Un ejemplo de un texto y su correspondiente trie de sufijos.*

3 Líneas de Investigación

3.1 Índices Comprimidos para Bases de Datos de Texto

Como ya mencionáramos, el principal problema que surge al indexar una bases de datos de texto es el espacio ocupado por el índice.

En bases de datos tradicionales, si construimos un índice para una relación R la cantidad de puntos de indexación está dado por la cantidad de nuplas de R y no por el espacio ocupado por R: si la relación ocupa k bytes y tiene n nuplas, en el índice existirán n puntos de indexación; notar que siempre $n < k$. En bases de datos de texto, cada caracter del texto debe ser considerado dentro del índice, en consecuencia la cantidad de puntos de indexación está dado por el tamaño del texto: si el texto ocupa k bytes, existirán k puntos de indexación.

Una forma de tratar con este problema es buscar una representación compacta del índice, manteniendo las facilidades de navegación sobre la estructura. Esto significa encontrar una representación que ocupe menos espacio que la representación clásica, pero que permita navegar sobre el índice sin necesidad de descomprimirlo [3, 4, 6, 7, 10, 12, 13, 15].

Un ***trie de sufijos*** es un índice que permite resolver eficientemente las operaciones de búsquedas en texto pero que necesita en espacio 10 veces el tamaño del texto indexado. Por ejemplo, si construimos un trie de sufijos sobre un texto de 10GB, el espacio requerido para almacenarlo será de 100GB. Por esta razón, el diseño de técnicas de representación compactas para este índice es de interés.

La representación habitual de un trie consiste en mantener en cada nodo los punteros a sus hijos, junto con el rótulo correspondiente a cada uno de ellos. Existen distintas variantes de representación que consisten en organizar estos punteros a los hijos sobre una lista secuencial, sobre una lista vinculada o sobre una tabla de hashing [9]. Una de las propuestas de representación que mejor desempeño tiene en memoria principal es la de Kurtz, quien basándose en la idea de la representación sobre una lista vinculada, propuso que cada nodo mantenga un apuntador al primer hijo y almacenar los nodos hermanos en posiciones consecutivas de memoria. Esto permite durante una búsqueda, realizar una búsqueda binaria

sobre los rótulos para decidir por cual hijo seguir.

La mayoría de las propuestas existentes mantienen explícitamente la forma del árbol con punteros, los que pueden ser punteros físicos (direcciones de memoria principal) o punteros lógicos (posiciones de un arreglo).

En [14] se presenta una nueva representación de un trie de sufijos que permite reducir el espacio necesario para almacenar el índice, eliminando la necesidad de mantener los punteros explícitos a los hijos. Esta representación surge como una extensión a árboles r-arios de la técnica presentada en [8] y tiene la ventaja de permitir un posterior proceso de paginado para manejar eficientemente el trie de sufijos en memoria secundaria [16].

Notar que la información contenida en el trie está compuesta por: la forma del árbol, el rótulo de cada rama, el valor de salto de cada nodo, el grado de cada nodo y el índice del sufijo asociado a cada hoja. En [14] se propone representar de manera secuencial cada una de estas componentes, manteniendo la posibilidad de navegar eficientemente sobre el trie.

Hemos realizado una implementación que mejora en espacio a la anterior en un 40%, sin afectar los tiempos de búsqueda. Esta nueva versión compacta del trie de sufijos consiste en usar códigos *DAC* (*Directly Addressable Variable-Length Code* [1]), para los arreglos que representan la secuencia de saltos y de grados. La navegación sobre esta nueva representación sigue los lineamientos generales propuestos en [14], adaptándolo a los códigos *DAC*.

Con respecto al trabajo futuro, nos proponemos integrar esta nueva representación con la técnica de paginado propuesta en [14], a fin de lograr un índice comprimido en memoria secundaria.

3.2 Espacios Métricos y Comercio Electrónico

El comercio electrónico, también conocido como e-commerce consiste en la compra y venta de productos o de servicios a través de la web. El desarrollo de nuevas tecnologías han permitido que la capacidad y volumen de las comunicaciones se expanda de una manera exponencial, lo que ha facilitado que el comercio electrónico tenga también un crecimiento exponencial.

Para el desarrollo de un sitio de comercio electrónico hay varios problemas que deben resolverse tales como administración de categorías de productos, búsqueda de productos, encriptación de datos, registros de usuarios, administración de medios de pago, entre otros. En este trabajo nos hemos centrado básicamente en el problema de búsqueda de productos.

Nuestro objetivo es utilizar búsquedas por similitud sobre las descripciones asociadas a los productos con el fin de estudiar el desempeño de los algoritmos basados en pivotes en un caso real de estudio.

Se ha diseñado e implementado el sistema que permite cumplir con el objetivo mencionado. Nos encontramos en la etapa de evaluación experimental del mismo a fin de establecer cuáles de las variantes utilizadas 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 métricos. 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 un Trabajo Final de la Licenciatura, una Tesis de Maestría y una Tesis de Doctorado, todas ellas en el área temática de Ciencias de la Computación, en la Universidad Nacional de San Luis.

References

[1] Nieves R. Brisaboa, Susana Ladra, and Gonzalo Navarro. Directly addressable variable-length codes. In *SPIRE*, pages 122–130, 2009.

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

[3] P. Ferragina and G. Manzini. Indexing compressed text. *J. ACM*, 52(4):552–581, 2005.

[4] P. Ferragina, G. Manzini, V. Mäkinen, and G. Navarro. Compressed representations of sequences and full-text indexes. *ACM Trans. Algorithms*, 3(2):20, 2007.

[5] 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.

[6] 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.

[7] 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.

[8] N. Herrera and G. Navarro. Árboles de sufijos comprimidos en memoria secundaria. In *Proc. XXXV Latin American Conference on Informatics (CLEI)*, Pelotas, Brazil, 2009.

[9] A. Thomo M. Barsky *, U. Stege. A survey of practical algorithms for suffix tree construction in external memory. In *Software: Practice and Experience*, 2010.

[10] V. Mäkinen and G. Navarro. *Compressed Text Indexing*, pages 176–178. Springer, 2008.

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

[12] G. Navarro. Indexing text using the ziv-lempel trie. *Journal of Discrete Algorithms (JDA)*, 2(1):87–114, 2004.

[13] G. Navarro and K. Sadakane. *Compressed Tree Representations*. Springer, 2nd edition, 2015.

[14] D. Ruano and N. Herrera. Representación secuencial de un trie de sufijos. In *XX Congreso Argentino de Ciencias de la Computación*, Buenos Aires, Argentina, 2014.

[15] K. Sadakane. New text indexing functionalities of the compressed suffix arrays. *J. Algorithms*, 48(2):294–313, 2003.

[16] J. Vitter. External memory algorithms and data structures: Dealing with massive data. *ACM Computing Surveys*, 33(2):209–271, 2001.