Indexación y Consultas para Bases de Datos no Convencionales

Jorge Arroyuelo, Susana Esquivel, Alejandro Grosso, Verónica Ludueña, Nora Reyes
Departamento de Informática, Universidad Nacional de San Luis
{bjarroyu,esquivel,agrosso,vlud,nreyes}@unsl.edu.ar

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

Resumen

La constante aparición de datos en forma digital de diferentes tipos y tamaños ha dado lugar a la aparición de depósitos no estructurados de información, *Bases de Datos Multimedia*, donde se consultan nuevos tipos de datos (texto libre, imágenes, audio, vídeo, etc.). Esto requiere un modelo más general tal como las *Bases de Datos Métricas*, que además alcance un nivel de madurez similar al de las bases de datos tradicionales. Por otro lado, la creciente cantidad de estos datos exige dispositivos de almacenamiento capaces de mantenerlos y de proveer un acceso eficiente a los mismos. Dado que la brecha entre los tiempos de CPU y los de I/O se ha mantenido creciente, se hace necesario considerar memorias con mayor capacidad y más rápidas. Este panorama ha promovido la aparición estructuras de datos especializadas que tienen en cuenta estas arquitecturas como las *Estructuras de datos compactas* y las *Estructuras de datos con I/O eficiente*. Nuestra investigación apunta a contribuir a la madurez de estas nuevas bases de datos.

Palabras Claves: bases de datos no convencionales, lenguajes de consulta, índices, expresividad.

Contexto

Esta línea de investigación pertenece al Proyecto Tecnologías Avanzadas de Bases de Datos. En el marco de este proyecto se vienen desarrollando actividades vinculadas al tratamiento de objetos de diversos tipos, estructurados y no estructurados que son de utilidad en diversos campos de aplicación, por ejemplo, robótica, visión artificial, computación gráfica, sistemas de información geográfica, computación móvil, diseño asistido por computadora, motores de búsqueda en internet, entre otras, y que se relacionan en tales bases de datos.

Este proyecto pertenece a la Universidad Nacional de San Luis y se encuentra dentro del Programa de Incentivos a la Investigación.

Las actividades centrales de esta línea están relacionadas con la investigación de aspectos teóricos, empíricos y aplicativos del problema general de administrar una base de datos capaz de manipular tipos de datos no convencionales. Esto incluye analizar distintos tipos de bases de datos, la expresividad de los lenguajes de consulta, los operadores necesarios para responder consultas de interés, como así también las estructuras y operaciones necesarias para responderlas eficientemente.

Además nuestras investigaciones se encuadran en el marco de un proyecto dentro del Programa de Promoción de la Universidad Argentina para el Fortalecimiento de Redes Interuniversitarias III en los que participa nuestra universidad junto con las universidades de: Chile y de La Coruña (España).

1. Introducción y Motivación

La brecha entre los tiempos de CPU y los de I/O se ha mantenido creciente durante las últimas décadas. Asimismo han aparecido nuevos niveles en la jerarquía de memoria (caches de tamaño cada vez más considerable). Por ello, se ha hecho cada vez más atractivo el uso de estructuras de datos que ocupen poco espacio, incluso a veces comprimiendo la información sobre la que actúan. Si bien trabajar sobre esta información compacta es más laborioso, el hecho de poder mantenerla en una memoria de órdenes de magnitud más rápida la convierte en una alternativa muy conveniente a las implementaciones clásicas. Además, la transferencia de los datos sobre una red local cuesta casi lo mismo que la transferencia a disco, por lo cual ésta también se ve favorecida con la compresión. Este escenario ha originado líneas de investigación que consideran estas diferencias de costos de operaciones, y diseñan estructuras

de datos más eficientes para memorias jerárquicas, utilizando la compacticidad o la I/O eficiente.

Particularmente nos centraremos en las estructuras de datos capaces de manipular los siguientes tipos de datos: secuencias, textos, árboles, grafos, y espacios métricos, entre otros, sobre los cuales carece de sentido realizar búsquedas exactas. Por lo tanto, se hace necesario un modelo más general tal como el de *espacios métricos* donde las búsquedas por similitud, más naturales sobre estos tipos de datos, son posibles. Además de diseñar estructuras de datos estáticas, planeamos investigar otros aspectos tales como la construcción eficiente (en espacio o en términos de la I/O u otras medidas de eficiencia), el dinamismo (es decir actualizaciones eficientes), operaciones de búsqueda complejas (más allá de las básicas soportadas por las estructuras de datos clásicas), tratar de obtener una mayor expresividad en los lenguajes que permitan expresarlas y caracterizar la clase de consultas computables.

En los espacios de vectores, representación más común para datos multimedia, la "maldición de la dimensionalidad" describe el fenómeno por el cual el desempeño de todos los índices existentes se deteriora exponencialmente con la dimensión. Aunque los espacios vectoriales son un caso particular de espacios métricos, aún no está determinado completamente cómo afecta la dimensión a los índices para espacios métricos. El estudio de distintas maneras de optimar algunas de las estructuras para búsquedas por similitud en espacios métricos se debe a que, de las numerosas estructuras que existen, sólo unas pocas trabajan eficientemente en espacios de alta o mediana dimensión, y la mayoría no admiten dinamismo, ni están diseñadas para trabajar sobre grandes volúmenes de datos; es decir, en memoria secundaria.

2. Lenguajes de Consulta

Como es deseable obtener información de una base de datos, es necesario contar con un lenguaje de consulta. Algunos de estos lenguajes son equivalentes, en su poder expresivo a la Lógica de Primer Orden (FO). Sin embargo no todas las consultas pueden ser expresadas en FO, esto ha llevado a la búsqueda de un mayor poder expresivo por medio de extensiones a la misma. Se logró incrementar la expresividad de esta lógica pero todavía aparece como incompleta, entonces se enfatizó el estudio de la expresividad de la Lógica de Segundo Orden (SO). En los últimos años se han realizado importantes investigaciones sobre la relación entre la teoría de modelos finitos y la teoría de complejidad computacional. Hay una relación cercana entre la complejidad computacional, es decir la cantidad de recursos necesarios para resolver un problema sobre algún modelo de máquina computacional, y la complejidad descriptiva, o sea el orden de la lógica que se necesita para describir el problema. La consecuencia más importante de esta relación es el resultado de Fagin [5]. Allí se establece que las propiedades de las estructuras finitas que son definidas por sentencias existenciales de segundo orden coinciden con las propiedades que pertenecen a la clase de complejidad NP, lo cual fue extendido por Stockmeyer [12] estableciendo una relación cercana entre la lógica de segundo orden y la jerarquía polinomial. Hay muchos resultados igualando la expresividad lógica a la complejidad computacional, pero ellos requieren estructuras ordenadas (ver [6], [7]).

En nuestra investigación hemos introducido una restricción de SO para la estructura finita, SO^F. En esta restricción los cuantificadores se extienden sobre la relación cerrada por la relación de equivalencia $\equiv^{FO}$. En esta relación de equivalencia las clases de equivalencia están formadas por $k-$tuplas cuyo *FO type* es el mismo, para algún entero $k \geq 1$. Esta lógica es una extensión de la lógica SO^ω definida por Dawar [3]. Además SO^ω está estrictamente incluida en SO^F. En el fragmento existencial de SO^F, $\Sigma_1^{1,F}$, podemos expresar cierto tipo de consultas booleanas que no puede ser expresadas en SO^ω. También definimos la clase de complejidad NP^F y demostramos que esta clase de complejidad es capturada por el fragmento existencial $\Sigma_1^{1,F}$.

Esta temática se está desarrollando como parte de las tesis de maestría y doctorado de dos investigadores de la línea.

3. Métodos de Acceso Métricos

Tomando como modelo para las bases de datos no convencionales a los espacios métricos, surge la necesidad de responder consultas por similitud eficientemente haciendo uso de *métodos de acceso métricos* (MAMs). En espacios métricos generales la complejidad se mide como el número de cálculos de distancias realizados. Por ello, se analizan aquellos MAMs que han mostrado buen desempeño en las búsquedas, con el fin de optimizarlos aún más, teniendo en cuenta la jerarquía de memorias.

3.1. Dimensión Intrínseca

En los espacios vectoriales existe una clara relación entre la dimensión (*intrínseca*) del espacio y la dificultad de buscar. Se habla de "intrínseca", como opuesta a "representacional". Los algoritmos más ingeniosos se comportan más de acuerdo a la dimensión intrínseca. Hay varios intentos de medir la dimensión intrínseca en espacios de vectores, como la transformada de *Karhunen-Loève (KL)* y otras medidas como *Fastmap* y, para espacios no uniformemente distribuidos, la *dimensión fractal* [1].

Existen sólo unas pocas propuestas diferentes sobre cómo estimar la dimensión intrínseca de un espacio métrico tales como el *exponente de la distancia* [13], y la medida de dimensión intrínseca como una medida cuantitativa basada en el histograma de distancias [2]. Aunque, también parece posible adaptar algunos de los estimadores de distancia en espacios de vectores para aplicarlos a espacios métricos generales, como por ejemplo *Fastmap* y *dimensión fractal*.

Muchos autores [2] han propuesto usar histogramas de distancia para caracterizar la dificultad de las búsquedas en espacios métricos arbitrarios. Existe al menos una medida cuantitativa [2], pero ella no refleja fielmente la facilidad o dificultad de buscar en un espacio métrico dado.

En aplicaciones reales de búsqueda en espacios métricos, sería muy importante contar con un buen estimador de la dimensión intrínseca porque nos permitiría decidir el índice adecuado a utilizar en función de la dimensión del espacio. Además, tener una buena estimación de la dimensión nos permitiría, en algunas ocasiones, elegir la función de distancia de manera tal que se obtenga una menor dimensión.

Este tema ha dado lugar a un trabajo final de la Lic. en Ciencias de la Computación, que está en desarrollo.

3.2. Árbol de Aproximación Espacial Dinámico

El estudio del *Árbol de Aproximación Espacial* [9], uno de los MAMs que había mostrado un muy buen desempeño en espacios de mediana a alta dimensión, pero que era totalmente estático, nos permitió el desarrollo de un nuevo índice llamado *Árbol de Aproximación Espacial Dinámico (SATD)* [10] que permite realizar inserciones y eliminaciones, conservando el buen desempeño en las búsquedas. Este logro ha sido importante ya que muy pocas estructuras para espacios métricos son completamente dinámicas.

El *SATD* es una estructura que realiza una partición del espacio considerando la proximidad espacial; pero, si el árbol agrupara los elementos que se encuentran muy cercanos entre sí, lograría mejorar las búsquedas al evitar recorrerlo para alcanzarlos.

Podemos pensar entonces que construimos un *SATD*, en el que cada nodo representa un grupo de elementos muy cercanos ("clusters") y relacionamos los clusters por su proximidad en el espacio. La idea sería que en cada nodo se mantenga el centro del cluster correspondiente, y se almacenen los k elementos más cercanos a él; cualquier elemento a mayor distancia del centro que los k almacenados, pasa a formar parte de otro nodo en el árbol.

Esperamos así obtener una estructura más eficiente en espacios donde de antemano se sabe que pueden existir "clusters" de elementos y que aprovechando la existencia de los mismos mejore las búsquedas.

Otro aspecto a analizar es cuán bueno es el agrupamiento o "clustering"que lograría esta estructura, lo cual podría estudiarse haciendo uso de nuevas estrategias de optimización de funciones a través de heurísticas bioinspiradas, las cuales han mostrado ser útiles en detección de clusters.

Estos temas han promovido un trabajo final de la Lic. en Ciencias de la Computación, en desarrollo actualmente.

3.3. Join Métricos

El modelo de espacios métricos permite cubrir muchos problemas de búsqueda por similitud o proximidad, aunque en general se deja fuera de consideración al ensamble o "join"por similitud, otra primitiva extremadamente importante [4]. De hecho, a pesar de la atención que esta primitiva ha recibido en las bases de datos tradicionales y aún en las multidimensionales, no han habido grandes avances para espacios métricos generales.

Nos hemos planteado resolver algunas variantes del problema de join por similitud: (1) *join por rango:* dadas dos bases de datos de un espacio métrico y un radio r, encontrar todos los pares de objetos (uno desde cada base de datos) a distancia a lo sumo r, (2) *k–pares más cercanos:* encontrar los k pares de objetos más cercanos entre sí (uno desde cada base de datos). Para resolver estas operaciones de manera eficiente hemos diseñado un nuevo índice métrico, llamado *Lista de Clusters Gemelos* (*LTC*) [11],

éste se construye sobre ambas bases de datos conjuntamente, en lugar de indexar una o ambas bases de datos independientemente. Esta nueva estructura permite además resolver las consultas por similitud clásicas en espacios métricos sobre cada una de las bases de datos independientemente.

A pesar de que esta estructura ha mostrado ser competitiva y obtener buen desempeño en relación a las alternativas más comunes para resolver las operaciones de join, aún queda mucho por mejorar para que se vuelva una estructura práctica y mucho más eficiente para trabajar con grandes bases de datos métricas. De esta manera sería posible pensar en extender apropiadamente álgebra relacional como lenguaje de consulta y diseñar soluciones eficientes para nuevas operaciones, teniendo en cuenta aspectos no sólo de memoria secundaria, sino también de concurrencia, confiabilidad, etc. Algunos de estos problemas ya poseen solución en las bases de datos espaciales, pero no en el ámbito de los espacios métricos.

Este tema forma parte de la tesis doctoral de uno de los investigadores de la línea.

4. Índices Compactos para Texto

Un tipo de dato no convencional es el texto, en él se necesita también realizar búsquedas por similitud; donde el problema de la *búsqueda aproximada de un patrón* en una secuencia puede verse como: sea $T = T[1, n]$ un *texto*, y $P = P[1, m]$ un *patrón* sobre el alfabeto Σ (con $m << n$) y un entero k, se desea encontrar y devolver todos los *substrings* en el texto T que sean una ocurrencia aproximada de P, con a lo más k diferencias. La diferencia entre dos strings α y β se obtiene con la *distancia de edición* d; $d(\alpha,\beta)$ es el mínimo número de inserciones, eliminaciones y/o sustituciones de caracteres que se deben realizar para convertir β en α. Este tipo de búsqueda tiene aplicaciones tales como la recuperación de errores (en reconocimiento óptico de caracteres, spelling), biología computacional, comunicaciones de datos, data mining, bases de datos textuales, entre otras.

Una variante de los índices clásicos son los llamados índices *compactos* que suelen aprovechar la existencia de la jerarquía de memorias. Éstos funcionan en espacio reducido y se pueden dividir en dos grupos: *sucintos* y *comprimidos*. Una medida popular de compresibilidad de secuencias es la entropía de orden k-ésimo (H_k), según lo definido por Manzini [8]. Siendo s el alfabeto de una secuencia de símbolos S, $n(i)$ el número de ocurrencias del i-ésimo símbolo, y n la longitud de S, otra medida popular es $H_0(S) = \Sigma(n(i)\log(n/n(i))$. Sea w una secuencia de longitud k y $w(S)$ la subsecuencia de símbolos de S que siguen a w, entonces $H_k(S) = 1/n\Sigma(|w|H_0(w))$ sobre todos los posibles w. Se usa en aquellos índices comprimidos que codifican un texto T de n símbolos sobre un alfabeto s usando espacio de $H_k(T) + o(n \log s)$ bits y además pueden buscar eficientemente patrones en el texto. Observar que la codificación llana del texto toma $n \log s$ bits.

La idea de los índices compactos se diferencia de la compresión pura en su capacidad de manipular los datos en forma comprimida, sin tener que descomprimirlos primero. En la actualidad los índices compactos pueden manipular secuencias de bits o de símbolos generales, árboles en general, grafos, colecciones de texto, permutaciones y mapping, sumas parciales, búsqueda por rango en una y más dimensiones, etc.

Indexación con q-gramas. Un q-grama es una subsecuencia de tamaño q de un texto. Un índice de q-gramas es una estructura de datos que permite encontrar rápidamente en el texto todas las ocurrencias de un q-grama dado. Existen distintas implementaciones de índices para q-gramas. La básica es un arreglo de punteros de tamaño $|\Sigma|^q$. Cada posición del arreglo referencia a la lista de ocurrencias en el texto del q-grama correspondiente. Una mejora a este esquema de indexación es reducir el tamaño del arreglo de punteros por medio de un hashing eficiente, sin una demora significativa en las búsquedas. Otro enfoque usa una estructura de trie construído con los distintos q-gramas que aparecen en el texto. Cada hoja del trie contiene un puntero a la lista de ocurrencias del correspondiente q-grama en el texto.

Las implementaciones anteriores encuentran todas las ocurrencias de un q-grama dado en tiempo óptimo en función del número de ocurrencias encontradas. Pero todas sufren el mismo inconveniente: el tamaño del índice se vuelve impráctico al crecer la longitud del texto. Existe un índice para q-gramas que reemplaza las listas de ocurrencias con una estructura de datos más compacta, es el *Lempel-Ziv* (LZ). En él, la estructura (hashing o trie) para los distintos q-gramas, ahora llamada *índice primario*, es aún necesaria para proveer un punto de comienzo para las búsquedas. La representación compacta de las listas de ocurrencias toma ventaja de las *repeti-*

ciones en el texto. La primera ocurrencia del string será llamada *definición* y las siguientes se llamarán *frases*. Luego, cada ocurrencia de un q-grama es o el primero de su clase o parte de alguna frase. La primera ocurrencia se almacenará en el índice primario y las demás se encontrarán usando un parsing de Lempel-Ziv[14] que usa la información sobre repeticiones. El índice LZ encontrará todas las ocurrencias de un q-grama en igual tiempo que los índices tradicionales.

Esta temática se investiga en el marco de la tesis de maestría de un investigador de la línea.

5. Conclusiones y Trabajos Futuros

Como trabajo futuro de esta línea de investigación se consideran varios aspectos relacionados al diseño de estructuras de datos que, concientes de que existe una jerarquía de memorias y de las características particulares de los datos a ser indexados, saquen el mejor partido haciendólas eficientes tanto en espacio como en tiempo.

Se trabajará en particular con estructuras de datos compactas para textos, implementando un índice LZ para q-gramas y estudiando su comportamiento en comparación con los índices tradicionales.

En el caso de espacios métricos, se intentará que las estructuras se adapten mejor al espacio métrico particular considerado, gracias a la determinación de su dimensión intrínseca, y también al nivel de la jerarquía de memorias en que se deba almacenar. Es importante destacar que estos estudios sobre espacios métricos y sobre algunas estructuras de datos particulares (como el *SATD*) permitirán no sólo mejorar el desempeño de las mismas sino también aplicar, eventualmente, muchos de los resultados que se obtengan a otras estructuras para espacios métricos.

Respecto de los lenguajes de consulta se continuará analizando la expresividad de distintas extensiones de FO y de posibles restricciones SO, con el propósito de lograr caracterizar la clase de las consultas computables sobre bases de datos no convencionales.

Referencias

[1] F. Camastra. Data dimensionality estimation methods: a survey. *Pattern Recognition*, 36(12):2945–2954, 2003.

[2] E. Chávez and G. Navarro. Towards measuring the searching complexity of metric spaces. In *Proc. International Mexican Conference in Computer Science*, volume II, pages 969–978. Sociedad Mexicana de Ciencias de la Computación, 2001.

[3] A. Dawar. A restricted second order logic for finite structures. *Information and Computation*, 143:154–174, 1998.

[4] V. Dohnal, C. Gennaro, P. Savino, and P. Zezula. Similarity join in metric spaces. In *Proc. 25th European Conf. on IR Research*, LNCS 2633, pages 452–467, 2003.

[5] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of Computation*, 7:43–73, 1974.

[6] N. Immerman. Descriptive and computational complexity. *Computational Complexity Theory*, 38:75–91, 1989.

[7] N. Immerman. Descriptive complexity. *Springer*, 1998.

[8] G. Manzini. An analysis of the burrows-wheeler transform. *J. ACM*, 48(3):407–430, 2001.

[9] G. Navarro. Searching in metric spaces by spatial approximation. *The Very Large Databases Journal (VLDBJ)*, 11(1):28–46, 2002.

[10] G. Navarro and N. Reyes. Fully dynamic spatial approximation trees. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, LNCS 2476, pages 254–270. Springer, 2002.

[11] R. Paredes and N. Reyes. Solving similarity joins and range queries in metric spaces with the list of twin clusters. *J. of Discrete Algorithms*, 7(1):18–35, 2009.

[12] L. Stockmeyer. The polynomial-time hierarchy. *Theoret. Comput. Sci.*, 3:1–22, 1976.

[13] C. Traina Jr., A. Traina, and C. Faloutsos. Distance exponent: A new concept for selectivity estimation in metric trees. In *ICDE*, page 195, 2000.

[14] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.