# Indexación y Administración de Grandes Volúmenes de Datos

*Luis Britos, Fernando Kasián, Verónica Ludueña, Franco Merenda, Diego Olivares, Marcela Printista, Nora Reyes, Patricia Roggero, Pablo Samat*
LIDIC, Dpto. de Informática, Fac. de Cs. Físico Matemáticas y Naturales, Universidad Nacional de San Luis
{*fkasian, vlud, mprinti, nreyes, proggero*}@unsl.edu.ar,
{*ldoorz, merenda.franco83, olivarestello.diego, samatpablo*}@gmail.com
*Karina Figueroa*
Universidad Michoacana de San Nicolás de Hidalgo, México
*karina*@fismat.umich.mx
*Claudia Deco*
Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario
*deco*@fceia.unr.edu.ar

## Resumen

*En la actualidad es habitual contar con grandes repositorios de datos no estructurados provenientes de distintas fuentes, los cuales son difíciles de administrar bajo el modelo relacional de base de datos. Este crecimiento sostenido tanto de la cantidad de datos en formato digital disponible, como en la variedad de los mismos, se debe a la rápida evolución de las tecnologías de la información y comunicación. Al considerar tipos de datos tales como texto libre, imágenes, audio, video, secuencias biológicas de ADN o proteínas, entre otros, las consultas no se corresponden a las habituales y forzar una estructuración podría restringir de antemano los diversos tipos de consultas que se puedan responder sobre ellos.*

*Así, al considerar el procesamiento de grandes conjuntos de datos no estructurados y recuperar a partir de ellos información de interés, se hace evidente la necesidad de proponer nuevos modelos y herramientas para su indexación y administración. Como el objetivo de cualquier sistema de recuperación de información es obtener información valiosa para el usuario, realizando una consulta a una base de datos, para resolverla de manera eficiente es necesario contar con índices apropiados.*

**Palabras Claves:** *bases de datos masivas, computación de alto desempeño, recuperación de información.*

## 1. Contexto

Esta línea de investigación "Recuperación de Datos e Información" se encuentra enmarcada dentro del Proyecto Consolidado 3-03-2018 de la Universidad Nacional de San Luis (UNSL - Res. CS 126/18) y en el Programa de Incentivos (Código 22/F834): "Tecnologías Avanzadas Aplicadas al Procesamiento de Datos Masivos", se desarrolla en el Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC) de la UNSL y finaliza en 2022.

El principal objetivo de esta línea es desarrollar herramientas eficientes para indexar y administrar bases de datos masivas, que almacenan datos no estructurados. El análisis de nuevas técnicas que provean una buena interacción con el usuario, al igual que el desarrollo de nuevas estructuras de datos capaces de manipular eficientemente un gran volumen de datos no estructurados, que aporten a la recuperación de información sobre estos tipos de datos, están orientados en ese sentido. Para ello, se considera el diseño y desarrollo de índices para conjuntos de datos no estructurados masivos (datos multimedia, texto, secuencias de ADN, huellas digitales, etc.), que sean eficientes y escalables, para memorias jerárquicas, aplicando de técnicas de computación de alto desempeño (HPC). Además, se busca su incorporación en un sistema de administración para estas bases de datos, que apoye la recuperación de información sobre estos tipos de datos.

## 2. Introducción y Motivación

La evolución de las tecnologías de información y comunicación, el uso masivo de internet y la gran disponibilidad de dispositivos electrónicos, ha permitido que continuamente se esté generando un gran volumen de datos y que, por provenir de fuentes muy diversas, los tipos de datos producidos son también muy variados. Este escenario obliga a redefinir las técnicas de procesamiento, análisis y obtención de información útil, y a formular nuevas metodologías más aplicables a las soluciones que se proponen.

En el contexto de problemas de "big data" apare-

cen dos características importantes, que son: el volumen y variedad de los mismos, lo que hace imposible restringir las consultas a búsquedas sobre datos estructurados tradicionales, porque obligaría a representar una visión parcial del problema, dejando fuera información que podría ser relevante para la resolución efectiva del mismo. Las búsquedas por similitud son un tipo de búsqueda más general y aplicable a este contexto. Para lograr resolver dichas búsquedas eficientemente, es necesario considerar *métodos de acceso* o *índices métricos* [5]. Un enfoque útil para sistemas de recuperación por similitud es *la búsqueda basada en contenidos*, donde se usa el dato no estructurado en sí mismo para describir lo que se está buscando.

Entonces, al considerar grandes cantidades de datos no estructurados y necesitar responder consultas de recuperación de información, se pueden utilizar estos índices métricos para lograr eficiencia en la respuesta. Más aún, en aplicaciones reales los índices deben ser eficientes, dinámicos y escalables.

El modelo habitual para las búsquedas por similitud es el de *espacios métricos*. Una de las principales ventajas de este modelo es que, además de brindar un marco formal, es independiente del dominio de la aplicación. Un espacio métrico está compuesto por un *universo* $\mathcal{U}$ de objetos y una *función de distancia* $d : \mathcal{U} \times \mathcal{U} \longrightarrow \mathbb{R}^+$, la cual cumple con las propiedades de una métrica. Sobre una *base de datos* $\mathcal{S} \subseteq \mathcal{U}$, se suelen considerar dos tipos básicos de búsqueda por similitud: la *búsqueda por rango* y la *búsqueda de los $k$ vecinos más cercanos*. La función de distancia permite medir la disimilitud entre dos objetos. El cálculo de distancia sobre algunos tipos de datos no estructurados puede ser muy costoso. Por lo tanto, un objetivo importante en el diseño de los índices es ahorrar cálculos de distancia..

Al considerar que $\mid \mathcal{S} \mid = n$, cualquier consulta se resuelve de manera trivial examinando los $n$ elementos y calculando así $n$ evaluaciones de distancia. Sin embargo, como la distancia puede ser costosa de calcular (por ej.: comparación de huellas digitales), en la mayoría de las aplicaciones sobre grandes volúmenes de datos no es factible aplicar la solución trivial. Entonces, para resolver consultas minimizando los cálculos de distancia, es necesario preprocesar la base de datos para construir un índice.

En ciertos casos particulares, es probable que la base de datos, el índice, o ambos, no puedan almacenarse en memoria principal y deban hacer uso de niveles más bajos de la jerarquía de memorias, como la memoria secundaria. Sin embargo, ello acarrea altos costos en las operaciones de E/S. Por lo tanto, para lograr eficiencia, se debe minimizar también el número de operaciones de E/S, considerar el nivel de la jerarquía de memorias sobre la que se trabaja y en algunos casos admitir respuestas no exactas; utilizando, cuando se pueda, técnicas de HPC.

En este contexto, se considera como objetivo principal obtener herramientas de recuperación de información para procesar conjuntos masivos de datos, desarrollando nuevas técnicas y aplicaciones que soporten la interacción con el usuario, diseñando índices que permitan la manipulación eficiente de grandes volúmenes de datos no estructurados y faciliten la realización de diferentes tipos de consultas. De esta manera, se espera contribuir al desarrollo de aplicaciones reales para problemas de big data.

## 3. Líneas de Investigación

Dado que en esta investigación se pretende contribuir a distintos aspectos de los sistemas de recuperación de información sobre grandes volúmenes de datos no estructurados, se ha considerado el diseño de nuevos índices y la optimización de índices existentes, la resolución de distintas consultas sobre estos tipos de bases de datos y cómo lograr eficiencia y escalabilidad para grandes volúmenes de datos.

### Índices

Los índices métricos resultan apropiados para realizar búsquedas sobre bases de datos que contienen datos no estructurados [5]. Éstos aprovechan una propiedad de la función de distancia, la desigualdad triangular, y las distancias almacenadas en el índice, para ahorrar algunos cálculos de distancia y ahorrar tiempo. Si se mantienen algunas distancias precalculadas entre los elementos de la base de datos y objetos particulares o distinguidos, la desigualdad triangular permitirá estimar la distancia entre cualquier objeto de consulta $q$ y los elementos de la base de datos. Los dos enfoques más comunes se diferencian en si esos objetos distinguidos son *pivotes* o *centros*. Si son pivotes se almacenan las distancias de todos los objetos de la base de datos a ellos y si por el contrario son centros se particiona el espacio en zonas denominadas *particiones compactas*, por cercanía a los centros, almacenando en general un radio de cobertura que determina la zona de cada centro.

Los aspectos que se consideran de interés al diseñar índices incluyen: dinamismo, en qué nivel de la jerarquía de memorias deben almacenarse, si pue-

den aplicar técnicas de computación de alto desempeño para mejorar los tiempos de respuesta, si deben proporcionar una respuesta exacta o basta con una respuesta aproximada y la dimensionalidad del espacio métrico considerado.

Como los conjuntos de interés son los de datos no estructurados, los volúmenes de información con los que se debe trabajar (millones de imágenes en la Web) hacen necesario que los índices sean almacenados en memoria secundaria. En este caso, para lograr eficiencia, no sólo se debe considerar que realicen el menor número de cálculos de distancia sino también, que efectúen una menor cantidad de operaciones sobre el disco (E/S), debido a su costo.

Así, esta línea se dedica a diseñar índices adaptados para memoria secundaria, cuyo desempeño en las búsquedas sea adecuado. Éste es el caso de los nuevos índices basados en la *Lista de Clusters*(*LC*) [5, 4] que son totalmente dinámicos, es decir, admiten inserciones y eliminaciones de objetos y están especialmente diseñados para trabajar sobre grandes volúmenes de datos [13]. La *Lista de Clusters Dinámica* (*DLC*), que tiene una buena ocupación de página y operaciones eficientes tanto en cálculos de distancia como en operaciones de E/S, logra un buen desempeño en espacios de alta dimensión. Sin embargo, la necesidad de recorrer toda la lista de centros de los clusters durante las búsquedas, eleva sus costos. Por ello, el *Conjunto Dinámico de Clusters* (*DSC*), aunque también mantiene los clusters en memoria secundaria, organiza los centros de clusters en un *DSAT* en memoria principal, para lograr búsquedas con menos cálculos de distancia y accesos páginas/clusters. La información en el *DSAT* permite mejorar los costos en cálculos de distancia, y mantener bajos los costos de acceso a disco.

El *DSC* basa su buen desempeño en la calidad de los clusters generados. Por ello, se está analizando una variante para el *DSC* en la cual, en lugar de insertar los elementos en el índice a medida que van llegando, se demora la incorporación de cada nuevo objeto a un clúster, hasta tener varios elementos y poder determinar un mejor agrupamiento de los mismos, que mejore los costos de búsqueda, al lograr clusters más compactos y que aseguren una total ocupación de la página del disco, achicando el tamaño del archivo y reduciendo los tiempos de acceso. Además, permitiría reducir el costo de construcción por amortizar el costo de la escritura de *un* clúster en disco, luego de varias inserciones. Esta idea dio lugar a la *Buffered On Line Dynamic List of Clusters* (BOLDLC), una variante de la *Dynamic List of Clusters* (DLC), que agrega un espacio en memoria para mantener los objetos que son insertados (bolsa), además de un índice auxiliar (de pivotes) sobre estos elementos. Como pivotes se eligen elementos de la bolsa, y cuando ésta se llena, se selecciona el pivote que necesita el menor radio de cobertura para encerrar la cantidad de elementos que caben en un clúster y todos éstos se sacan de la bolsa y se graban en un clúster completo. La grabación de clusters llenos mejora la cantidad de E/S, pero puede provocar que objetos cercanos, que se insertan en diferentes momentos, queden en clusters diferentes, ocasionando degradación en las búsquedas. Por ello, se está estudiando una variante de la BOLDLC que considera grabar clusters *casi* llenos, cuya cantidad de objetos depende de un factor de carga $\rho$. Esto permite un espacio libre en cada clúster, que se considera al insertar un nuevo objeto. Primero se verifica si hay clusters que lo pueden incorporar y de ellos se elige al más cercano a él. Si esto no es posible, se lo incorpora a la bolsa y se actualiza el índice de pivotes. Así, los clusters se adaptan a inserciones y se mejoran los costos en las búsquedas.

Además, se está trabajando en el diseño e implementación de una versión paralela del *Conjunto Dinámico de Clusters* (*DSC*) [13]; esperando, no sólo bajar la cantidad de cálculos de distancia y operaciones de E/S necesarias para responder a una consulta, sino también la cantidad de consultas simultáneas que puedan resolverse, aprovechando al máximo las operaciones de E/S que se realicen durante las mismas. Para ello, se buscará aplicar y comparar distintas estrategias de paralelización en él.

Como el *DSC* basa su buen desempeño en la calidad de la partición generada por los centros de sus clusters, otro aspecto a considerar es mejorar la poda en estos agrupamientos. Para ello, es posible seleccionar un conjunto de "pivotes" globales que permitan caracterizar sobre cada uno de los clusters las zonas dentro de las cuales existan efectivamente elementos [11]. Así, es posible mantener para cada clúster información sobre la mínima y la máxima distancia a la que cada pivote encuentra elementos de dicho clúster. Esto permite identificar los "huecos" en la zona del clúster para que luego, durante las consultas, se pueda descartar un clúster si la bola de la consulta no posee intersección efectiva con la zona del clúster que realmente contiene elementos de interés. Los clusters en *DSC* almacenan un centro y un radio de cobertura y en las búsquedas se

recupera desde memoria secundaria cada cluster que intersecta la bola de consulta con centro $q$ y radio $r$. De esta manera, identificando las zonas del clúster que no contienen efectivamente elementos, se puede descartar un cluster, ahorrando cálculos de distancia y lecturas a disco, porque la bola de consulta intersecta al cluster en su zona "hueca".

Una técnica que ha logrado mucho éxito es la de *Algoritmos Basados en Permutaciones* (PBA). Una manera de indexar estas permutaciones es usando un archivo invertido. En [1] se propone acortar las listas de posteo de cada permutante, considerando un parámetro $m_i < K$, donde $K$ es el número de permutantes considerados. Así, en el archivo invertido, cada permutante $p$ mantiene una lista de pares $(u, \Phi)$, siendo $u$ un elemento y $\Phi$ una posición, si la permutación del elemento $u$ tiene al permutante $p$ en la posición $\Phi$ y $\Phi < m_i$. Durante las consultas, se introduce un nuevo parámetro $m_s$, donde $m_s \leq m_i \leq K$ y se consideran las listas de posteo de los permutantes que aparecen en la permutación de la consulta hasta la posición $m_s$. De esta manera se seleccionan los candidatos y se calcula la distancia entre ellos y la consulta $q$, para dar la respuesta.

Aunque el archivo invertido es muy útil para determinar la lista de elementos candidatos, en [1] no se utiliza toda la posible información. Así, en [8] se ha propuesto aprovechar toda la información disponible desde el archivo invertido. Además, para ahorrar espacio, se almacena sólo al objeto $u$ en la lista del permutante $p$, si la permutación de $u$ tiene a $p$ en sus primeras $m_i$ posiciones. Sin embargo, se mantiene además para cada $u$ su permutación acortada a las primeras $m_i$ posiciones. Durante las búsquedas se procede de manera similar, recuperando las listas de los primeros $m_s$ permutantes; pero luego, antes de calcular las distancias entre el objeto de consulta $q$ y los candidatos, se ordenan los candidatos por distancia entre la permutación acortada de $q$ y las de los candidatos; así, los elementos con más probabilidad de aparecer en la respuesta se encuentren primero. Se está evaluando la posibilidad de definir el valor de $m_s$ en función del radio $r$ de la consulta.

Por otra parte, se está estudiando cómo aprovechar los índices sobre conjuntos masivos de datos, como herramienta de apoyo para solucionar un problema de estacionamiento de vehículos.

### DBMS para Bases de Datos Multimedia

A pesar de que las operaciones más comunes sobre bases de datos multimedia son las búsquedas por rango o de $k$-vecinos más cercanos, la operación de *join* por similitud debería brindarse en un sistema administrador para bases de datos multimedia [15].

Hay distintas variantes del join por similitud entre dos bases de datos $A$ y $B$, con $A, B \subseteq \mathcal{U}$, dependiendo del criterio de similitud $\Phi$ utilizado. Si $A = B$, la operación se denomina *auto-join*.

Dadas $A, B \subseteq \mathcal{U}$, se define el *join por similitud* entre $A$ y $B$ ($A \underset{\Phi}{\bowtie} B$) como el conjunto de los pares $(x, y)$, donde $x \in A$ e $y \in B$; es decir, $(x, y) \in A \times B$, tal que $\Phi(x, y)$ es verdadero. Las variantes más conocidas son: el join por rango, el join de $k$-vecinos más cercanos y el join de $k$ pares de vecinos más cercanos; entre otras. Al resolver el join por similitud es posible que ambas, una o ninguna de la bases de datos posean un índice; o que ambas bases de datos se indexen conjuntamente con un índice diseñado para el join. Calcular cualquiera de las variantes del join por similitud de manera exacta es muy costoso [14], entonces hay que analizar cómo obtener más rápidamente una respuesta aproximada al join y de buena calidad.

*PostgreSQL* es el primer sistema de base de datos que permite realizar consultas por similitud sobre algunos atributos, particularmente indexa para búsquedas de $k$-vecinos más cercanos (índices *KNN-GiST*) sobre texto, comparación de ubicación geoespacial, etc. Sin embargo, estos índices proveen plantillas sólo para *árboles balanceados* (*B-tree*, *R-tree*), pero el "balance" no siempre es bueno para los índices para búsquedas por similitud [2]. Por otro lado, no se dispone de este tipo de consultas sobre todo tipo de datos métricos. Así, es importante proveer un DBMS para estos datos y sus operaciones [10].

Como las respuestas a consultas de join suelen ser conjuntos muy grandes de pares de objetos y algunos de ellos muy similares entre sí, se planea introducir sobre los joins la posibilidad de diversificar las respuestas [16]; asegurando así un conjunto más pequeño y diversificado de respuestas útiles y más rápido de obtener. Además, se ha diseñado un algoritmo simple y eficiente que permite responder consultas de join por similitud de $k$ vecinos más cercanos aproximados dentro del mismo conjunto (auto-join), con una razonable precisión en la respuesta [6]. Estos desarrollos permitirán tener un DBMS más aplicable en sistemas de información reales.

## 4. Resultados

Se han publicado en [9, 7] nuevas alternativas para búsquedas aproximadas sobre grandes volúmenes

de datos y en [8] una optimización al archivo invertido de permutaciones. Por otro lado, se está evaluando experimentalmente la versión paralela del índice *DSC*, diseñada para memoria secundaria, que admite inserciones/eliminaciones de objetos y permitirá responder eficientemente lotes de consultas. Además, se encuentran en etapa de implementación las distintas mejoras al *DSC*. Se continúa trabajando en la extensión de *PostgreSQL* para soportar a más tipos de consultas por similitud, sobre distintos tipos de datos, considerar consultas por similitud aproximadas y la diversificación de respuestas para los joins por similitud. Se ha publicado un algoritmo para realizar el auto-join aproximado [6].

## 5. Formación de Recursos

Se están realizando las tesis de Maestría: (1) - "Estructuras Eficientes sobre Datos Masivos para Búsquedas en Espacios (UNSL) , (2) - "Sistema Administrador para Bases de Datos Métricas" (UNSL), (3) - "Índices Métricos – Optimización del DSC usando Cortes de Regiones" (UNSJ) y (4) - "Optimización del BOLDLC por la Mejora de la Densidad y Solapamiento de los Clusters"(UNSJ).

Además, está en desarrollo un trabajo final de la Ingeniería en Computación (UNSL).

## Referencias


[1] G. Amato and P. Savino. Approximate similarity search in metric spaces using inverted files. In *Proc. of the 3rd International Conference on Scalable Information Systems*, InfoScale '08.

[2] E. Chávez, V. Ludueña, and N. Reyes. Revisiting the VP-forest: Unbalance to improve the performance. In *Proc. de las JCC08*, page 26, 2008.

[3] E. Chávez, V. Ludueña, N. Reyes, and P. Roggero. Faster proximity searching with the distal sat. *Information Systems*, 59:15 – 47, 2016.

[4] E. Chávez and G. Navarro. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, 26(9):1363–1376, 2005.

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

[6] S. Ferrada, B. Bustos, and N. Reyes. An efficient algorithm for approximated self-similarity joins in metric spaces. *Information Systems*, 91:101510, 2020.

[7] K. Figueroa and N. Reyes. Permutation's signatures for proximity searching in metric spaces. In *Similarity Search and Applications - 12th International Conference, SISAP 2019*, volume 11807, 151–159,2019. Springer.

[8] K. Figueroa, N. Reyes, and A. Camarena-Ibarrola. Candidate list obtained from metric inverted index for similarity searching. In *Advances in Computational Intelligence*, 29–38, 2020. Springer.

[9] K. Figueroa, N. Reyes, A. Camarena-Ibarrola, and L. Valero-Elizondo. Improving the list of clustered permutation on metric spaces for similarity searching on secondary memory. In *Pattern Recognition*, 82–92, 2018. Springer.

[10] F. Kasián and N. Reyes. Búsquedas por similitud en PostgreSQL. In *Actas del XVIII Congreso Argentino de Ciencias de la Computación (CACiC)*, 1098–1107, 2012.

[11] J. Lokoč, J. Moško, P. Čech, and T. Skopal. On indexing metric spaces using cut-regions. *Information Systems*, 43:1–19, 2014.

[12] G. Navarro. Searching in metric spaces by spatial approximation. *VLDBJ*, 11(1):28–46, 2002.

[13] G. Navarro and N. Reyes. New dynamic metric indices for secondary memory. *Information Systems*, 59:48 – 78, 2016.

[14] R. Paredes and N. Reyes. Solving similarity joins and range queries in metric spaces with the list of twin clusters. *JDA*, 7:18–35, 2009.

[15] C. Rong, C. Lin, Y. N. Silva, J. Wang, W. Lu, and X. Du. Fast and scalable distributed set similarity joins for big data analytics. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 1059–1070, 2017.

[16] L. Santos, L. Carvalho, W. Oliveira, A. Traina, and C. Jr. Traina. Diversity in similarity joins. In *Similarity Search and Applications*, volume 9371 of *LNCS*, 42–53, 2015. Springer.