# Búsquedas por similitud en PostgreSQL

Fernando Kasián, Nora Reyes

Departamento de Informática, Universidad Nacional de San Luis,
Ejército de los Andes 950, San Luis, Argentina.
{*fkasian, nreyes*}@unsl.edu.ar

**Abstract.** Las búsquedas en espacios métricos y los operadores para búsquedas por similitud han sido estudiados y son actualmente material de estudio recurrente debido al auge de datos no convencionales como por ejemplo audio o video disponibles en grandes repositorios de datos. Por lo tanto, surge la necesidad de almacenar y posteriormente consultar dichos datos. A pesar de ello no se encuentran gestores de bases de datos que implementen todos los operadores relevantes sobre datos de estas características, en los cuales tiene mayor sentido la búsqueda por similitud. Así, nuestro trabajo propone desarrollar un gestor de bases de datos, conteniendo datos no estructurados y que sea capaz de responder las operaciones por similitud más comunes sobre estos tipos de datos, basándonos para ello en: PostgreSQL.

**Keywords:** Bases de datos métricas, consultas por similitud, PostgreSQL.

## 1 Introducción

Las décadas pasadas han sido testigos de una alarmante velocidad de crecimiento de los datos disponibles en forma digital, así como un paralelo crecimiento de las capacidades de almacenamiento utilizables a precios moderados. Además, con la evolución de las tecnologías de información y comunicación, han surgido depósitos no estructurados de información. No sólo se consultan nuevos tipos de datos tales como texto libre, imágenes, audio y video, sino que además ya no se puede estructurar más la información en claves y registros. Tal estructuración es muy dificultosa (tanto manual como computacionalmente) y restringe de antemano los tipos de consultas que luego se pueden hacer. Aún cuando sea posible una estructuración clásica, nuevas aplicaciones tales como minería de datos (data mining) requieren acceder a la base de datos por cualquier campo, no sólo aquellos marcados como "claves". Así, la recuperación desde estos repositorios requiere lenguajes de consulta más poderosos, los cuales exceden las capacidades de la tecnología de bases de datos tradicional. Por lo tanto, son necesarios nuevos modelos más generales de base de datos que permitan administrar y buscar en almacenamientos de datos no estructurados.

Recientemente, los Sistemas Administradores de Bases de Datos (DBMS) incorporan la capacidad de almacenar nuevos tipos de datos tales como imágenes u "objetos multimedia", sin embargo la búsqueda se realiza todavía sobre un número predeterminado de claves de tipos numérico o alfabético y muy raramente se los

puede buscar por contenido. Un concepto unificador es el de bases de datos métricas que utiliza el concepto de “búsqueda por similitud” o “búsqueda por proximidad”, es decir buscar elementos de la bases de datos que sean similares o “próximos” a un elemento de consulta dado. Una base de datos métrica es una colección de objetos digitales (de cualquier clase) con una similitud percibida y una manera formal de calcular esta similitud percibida como una métrica. La similitud percibida es provista por la experiencia o por expertos.

Por lo tanto, debido a los cambios constantes en la estructura propia de los datos computacionales, que en general dichos datos se transforman como vectores de características de un espacio multidimensional, se ha debido repensar la forma de resolver las consultas que se hacen sobre ellos y así el enfoque tradicional de búsqueda exacta utilizando claves ha dado lugar a las búsquedas por similitud en espacios métricos, en las que generalmente se evalúa la distancia que existe entre los puntos del espacio multidimensional obtenidos a partir de un vector de características del objeto computacional (por ejemplo una imagen, Figura 1). El hecho de que los vectores de características sobre los cuales se trabaja son de muy alta dimensión representacional hace que soluciones pensadas para Bases de Datos Multidimensionales no sean aplicables por sufrir, en su mayoría, de la maldición de la dimensión, en cambio el desempeño de las soluciones disponibles para Bases de Datos Métricas depende de la dimensión intrínseca del espacio de datos, que en general es mucho menor que la representacional. La *dimensión intrínseca* de un espacio métrico refleja la facilidad o dificultad intrínseca para las búsquedas en dicho espacio métrico.

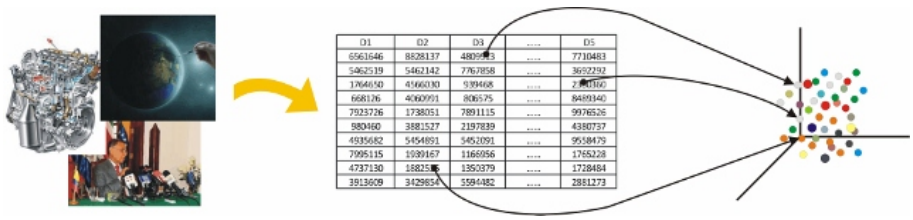


**Fig. 1.** Representación de imágenes en espacios métricos.

Las búsquedas por similitud tienen un campo de aplicación amplio como por ejemplo: recuperación de información, textos o multimedia (audio y/o video) entre otros. Hoy en día cualquier persona utiliza la tecnología para buscar imágenes, música o libros por citar ejemplos concretos de los campos antes mencionados, pero a pesar de ello los gestores de bases de datos más populares no incluyen búsquedas por similitud en espacios métricos, aun cuando existen numerosos trabajos sobre búsquedas y join por similitud en bases de datos métricas.

El presente trabajo se organiza como sigue: en la Sección 2 se describen los conceptos previos, en la Sección 3 se describen las características del gestor de base de datos relacionales PostgresSQL, en la Sección 4 se explican las demás operaciones

de interés, en la Sección 5 se detalla nuestra propuesta y finalmente se dan las conclusiones.

## 2 Espacios Métricos

Un *espacio métrico* está formado por un universo de objetos $\boldsymbol{U}$ y una función de distancia definida sobre dicho universo: $\boldsymbol{d}$**:** $\boldsymbol{U}$ **x** $\boldsymbol{U} \rightarrow \mathbf{R}^{+}$. Un subconjunto finito de él, $\boldsymbol{X} \subseteq \boldsymbol{U}$, de tamaño $n = |\boldsymbol{X}|$, es el conjunto de objetos donde se busca. $X$ será la *base de datos* o simplemente nuestro conjunto de objetos o elementos. La función de distancia además cumple las siguientes propiedades:

1. $\forall x, y \in \boldsymbol{U}$, $d(x,y) \geq 0$ positividad,
2. $\forall x, y \in \boldsymbol{U}$, $d(x,y) = d(y,x)$ simetría,
3. $\forall x \in \boldsymbol{U}$, $d(x,x) = 0$ reflexividad,
4. $\forall x, y \in \boldsymbol{U}$, $x \neq y \Rightarrow d(x, y) > 0$ estrictamente positiva,
5. $\forall x, y, z \in \boldsymbol{U}$, $d(x, y) \leq d(x, z) + d(z, y)$ desigualdad triangular,

si se cumplen estas propiedades entonces decimos que el par (**U**, $d$) es un espacio métrico. En los casos en que la distancia no satisface la propiedad 4 el espacio es llamado un espacio *pseudométrico*, y dependiendo el dominio de aplicación podemos tener un espacio *cuasimétrico* si la propiedad 2 no se cumple.

Los tipos de consultas (o queries en inglés) por proximidad o similitud en espacios métricos de mayor interés son los siguientes:

- *k-Nearest Neighbor query* o $NN_k(q)$ (su forma abreviada): la consulta recupera los $k$ vecinos más cercanos a $q$ *en* $\boldsymbol{X}$.
- *Range query* o $(q, r)_d$ (su forma abreviada): recupera todos los de **X** elementos a un distancia $r$ de $q$, es decir $\{ x \in \boldsymbol{X}, d(x, y) \leq r\}$ (Figura 2).

Las consultas por rango $(q, r)_d$ constituyen el tipo más común de consulta, ya que las consultas de $NN_k(q)$ en general se resuelven como consultas $(q, r)_d$ en las cuales se va ajustando el radio de la consulta a medida que se conoce la distancia a más elementos.

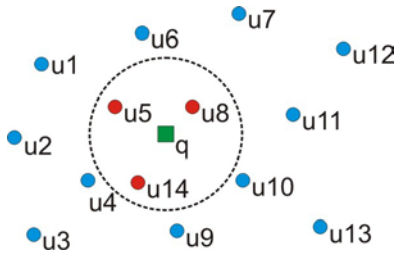


**Fig. 2.** Range query en $\mathbf{R}^2$.

Existen numerosas estructuras para indexar los objetos del conjunto y luego utilizarlas en búsquedas por rango o de los vecinos más cercanos, en dichas

estructuras se utilizan árboles, matrices o agrupamientos (clusters). En general, los algoritmos de indexación particionan el conjunto ***U*** en subconjuntos y construyen un índice que facilita decidir si los elementos incluidos en cada subconjunto son relevantes a la consulta realizada. Estos subconjuntos que se considera que pueden contener objetos relevantes son completamente verificados cada vez que se realiza una consulta. Algunos ejemplos son los árboles: BKT, FQT, FHQT y FQA para distancia discreta, VPT, MVPT, VPF, BST, GHT, GNAT, VT, MT y SAT para distancia continua. AESA y LAESA son ejemplos de estructuras basadas en pivotes y LC para las basadas en particiones compactas, ver [4], [10], [18], [23]. Estas estructuras originalmente propuestas para búsquedas por rango han sido modificadas con técnicas como *incremento de radio*, *retroceso con decremento de radio* o *retroceso por prioridad* para permitir búsquedas de vecinos.

Algunos de estos índices son más adecuados para espacios métricos en los que la dimensión intrínseca es baja (por ejemplo: algoritmos basados en pivotes) y otros sirven principalmente para espacios métricos de mediana a alta dimensión intrínseca o consultas de baja selectividad (por ejemplo: algunos índices basados en particiones compactas como el SAT [15] y LC [3]).

## 3 PostgreSQL

PostgreSQL [5] es un potente motor de bases de datos relacionales reconocido por su fiabilidad, integridad de datos y correcto desempeño, como así también por su alta portabilidad a los principales sistemas operativos Linux, Unix (y sus derivados) y Windows. Su código fuente está disponible bajo licencia de código abierto por lo que es posible su uso, modificación y distribución.

PostgreSQL ofrece entre sus numerosas características:

- Soporte para consultas con UNION, UNION ALL y EXCEPT.
- Outer Joins.
- Sub-selects.
- Cumple con ANSI SQL (implementando el estándar SQL92 y SQL99).
- Integridad referencial.
- Replicación (soluciones comerciales y no comerciales) que permiten la duplicación de bases de datos maestras en múltiples sitios de réplica.
- Interfaces nativas para ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python y Ruby.
- Reglas.
- Vistas.
- Procedimientos almacenados.
- Triggers.
- Unicode.
- Secuencias.
- Herencia: Incluye herencia entre tablas por ello se lo clasifica como gestor objeto-relacional.
- Una API abierta.

- Soporte nativo SSL.
- Lenguajes procedurales.
- Respaldo en caliente.
- Bloqueo a nivel.
- Índices parciales y funcionales.
- Autentificación Kerberos nativa.
- Extensiones para SHA1, MD5, XML y otras funcionalidades.
- Herramientas para generar SQL portable para compartir con otros sistemas compatibles con SQL.
- Sistema de tipos de datos extensible para proveer tipos de datos definidos por el usuario, y rápido desarrollo de nuevos tipos.
- Funciones de compatibilidad para ayudar en la transición desde otros sistemas menos compatibles con SQL.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP, etc.), cadenas de bits entre otros. También permite la creación de tipos propios.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Cumple completamente con ACID (ACID es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español).
- Replicación Sincrónica: permitiendo alta disponibilidad con consistencia sobre múltiples servidores.
- Regionalización por columna: soportando correctamente el ordenamiento por lenguaje en las bases de datos, tablas o columnas.
- Tablas unlogged: importante incremento del rendimiento para datos efímeros.
- Nivel de Aislamiento Serializable a través de "Snapshots": mantiene consistentes múltiples transacciones concurrentes sin el uso de bloqueos, usando verdadera serialización.
- Writeable Common Table Expressions: ejecuta actualizaciones multi-fases complejas en una simple consulta.
- Security-Enhanced Postgres: despliega seguridad de nivel militar y control de acceso mandatorio.
- Indexamiento de los *k* vecinos más cercanos (*k-Nearest-Neighbor*): índices basados en distancias para consultas rápidas de ubicación y búsquedas de texto (Figura 3).

```
CREATE INDEX pgweb_idx ON pgweb USING gin(to_tsvector('english', body));
```

**Fig. 3.** Creación de indices en PostgreSQL.

PostgreSQL además ofrece soporte para una amplia variedad de índices, para los que implementa diferentes estructuras de almacenamiento como B-tree, R-tree, Hash o GiST [9], que es el usado en su versión más reciente para la implementación de búsquedas sobre texto. Estas búsquedas se realizan normalizando el texto y sobre el

texto normalizado buscar la frecuencia de aparición del query y retornar los resultados basándose en dicha frecuencia (Figura 4).

PostgreSQL es el primer sistema de base de datos en haber incorporado la posibilidad de realizar consultas por similitud sobre algunos atributos, particularmente indexación para búsquedas de *k*-vecinos más cercanos (*k*-NN-GiST indexes). Estos índices pueden ser usados sobre texto, comparación de ubicación geoespacial, entre otras. Sin embargo, los índices *k*-NN GiST proveen sólo plantillas para índices cuya estructura es de árbol balanceado, tales como el B-tree o el R-tree y sólo para algunos tipos de datos no estructurados.

```
SELECT to_tsvector('english', 'a fat  cat sat on a mat - it ate a fat rats');
                  to_tsvector
-----------------------------------------------------
 'ate':9 'cat':3 'fat':2,11 'mat':7 'rat':12 'sat':4
```

**Fig. 4.** Ejemplo de búsqueda de texto con PostgreSQL.

## 4 Operaciones en Bases de Datos Métricas

En bases de datos métricas no sólo se está interesado en responder consultas por rango o de *k* vecinos más cercanos. Otras operaciones de interés son distintas variantes del "join por similitud" (ensamble por similitud); es decir, dadas dos bases de datos métricas encontrar todos los pares de objetos (un objeto desde cada base de datos) que satisfacen algún predicado de similitud. Si ambas bases de datos coinciden, se denomina auto-join por similitud (Figura 5).

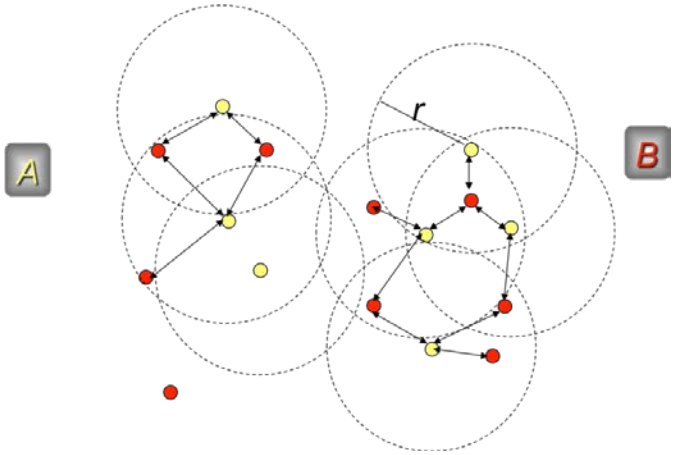


**Fig. 5.** Join por rango entre dos bases de datos en $R^2$.

Algunas variantes del join por similitud son:

- *join por rango*, que dadas dos bases de datos métricas $A$ y $B \subseteq U$ y un radio $r \geq 0$ encuentra todos los pares de objetos a distancia a lo más $r$ entre sí,
- *join de k pares de vecinos más cercanos*, que dadas dos bases de datos métricas $A$ y $B \subseteq U$ selecciona del producto cartesiano los $k$ pares de elementos que se encuentran a menor distancia entre ellos, o
- *join de k* vecinos más cercanos, que dadas dos bases de datos métricas $A$ y $B \subseteq U$ encuentra los pares de vecinos formados por cada elemento de $A$ con sus $k$ vecinos más cercanos en $B$.

Formalmente, el join por similitud generalmente se denota por: $A \bowtie_{\theta S} B = \{(a, b) \mid (a, b)$ satisface $\theta_S$ y $a \in A, b \in B\}$ donde $\theta_S$ es el criterio de semejanza. En el caso particular del join por rango se define como $A \bowtie_r B = \{(a, b) \mid d(a, b) \leq r, a \in A, b \in B\}$.

Numerosos estudios se han realizado sobre el tema con diferentes enfoques, como por ejemplo LTC (lista de clusters gemelos) [17] la cual es un índice especialmente diseñado para responder, de manera eficiente, distintas variantes del join por similitud sobre un par de bases de datos métricas y que a la vez permite responder consultas por similitud sobre cualquiera de las dos bases de datos. Su implementación basada en clusters, en la cual se indexan conjuntamente los elementos de ambas bases de datos, particionando el universo de datos en regiones y se arma la lista de clusters del conjunto A tomando como centro de cada uno de ellos un elemento de B y viceversa, además usa estructuras auxiliares para guardar las distancias entre los elementos del cluster A a los centros del cluster B. Para resolver el join se toma en cuenta cada cluster en la lista y los de la lista gemela. En la Figura 6 se muestra un ejemplo de un par de clusters gemelos para dos bases de datos $A$ y $B$ en $R^2$, donde los objetos de $A$ se agrupan con un centro de $B$ y viceversa.

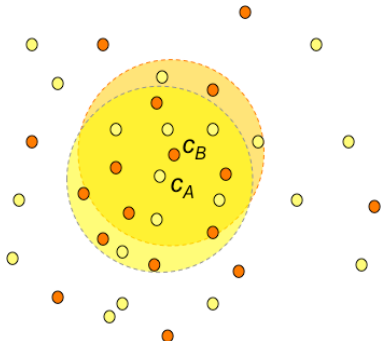


**Fig. 6.** Clusters gemelos para dos bases de datos $A$ y $B$ en $\mathbf{R^2}$

Otros enfoques basan sus algoritmos en Range Join ([1], [6], [8], [11], [12], [13], [19] , [20], [21], [22]), en los que para cada elemento del subconjunto A se compara la distancia con cada elemento del subconjunto B, si la distancia es menor o igual a $r$ (el radio de join) forma parte del conjunto de objetos relevantes a la consulta. Para alguno de estos enfoques se utiliza como estructura auxiliar un D-Index [7] que se basa en el concepto de partición separable mostrado en Figura 7, gracias al uso de una función de división *bps* (ball partitioning split), que se basa en una división que utiliza

la información de la mediana del histograma de distancias del espacio métrico y un parámetro ρ que establece el "ancho" de la zona de exclusión alrededor de la mediana.

En todos los casos mencionados anteriormente se habla de estructuras auxiliares, en las que generalmente se guarda la distancia, esto es porque en las búsquedas por similitud uno de los aspectos que más preponderancia tiene al considerar los costos es la evaluación de la función de distancia.

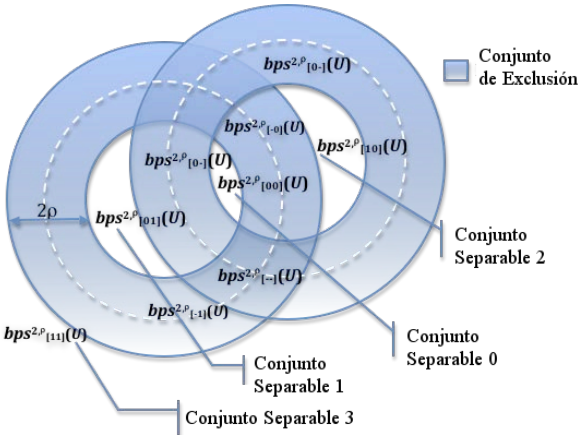


**Fig. 7.** Ejemplo de partición separable usando una función bps.

## 5 Nuestra propuesta

A pesar de que PostgreSQL provee muchas características interesantes en un gestor de base de datos y que ha incorporado la posibilidad de realizar consultas por similitud sobre algunos atributos utilizando los *k*-NN-GiST índices, éstos proveen sólo plantillas para índices cuya estructura es de árbol balanceado y están disponibles sólo para algunos tipos de datos no estructurados. En particular para los índices para búsquedas por similitud la condición de "balanceado" no necesariamente hace más eficientes las consultas [2]. Además, estas consultas por similitud no están disponibles para todo tipo de datos métricos. Por lo tanto, otro aspecto importante en nuestras investigaciones es proveer a un manejador de bases de datos con la capacidad total de manejar bases de datos métricas; es decir, que permita manejar la mayoría de los posibles datos métricos, con las operaciones de interés más comunes sobre estos tipos de datos.

Nuestra propuesta incopora las búsquedas por similitud como parte integral del motor PostgreSQL, proporcionando la funcionalidad y los comandos necesarios para que esto suceda. Para las consultas por similitud básicas: consulta por rango y consulta de *k* vecinos más cercano, se indexa la base de datos, dependiendo de la dimensionalidad intrínseca de la misma:

- *con un índice basado en pivotes (para espacios de baja dimensión) o*
- *con un índice basado en particiones compactas (para espacios de alta dimensión).*

Sin embargo, a la hora de implementar las consultas como el join por similitud, uno de los principales aspectos a tener en cuenta es cómo aplicar el criterio de semejanza o distancia en los operadores de **Join** y **Group by**. Para ello hacemos uso, dependiendo de la situación en que nos encontremos, de:

- Un índice para join, en caso de que sea posible preprocesar las bases de datos para indexarlas conjuntamente con el fin de construir el índice.
- Si ambas bases de datos se han indexado separadamente, algoritmos que calculan el join, aprovechando lo más posible toda la información de distancias disponible desde cada uno de los índices.
- Si ninguna de las bases de datos se encuentra indexada, un algoritmo que minimiza el número de evaluaciones de distancias necesarias para calcular el join, sabiendo que de antemano no se cuenta con ninguna infomación adicional sobre las distancias entre los elementos. En este caso, además de calcular el join, las distancias calculadas se aprovechan a fin de que se indexen una, otra o ambas bases de datos.

En el caso particular del auto-join, las situaciones a analizar son similares, salvo el caso del índice conjunto.

## 6 Conclusiones

Muchos estudios se han realizado sobre gestores de bases de datos relacionales sobre todo a la hora de determinar el desempeño de los algoritmos implementados, pero aún no hay una implementación concreta en las distribuciones disponibles para los usuarios, quizás puede considerarse como una excepción a PostgreSQL, que incluye funcionalidad para recuperación de texto solamente, aunque se anuncie en su sitio web como búsquedas por el vecino más cercano generales.

Nuestra propuesta brinda un panorama mucho más amplio en un gestor de bases de datos, tanto para la manipulación de distintos tipos de datos no estructurados como de las diferentes operaciones de interés sobre estos tipos de datos. En nuestro caso, por ser concientes del tipo de datos y del tipo de consultas que son de interés, se brinda un sistema capaz de ser utilizado en distintos escenarios en los que sea adecuado.

Sin embargo, actualmente se cuenta con un gestor capaz de trabajar con bases de datos estáticas, es decir en las que se conocen de antemano sus elementos antes de hacer consultas. Por lo tanto, se planea en el futuro considerar la posibilidad de que los índices puedan construirse incrementalmente y que sean eficientes para grandes volúmenes de datos [14], [16] y que en el caso de los joins por similitud al agregar elementos a una o ambas bases de datos, la respuesta pueda actualizarse sin volver a calcular nuevamente todos los pares.

Otro aspecto de interés futuro es hacer las búsquedas más eficientes, considerando grandes volúmenes de datos, gracias a a la aplicación de técnicas de computación de alto desempeño. En particular, la facilidad de contar con procesamiento paralelo de costo accesible en las placas gráficas de una computadora de propósito general, da lugar a que se trate de aprovechar esta posibilidad de cómputo para acelerar las consultas, ya sea de join o de búsquedas (por rango como de *k* vecinos más cercanos).

## 7 Referencias

1. F. Angiulli and C. Pizzuti. An approximate algorithm for top-k closest pairs join query in large high dimensional data. *Data & Knowledge Engineering (DKE)*, 53(3):263–281, 2005.
2. E. Chávez, and V. Ludueña, and N. Reyes: Revisiting the {VP}-Forest: Unbalance to Improve the Performance. Proc. de las Jornadas Chilenas de Computación 2008 (JCC08), p. 26.
3 E. Chávez and G. Navarro. A compact space decomposition for effective metric indexing. Pattern Recognition Letters (PRL), 26(9):1363–1376, 2005.
4. E. Chavez, G. Navarro, R. Baeza-Yates, and J. Marroquín: Searching in Metric Spaces. ACM Computing Surveys, Vol. 33, N°3 pp. 273-321.
5. Documentación del Servidor de Base de Datos PostgreSql: http://www.postgresql.org/docs/. Último acceso el 17 de julio de 2012.
6. V. Dohnal, C. Gennaro, P. Savino, and P. Zezula; Similarity Join in Metric Spaces. F. Sebastiani (Ed.): ECIR 2003, LNCS 2633, pp. 452–467.
7. V. Dohnal, C. Gennaro, P. Savino, P. Zezula: D-Index: Distance Searching Index for Metric Data Sets. To appear in Multimedia Tools and Applications, Kluwer, 2002
8. V. Dohnal, C. Gennaro, and P. Zezula. Similarity join in metric spaces using eD-index. In *Proc. 14th Intl. Conf. on Database and Expert Systems Applications (DEXA'03)*, LNCS 2736, pages 484–493, 2003.
9. Gist Indexing Project: http://gist.cs.berkeley.edu/. Último acceso el 17 de julio de 2012.
10. M. Hetland: The basic principles of metric indexing. In Swarm Intelligence for Multi-objective Problems in Data Mining, Studies in Computational Intelligence, vol. 242, pp. 199-232. Springer (2009).
11. E. Jacox, and H. Samet. 2008. Metric space similarity joins. ACM Trans. Datab. Syst. 33, 2, Article 7.
12. D. V. Kalashnikov and S. Prabhakar. Similarity join for low- and high- dimensional data. In *Proc. 8th Intl. Conf. on Database Systems for Advanced Applications*, pages 7–16, 2003.
13. M. Lopez and S. Liao. Finding k-closest-pairs efficiently for high dimensional data. In *Proc. 12th Canadian Conf. on Computational Geometry (CCCG'00)*, pages 197–204, 2000.
14. M. Marin, G.V. Costa, and C. Bonacic. A search engine index for multimedia content. In *Proc. 14th European Conf. on Parallel and Distributed Computing (EuroPar'08)*, LNCS 5168, pages 866–875, 2008.
15. G. Navarro and N. Reyes. Dynamic spatial approximation trees. Journal of Experimental Algorithmics, 12:1–68, 2008.
16. G. Navarro and N. Reyes. Dynamic spatial approximation trees for massive data. In Procs. of the 2nd Int. Conf. on Similarity Search and Applications, 81–88. IEEE Comp. Society, 2009.
17. R. Paredes and N. Reyes: Solving similarity joins and range queries in metric spaces with the list of twin clusters. Journal of Discrete Algorithms 7 (2009) 18–35
18. H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, New York, 2006.
19. Yasin N. Silva, Walid G. Aref, Mohamed H. Ali: Similarity Group-by. In Proceedings of the 2009 Int. Conf. on Data Engineering (ICDE '09). IEEE Computer Society, 904-915. 2009.
20. Yasin N. Silva, Walid G. Aref, Mohamed H. Ali: The Similarity Join Database Operator. Data Engineering (ICDE), 2010 IEEE 26th International Conference on , vol., no., pp.892-903, 1-6 March 2010.
21. Yasin N. Silva, Walid G. Aref, Paul Larson: SimDB: A Similarity-aware Database System. In Proc. of the Int. Conf. on Management of data (SIGMOD '10). ACM, 1243-1246.
22. Yasin N. Silva: Similarity-aware Query Processing and Optimization. VLDB 2009 PhD Workshop, 2009 (6 pages).
23. P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search - The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer, 2006.