Approximate Searches

Similarity searches in Postgresql using metric spaces

Enrico Pirozzi
www.psql.it
scotty@psql.it
info@enricopirozzi.info

PGCON 2008 - OTTAWA
We are going to talk about

- Searches with exact match
- Metric Spaces
- Approximate Searches
- Edit Distance
- Example
- Pivoting - Indexing
- Future issues
Requirements to Approx searches

- Postgresql
- C language
- SQL language
- PL-pgsql language
Exact Match

Select * from customers where name = 'Enrico'

Result:

<table>
<thead>
<tr>
<th>name</th>
<th>address</th>
<th>town</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enrico</td>
<td>2, red street</td>
<td>Ottawa</td>
</tr>
</tbody>
</table>

There is match!!
Exact Search : when can I use it?

It make sense to use an exact search when we presume that our result is inside our database -> .....where name='Enrico'

EXACT SEARCH

MATCH

NULL
Other kind of search?

Images Search

Sounds Search

Economic Search

DNA Search

PGCON 2008 - OTTAWA
Other kind of search?

Similarity Searches
Similarity Searches

Some Examples:

- Text retrieval
- Image searches
- Sound searches
- Other .....
In the exact searches we think our dataset as a set of rows and we make searches inside it.

When we make a similarity search we have to think our dataset as a set of objects. The query is an object too that can belong or not belong to the dataset.
The “similarity search” is the search of objects that belong to the dataset and that are closer to the query object.
Edit distance

Edit distance or Levenshtein distance:

The Edit distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character.
Edit distance

For example the distance between "kitten" and "sitting" is 3

- kitten $\rightarrow$ sitten (substitution of 's' for 'k')
- sitten $\rightarrow$ sittin (substitution of 'i' for 'e')
- sittin $\rightarrow$ sitting (insert 'g' at the end)
Edit distance

Now we can compare:

- Strings

- Every kind of object starting from its features -> Every object has features

- We find an object that has the smallest distance from our query.
Edit distance
Some Problems

Objects can have many dimensions: typical 50-60 dimensions 1 dimension for each feature (for example 3 for RGB images, and so on...)

We can spend much time to calculate the edit distance.
A metric space is a set $S$ with a global distance function (the metric $d$) so that for every two points $x, y$ in $S$, returns the distance between them as a nonnegative real number $d(x, y)$. 
The distance function $d(x,y)$ must be:

- **non-negative**: $d(x,y) \geq 0$
- **Strictly Positive**: $d(x,y) = 0$ iff $x = y$
- **Symmetric**: $d(x,y) = d(y,x)$
- **Have to satisfy the triangle inequality**: $d(x,z) \leq d(x,y) + d(y,z)$
Metric Spaces

Database objects are seen as points in a metric space
Query point can belong to the dataset

Multimedia Dataset
Query point can not belong to the dataset

Multimedia Dataset
Objects and queries are seen as points in a multidimensional metric space.
Metric Spaces: Searches

- Nearest neighbor: search of the object more close to the query point
- K - Nearest neighbor: search of the K objects more close to the query point
- Range query: search of objects that are inside the circle with a given radius r and center in query point q
The point p1 is the nearest neighbor for the query point q
SELECT parola, editdistance (parola, 'contorno') from parole order by 2 limit 5;

<table>
<thead>
<tr>
<th>parola</th>
<th>editdistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonno</td>
<td>4</td>
</tr>
<tr>
<td>comparsa</td>
<td>5</td>
</tr>
<tr>
<td>donna</td>
<td>5</td>
</tr>
<tr>
<td>volto</td>
<td>5</td>
</tr>
<tr>
<td>quattro</td>
<td>5</td>
</tr>
</tbody>
</table>

K – Nearest Neighbor: an example
Range Query

Points p1, p8, pd are inside the circle.
Range Query: an example

SELECT parola, editdistance (parola, 'contorno') from parole where editdistance(parola,'contorno') <= 5;

<table>
<thead>
<tr>
<th>parola</th>
<th>editdistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonno</td>
<td>4</td>
</tr>
<tr>
<td>comparsa</td>
<td>5</td>
</tr>
<tr>
<td>quattro</td>
<td>5</td>
</tr>
<tr>
<td>volto</td>
<td>5</td>
</tr>
<tr>
<td>donna</td>
<td>5</td>
</tr>
</tbody>
</table>

quattro terre comparsa donna scomparso nonno volto quattro
Similarity searches

- Instead of words we can use any kind of strings
- We can compare n-ple of values \((a_1, a_2, \ldots, a_n)\) that represents features of objects.

It is possible to make similarity searches between any kind of objects.
The dark side

- High dimensional spaces typical 50,60 dimensions for each object

Problems:

- A lot of memory
- A lot of time to calculate edit distance
The triangle inequality

\[ d(q,p_1) \leq d(q,P_v) + d(P_v,P_1) \]
The triangle inequality

Let \((X,d)\) be a metric space, where \(X\) is the universe of valid objects and \(d\) is the metric of the space, and let \(U\) a subset of objects of \(X\) \(|U|=n\) \(U\) is our database.

\((q,r) = \{u \text{ that belongs to } U \text{ so that } d(u,q) \leq r\} \) Range Query

Given a query \((q,r)\) and a set of \(k\) pivots \(\{p_1, \ldots, p_k\}\) by the triangle inequality it follows that \(d(p_i,x) \leq d(p_i,q) + q(q,x)\), and also that \(d(p_i,q) \leq d(p_i,x) + d(x,q)\) for any \(x\) that belongs to \(X\). From both inequalities, it follows that a lower bound on \(d(q,x)\) is \(d(q,x) \geq |d(p_i,x) - d(p_i,q)|\). The objects \(u\) of interest are those that satisfy \(d(q,u) \leq r\), so all the objects that satisfy the exclusion condition can be excluded, without actually evaluating \(d(q,u)\)

\(|d(p_i,u) - d(p_i,q)| > r\) for some pivots \(p_i\).
Do you need some coffee?

Are you still alive?

Yes?

Ok, Now we will enjoy :)
Building an Index - Pivoting

Some examples:

\[ d(P_v, P_4) = 1 \]
\[ d(P_v, P_3) = 2 \]

\[ \ldots \text{ and so on} \]

We can choose a point as pivot point and when we insert a new item we can pre-calculate the distance between our pivot and the new point.
We can store our informations in an index ordered by distance between the pivot and the other point of the metric space.
Building an Index – Range Query

Q is our query point
We want to find all the points such that
\[ d(q, p_i) \leq 1 \]
Building an Index – Range Query

Q is our query point
We want to find all the points such that $d(q, p_i) \leq 1$
Building an Index – Range Query

\[ d(q, pv) = 3 \]

\[ r = 1 \]

\[ d(Pv, Pi) - d(q, Pi) \leq 1 \]
Building an Index – Range Query

We center our search in $d(P_v,q) = 3$ and choose our candidates point inside the interval between $[d(P_v,q)-r, d(P_v+r)]$

Our candidates are $P_2, P_7, P_3, P_5$

After calculating the edit distance among those points and $q$, we will see that $P_3$ is in the result set.
If we have 2 or more pivots we consider as candidate points all the points that are in the intersections of the distance calculated among pivots.
Building an Index - 2 Pivots

P5 and P7 belong to the intersection
P5 and P7 are the candidates point
Building an Index - 2 Pivots

An example:

Range query
query point $q$
radius $r = 2$

$P_5$ belong to the circle with center $q$ and radius $r$
Building an Index - 2 Pivots

P5 is a candidate point and P5 is inside the circle with center q and radius r.
Other features

- We can implement an algorithm also for K-NN queries using our index structure.
- We can use an approximated edit-distance function (using AC or PAC algorithms) to minimize computational time.
Status of work

On Pg-foundry contrib Pg-edist

Approximated searches
Edit distance

Index for metric spaces to improve range queries and K-NN queries

It already works, but it is written in C Language and it is not yet present on PostgreSQL
I want to extend Pg-edist with the C code written about the index structure
I'm alone

And I hope that someone in the community will join my project, because it is an hard and big project, but I think that it is a very interesting one.
The End: we talked about

- Metric Spaces
- Approximate Searches
- Edit Distance
- Example
- Pivoting - Indexing
- Future issues
Bibliography

- My thesis: http://www.enricopirozzi.info
- Searching in metric spaces
  http://citeseer.ist.psu.edu/avez99searching.html
- Spaghettis: An Array Based Algorithm for Similarity Queries in Metric Spaces
  http://citeseer.ist.psu.edu/414510.html
- Overcoming the Curse of Dimensionality
  http://citeseer.ist.psu.edu/407814.html
- Fixed Queries Array: A Fast and Economical Data Structure for Proximity Searching
  http://citeseer.ist.psu.edu/ch01fixed.html
Contact Information

- Web Site: www.psql.it
- Email: scotty@psql.it
- Personal pages: www.enricopirozzi.info
- Email: info@enricopirozzi.info
- Skype contact: sscotty71
- Gtalk contact: sscotty71@gmail.com
This talk is copyright 2007 Enrico Pirozzi, and is licensed under the creative commons attribution license.