PostgreSQL … in 5 minutes

CYBERTEC
The PostgreSQL Database Company

PostgreSQL Fuzzy Searches in 5 minutes

In many cases people grow frustrated because they are not able to find entries in a database due to typos or some other kind of error. Clearly, a GUI is a lot more useful if it supports fuzzy searching. To do that PostgreSQL provides a simple mechanism called "trigrams". Here is how it works - in 5 minutes:

Let's start

First of all we can create a little table containing just one column. The goal of this example is to load the name of all Austrian cities and villages and do fuzzy matching:

```
fuzzy=# CREATE TABLE t_location (name text);
CREATE TABLE
```

Once we have created the table we can load the data. PostgreSQL is able to read directly from a pipe and store data in a table. Here is how it works:

```
fuzzy=# COPY t_location FROM PROGRAM 'curl www.cybertec.at/secret/orte.txt';
COPY 2354
```

We use a command line tool called "curl" to load the data directly from the web.
The data looks like this:

```
fuzzy=# SELECT * FROM t_location;
           name
--------------------------------
 Eisenstadt
Rust
...
```

To do fuzzy searching we have to enable an extension called pg_trgm which is part of the PostgreSQL distribution (contrib):

```
fuzzy=# CREATE EXTENSION pg_trgm;
CREATE EXTENSION
```

Then we can already create an index on this column. Note that we have to create a so called Gist index which is capable of performing so called "KNN-search" (= Nearest Neighbor Search):

```
fuzzy=# CREATE INDEX idx_trgm ON t_location USING gist(name gist_trgm_ops);
CREATE INDEX
```

pg_trgm will provide us with a "distance operator" (<->). We can use this operator to find those strings, which are closest to what we are looking for. In our example we want to find the 3 closest strings:

```
fuzzy=# SELECT * FROM t_location ORDER BY name <-> 'krammatneusiedel' LIMIT 3;
     name
----------------
 Gramatneusiedl
 Klein-Neusiedl
 Potzneusiedl
(3 rows)
```

The cool thing about this is that the query can actually use an index making things supersonic fast:

```
fuzzy=# explain SELECT * FROM t_location ORDER BY name <-> 'krammatneusiedel' LIMIT 3;
                              QUERY PLAN
-----------------------------------------------------------------------------------------
 Limit  (cost=0.14..0.40 rows=3 width=13)
   ->  Index Scan using idx_trgm on t_location  (cost=0.14..203.22 rows=2354 width=13)
         Order By: (name <-> 'krammatneusiedel'::text)
(3 rows)
```

Trigrams are especially useful if you want to index words, names, addresses and so on.