Instructions for the first project in course Advanced databases

SQLText search and advanced SQL

2016/2017

Develop an application (web/desktop/mobile) that will enable three things:

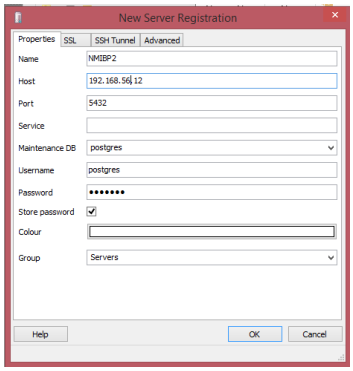
1. Insertion of the text content into PostgreSQL database containing an arbitrary data scheme (minimal but rich enough for the demonstration of all project exercises)
2. Search the text content stored in the database
3. Analysis of submitted queries in a given period of time

Application design is arbitrary as well as the choice of technologies you will use to build it.

Using PostgreSQL DBMS during work on your project

We recommend (but it is not mandatory) that you use (as a text/document repository) PostreSQL system installed on the virtual machine intended for exercises in this course.

After you have started the virtual machine (following the instructions on the course website), you can run SQL queries against PostgreSQL DBMS using Linux ***psql*** command. However, it is much simpler to use some of SQL editors customized for working with PostgreSQL, e.g. PgAdmin. It can be downloaded from http://www.pgadmin.org/download/.



Using PgAdmin define settings (as pictured on the right) to connect to PostgreSQL server installed in a virtual machine.
Use options: *File - Add Server*

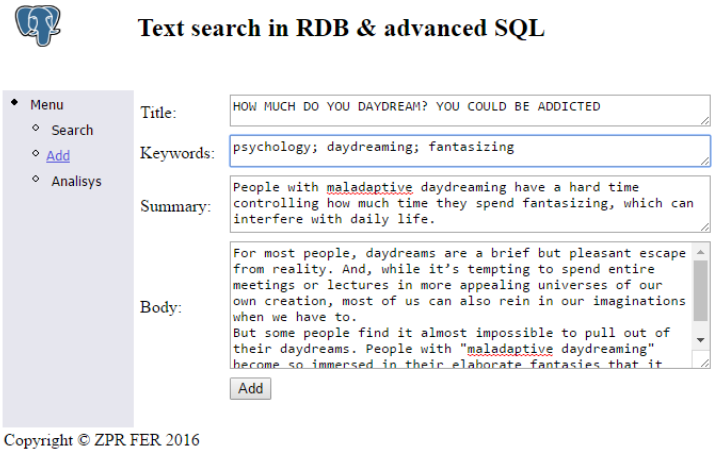
Connect to the PostgreSQL DBMS. Create a new database (right-click on Databases - New Database; Name: *asYouWish*, Owner: *postgres*). Refresh the content of the Databases sub-tree (right-click on Databases - Refresh), select the database you want to work with and open a query window (Tools - Query Tool).

Set the format of the Date data type to *dd.mm.yyyy* performing the following command:

```
set DateStyle ='German, DMY';
```

1. **Insertion of the text-type content in the database**

Store content in a PostgreSQL database. Use texts in the English language due to full support for Full Text Search and Fuzzy Text Search.



2. **Searching the text content stored in the database**

You should implement two types of searches:

a. Retrieval of documents containing the sample(s) in normalized form- **Based on morphology&semantic**

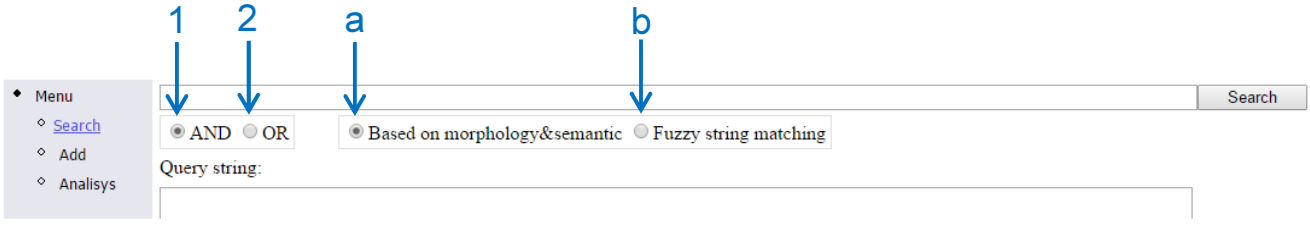
   It is necessary to retrieve documents containing any form of words from the search text, i.e. it is necessary to normalize words, remove stop words, etc.

b. Retrieval of documents containing string(s) similar to pattern(s) - **Fuzzy string matching**

   In this case, retrieval of documents containing somewhat changed search condition is fine - e.g. for search condition 'Haven from', the document 'A fast Trip from Heaven' should also be returned as a result (note that Haven and Heaven do not have the same word lexeme and that it does not matter in this case).

Support joining the given search samples by Boolean operator

1. AND
2. OR

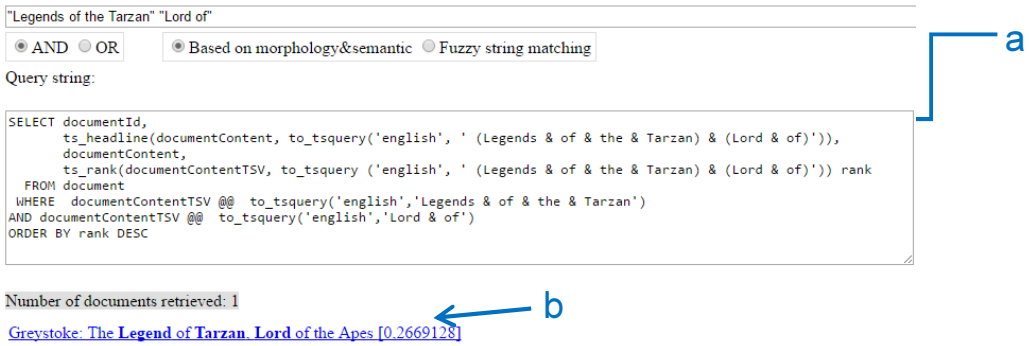


Model the database so that you speed up the search as much as you can - i.e. by additionally storing normalized text, creating special indexes (as many as, and any which may speed up any kind of search).
To implement searches a) and b) use any PostgreSQL operators and functions you find suitable. You must be able to corroborate your choices.

In the application it is necessary to:

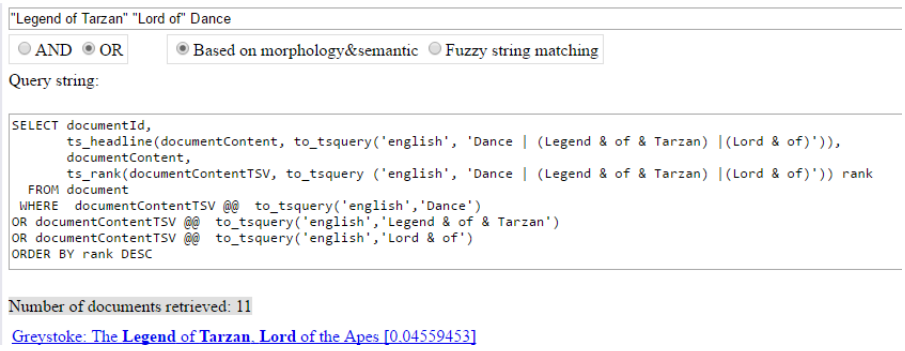
a. display SQL query text used to retrieve documents (SQL search string)

b. display informative data for the document to the user. Searched words found in document that qualified it as the result should be bolded. Also, display the document **rank**. Note that the words Legend Tarzan and Lord are bolded in the document shown in the picture.



Study how the document ranking functions implemented in PostgreSQL work (recommended material is http://shisaa.jp/postset/postgresql-full-text-search-part-3.html) and in your application use the most suitable ranking function with the parameters you find most suitable. Try to apply more sophisticated approach of implementing ranking functions than the one shown in the picture in which function ts_rank was called with the "default" parameters.

c. enable phrase search (multiple strings within quotation marks) and "simple" words combined with Boolean operators AND and OR.
The picture below shows one of the ways to find documents containing phrases "Legend of Tarzan" or "Lord of" or word Dance.



3. Analysis of queries submitted in a given period of time

To make the analysis possible it is necessary to log the queries (including submission timestamp) provided by users. It usually has to be taken into consideration during both database and application design. Assess what parameters about the submitted queries would be wise to record.

It is necessary to allow to the user:

1. specifying a time period in which the analysis should be conducted, in the form:
   date from – date to (e.g. 10/10/2014 -13/10/2014)
2. specifying the granularity of analysis:
   a. day or
   b. hour

Using the advanced features of SQL (**pivoting**) produce a report containing the number of submitting of a specific query for the specified period (e.g. 10/10/2014 -13/10/2014) per days or per hours, based on what the user has chosen.

Per days:

	querystring character(200)	d10102016 integer	d11102016 integer	d12102016 integer	d13102016 integer
1	'Dance' & 'Legend of Tarzan' & 'Lord'	4	3	2	
2	'Lord' & 'Dance'	3	2	2	

or per hours:

	querystring character(200)	s00_01 integer	s01_02 integer	s02_03 integer	s03_04 integer	s04_05 integer	s05_06 integer	s06_07 integer	s07_08 integer	s08_09 integer	s09_10 integer	s10_11 integer	s11_12 integer	s12 inte
1	'Dance' & 'Legend of Tarzan' & 'Lord'								1		3	3	1	
2	'Lord' & 'Dance'										3	3	1	

Help: To be able to use Full Text Search and Fuzzy Text Search functions as well as pivoting functions, you must include modules `fuzzystrmatch`, `pg_trgm` and `tablefunc` in the database you will use in project. You need to include these modules in each database you intend to use them in. They can be "registered" performing the following commands:

```
CREATE EXTENSION fuzzystrmatch;  -- (soundex, levenshtein, metaphone)
CREATE EXTENSION pg_trgm;        -- (similarity , show_Trgm,…, %, <->)
CREATE EXTENSION tablefunc;      -- (crosstab)
```

To display concise information about retrieved documents with the bolded keywords you may use function *ts_headline.*

Upload your solution of the project to your own folder **NMiBP\P1**. Upload in the root directory readme.txt file containing:

- database schema you used along with all the indexes you created
- short description of technologies you users (one to two sentences)
- instructions for running your solution

Half-done solutions will also be graded.

During the project presentation students may be asked to:

- explain a segment of their solution
- explain a concept from the text search topic or pivoting (e.g. how does an operator or function they used work)
- explain why they used a particular operator or function, and not some other
- run i.e. demonstrate their solution (and e.g. submit and run a query)
- etc.

The deadine for uploading your solution is: Thursday 27/10/2016 before 8:00 a.m.