Database tables

```
mysql> describe query;
+---------+------------------+------+-----+---------+----------------+
| Field   | Type             | Null | Key | Default | Extra          |
+---------+------------------+------+-----+---------+----------------+
| id      | int(15) unsigned | NO   | PRI | NULL    | auto_increment |
| type    | char(1)          | NO   |     | r       |                |
| urlargs | text             | NO   | MUL | NULL    |                |
+---------+------------------+------+-----+---------+----------------+

mysql> describe user_query;
+----------+------------------+------+-----+-------------+-------+
| Field    | Type             | Null | Key | Default     | Extra |
+----------+------------------+------+-----+-------------+-------+
| id_user  | int(15) unsigned | NO   | MUL | 0           |       |
| id_query | int(15) unsigned | NO   |     | 0           |       |
| hostname | varchar(50)      | YES  |     | unknown host |       |
| date     | datetime         | YES  |     | NULL        |       |
+----------+------------------+------+-----+-------------+-------+
```

Notes:

* Ignore the **type** column of the query table for now.
* Ignore the **hostname** column of the user_query table for now.
* query's **id** field is the same as user_query's **id_query** field (~ foreign key).
* user_query's **id_user** field is the same as user's id field (~ foreign key). Ignore the user table for now and consider the **id_user** field as unique users.
* query's **urlargs** field is the 'query' part of the search URL, for example: '`ln=en&sc=1&`**`p=author%3Aellis`**`&`**`f=`**`&action_search=Search&`**`c=Articles+%26+Preprints`**`&`**`c=Books+%26+Reports`**`&`**`c=Multimedia+%26+Arts`**' (the parts in **bold** are the interesting ones). It's 'quoted' so you might want to 'unquote' it to make more sense of it: '`ln=en&sc=1&`**`p=author:ellis`**`&`**`f=`**`&action_search=Search&`**`c=Articles+&+Preprints`**`&`**`c=Books+&+Reports`**`&`**`c=Multimedia+&+Arts`**' (check python's standard `urllib2` module for that). Interesting search parameters will be extracted from this field.
* user_query's **date** field has the information about when the query was executed and has the following format: '`2013-07-30 11:41:30`' (or `datetime.datetime` instance in python).

Python code

```
# import the dependencies
from invenio.dbquery import run_sql
import gzip, cPickle

# fetch query and user_query data from the DB
query = run_sql("select id, urlargs from query order by id")
user_query = run_sql("select id_query, id_user, date from user_query order
by date")

# convert query from a tuple to a dictionary
# ...

# write query data to a cPickled and gzipped file
f = gzip.open('query_dict.data','wb')
cPickle.dump(query, f)
f.close()

# read query data from a cPickled and gzipped file
f = gzip.open('query_dict.data','rb')
query = cPickle.load(f)
f.close()
# query is basically a dictionary, the query ids are the keys and the
urlargs the values

# write user_query data to a cPickled and gzipped file
f = gzip.open('user_query_tuple.data','wb')
cPickle.dump(user_query, f)
f.close()

# read user_query data from a cPickled and gzipped file
f = gzip.open('user_query_tuple.data','rb')
user_query = cPickle.load(f)
f.close()
# user_query is basically a tuple of tuples, each being a row of the
user_query DB table: ((id_user, id_query, date), …)
```

Project tasks

* Parse user_query, which contains all the user searches sorted by date.
* For each user search, fetch and analyze the urlargs.
* For each urlargs, extract the ‘interesting’ search parameters.
* Produce the necessary visualization data.
* We should have different ‘views’ on the visualization data (perhaps several files?):
    * global popular search terms
    * popular searched authors
    * popular searched collections
    * etc
* Optionally, consider if a search term is somewhat popular among many users or very popular among few users. For example, a search term may be used by 10 times by 10 users or 10 times by 2 users.
* Explore different visualization options: the main idea is to visualize the popularity of search terms over the passage of time.
* Once the visualization data format is decided, write a script that periodically updates the data with new user search data.

* Json should have partial data (based on dynamic threshold) (ex first 2000 most used queries)
* Normalized data: (key, value) : key for f or p (foo: bar)
* clean data: maybe discard a few searches (ex: ellis)