Advanced databases
ESIR 2, TP3456

Exercise 1 : Views and Design documents Read the introductory text on views `http://guide.couchdb.org/draft/views.html` and on design documents `http://guide.couchdb.org/draft/design.html` in couchDB. Pay particular attention on how to create permanent views with the help of design documents.

Create a design document to the DBLP database (TP12), with appropriate views, that enable to create some of the queries of TP12. Also formulate the queries as a URL.

Exercise 2 : Collecting Twitter messages Develop an application that uses a noSQL database (couchDB) as a back-end.

- Collect Tweet messages with a predefined filter, such that all the messages should contain geographic locations. (For example, collect all messages with a predefined hashtag).
- Store the messages that you obtain (as JSON documents) to a couchDB database.
- Define meaningful queries on the collected data (for example, list all cities with more than 200 tweet messages, or compute the number of messages per country, or other queries, depending on the data you collect)
- Define design documents for (some of) the queries. Use the URL of the query to display your results.

Beyond the basic functionality, you may add additional features (such as visualizing the results on google maps). However, do not try to add too many features, rather aim to develop well-tested, stable software that is well documented. You should focus on the data management aspects of the application rather than to develop a complete user interface with complex user interactions.

We propose to develop your code in java, but you might chose other languages. Links to some potential APIs that enable connecting your java classes to couchDB. You might use other libraries or components as well. You can decide whether you prefer the REST API of Twitter (simpler) or the streaming APIs (more challenging). In your report explain clearly the architecture of your application.

Exercise 3 : Analyzing a your database Analyse you application.

- Try to understand experimentally the limits of your application: For example try to determine (within the performance limitations of free Twitter API token queries, and your available disc space), what is the maximal rate with which you can manage to store the messages.
- Analyze the query response times, for different (types of) queries. Which (what types of) queries can you answer in the context of user interactions? (that is, the response time remains within seconds, even if for larger volumes of data).