A **First Touch** On **NoSQL** Servers

This practical exercise is intended to make you reason about the differences of principles between “classic” RDBMS and NoSQL servers for building and managing data. DO NOT expect to acquire an extensive practice of the use of the DBMS used here. Rather expect to be able to:

- Point out some of their principles concerning: data model, design process, internal and external data management, architecture;
- Compare these principles with those of classic RDBMS.

Therefore we propose an exercise organized into two parts where you are supposed to run some tests and answer control questions for making sure that you are achieving the expected objective of the exercise.

Requirements

- CouchDB
- cURL

1. Building and Querying a NoSQL Oriented Database

In this exercise you will populate and query a NoSQL database using data coming from the Deezer music service.[1] In particular, you will use data related with the rock band Muse. For instance, the following links give access to Muse’s albums and information about similar artists:[2]

- http://api.deezer.com/artist/705/albums
- http://api.deezer.com/artist/705/related

2. TO DO

2.1 Creating and populating a database

- Using a terminal, **create** the **database** Deezer:

```
curl -X PUT http://localhost:5984/deezer
```

- Download (*and save into files*) the data about **Muse’ album** and **similar artists**:

```
curl -X GET http://api.deezer.com/artist/705/albums > MuseAlbums.json
curl -X GET http://api.deezer.com/artist/705/related > MuseRelatedAlbums.json
```

- **Populate** the Deezer **database** with the retrieved information by issuing the following commands:

1 http://developers.deezer.com/api/

2 Note that Muse has the ID 705 in Deezer

```
curl -X PUT http://localhost:5984/deezer/muse_albums --upload-file "MuseAlbums.json"
curl -X PUT http://localhost:5984/deezer/muse_related_artist --upload-file "MuseRelatedAlbums.json"
```

- **Verify** the content of the database:

```
curl -v http://localhost:5984/deezer/muse_albums
curl -v http://localhost:5984/deezer/muse_related_artist
```

Note that the output includes the HTTP request (*and reply*) headers sent to (*by*) CouchDB.

- Open **Fouton** (*CouchDB Web Interface*) on your browser:

```
http://127.0.0.1:5984/_utils/index.html
```

- Access and observe the database **Deezer** on **Fouton**.

2.2 Querying a database

Execute the following queries:

- **Query 1**
  Retrieve the name and the web page of the groups that are similar to the rock band Muse.

Map
```
function(doc) {  var artists = doc.data;  if(doc._id == "muse_related_artists") {
        for(var i in artists) emit(artists[i].name, artists[i].link);
} }
```

- **Query 2**
  Compute the total number of the albums produced by the rock band Muse (requires to check the ***reduce check button*** in Fouton).

Map
```
function(doc) {  var artists = doc.data;  if(doc._id == "muse_related_artists") {
        for(var i in artists) emit('muse_albums', 1);
} }
```

Reduce
```
function(keys, values, rereduce) {
        return sum(values);
}
```

2.3 Theoretic questions (TO ANSWER)

1) Is it possible to represent data under the classic relational model (see the 1 Normal Form)?
2) Compare the notion of key in a relational schema with respect to the notion of key in key-value NoSQL approaches.
3) Why is it necessary to define views for querying a database in CouchDB? Is this a way of integrating data? Justify.

3. REFERENCES

- CouchDB, the definitive guide
- NoSQL