

**Cassandra Report**

By ALGERA Pieter, BAALI Karim and ALBIZZATI Grégoire

Table des matières

[Dataset and queries 3](#_Toc506238823)

[Loading data from JSON to Cassandra 4](#_Toc506238824)

[Performing queries to the database 4](#_Toc506238825)

# Dataset and queries

For this project on Cassandra database, we decided to choose the dataset named “Companies2.json”. It contains different kind of information about a very large set of company object, such as a name, a website URL or the lists of customers or the number of employees. It allows us to be able to query a lot of different information based on various parameters. For example, a little part of the first object in this dataset :



We decided to create 5 queries to test our dataset on a Cassandra database :

* [EASY] Get the name of companies which have more than N products.

SELECT name FROM companies WHERE number\_of\_products > 1 ALLOW FILTERING

* [EASY] Get the id of companies with a given number of employees.

SELECT name FROM companies WHERE number\_of\_employees = 25 ALLOW FILTERING

* [MEDIUM] Get the number of companies with a given category code.

SELECT name FROM companies WHERE category\_code = 'web' ALLOW FILTERING

* [MEDIUM] Get the name of companies which have offices in a given city.

SELECT name FROM companies where twitter\_username > 'A' and twitter\_username < 'B' ALLOW FILTERING

* [HARD] Get the name of companies which were created before a given date and have more than N providers.

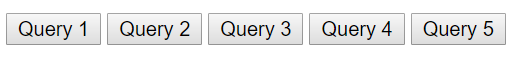
SELECT name FROM companies WHERE founded\_year > 2004 AND total\_money\_raised = '$39.8M' ALLOW FILTERING

# Loading data from JSON to Cassandra

We decided to use Node.js to create our application which load the data from the JSON file to the Cassandra database, using a node module called Cassandra-driver : <https://www.npmjs.com/package/cassandra-driver> . The first issue with our dataset was the presence of a lot of sub-objects in a company object, which were difficult to parse and add in our database. We finally succeed to import our data by using JSON.stringify() on these sub-objects and storing them as a simple string. To avoid loading to much data in our database, we decided to load only 5000 rows in the database, but this number could be modified in the script file simpleCassandra.js by changing the value for the nbRows constant.

# Performing queries to the database

The second part of the work is to be able to perform queries on the database and display the requested data to the user. So we decided to use Express framework to create a quick view to be able to launch the queries by clicking on a button, and display the results in a list.



Each button allows the user to perform a specified query on the Cassandra database. To do so, you need to have Node.js installed. Once the Cassandra server is running, you can run the command “node simpleCassandra.js” which gonna load the data in the database. Then, you can run “node cassandraNode.js” and access to the address localhost:8081 in your web browser, and click on the different buttons available.