University of Tours

Master of Big Data Management and Analytics

Big Data, Cloud Computing and Services Web project

Authors:
Akaich Ines
Nascimento Filho, Jessé .F

Tutors :
Dominique Li

January 21, 2019

Contents

List of Tables

1 Introduction

As part of our master degree 's first year program , we worked on a big data project that aims to verify the learning outcome of several courses of master 1 st year, specifically Big Data, Cloud Computing and Services Web course.
In this project , we work on the scholar data of the university of blois of the students that take part of the computer science department. This department offers Bachelor and master degrees in Computer Science.
Since its creation in 2001, each promotion has 200,000 students per year (including 1,000,000 students each year). In each semester, each student must enroll in the 100 mandatory UEs (of which a total of 1000 mandatory EUs for 10 semesters from L1 to M2) and 100 optional UEs from the 500 optional EUs (including 5,000 optional EUs in total for every 10 semesters).
This project aims to build version 1 of the Administration Intelligence web service of Blois University by providing REST APIs to access the database of schooling data. For the security reason,this version only considers the method GET.

The rest of this document is structured as follow:
We start in section 2 by presenting our database. In section 3 , we describe our workload. In section 4, we introduce our implementation method. In section 5 , we present an installation and user guide. Finally , we end with a conclusion section.

2 The system architecture

In this section we present our system architecture and then our database structure.

2.1 Architecture overview

Our database is an apache HBase that runs on a Hadoop cluster. Clients can access HBase data through a REST gateway using our implemented rest API. Afterward , data is processed using map reduce programming model .
The environment in which this project is implemented :

- Java SDK 1.8.
- Hadoop.
- HBase.
- MapReduce;

- Eclipse+m2e;

An overview of our system architecture is in figure 2.1.0-1.

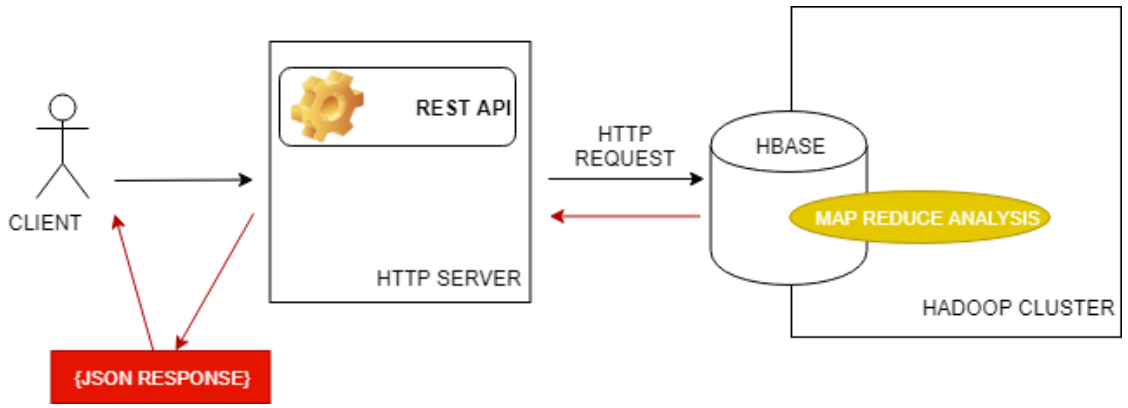

Figure 2.1.0-1: System architecture

2.2 Database Schema Design

Our Hbase database is composed of four tables :

- The student table.
- The course table.
- The instructor table.
- The grade table.

2.2.1 The student table Design

The Student table S (for Student) stores personal information of all students at Blois University that has only the Computer Science department. This table contains a required column family and a contact column family C (for Contact) containing the following columns:

- :F - First name of the student.
- :L - Last name of the student.

- :P - Program of the student, from 1 (L1) to 5 (M2).
- C:B - Birth date of the student, for instance 12/01/1996 or 1996-01-12.
- C:D - Domicile address of the student.
- C:E - E-mail address of the student.
- C:P - Phone number of the student.

A unique student ID with 10 digits (YYYYNNNNNN) is considered as row key, where YYYY is the year of entrance and NNNNNN is a serial ID, for instance 2008000123.

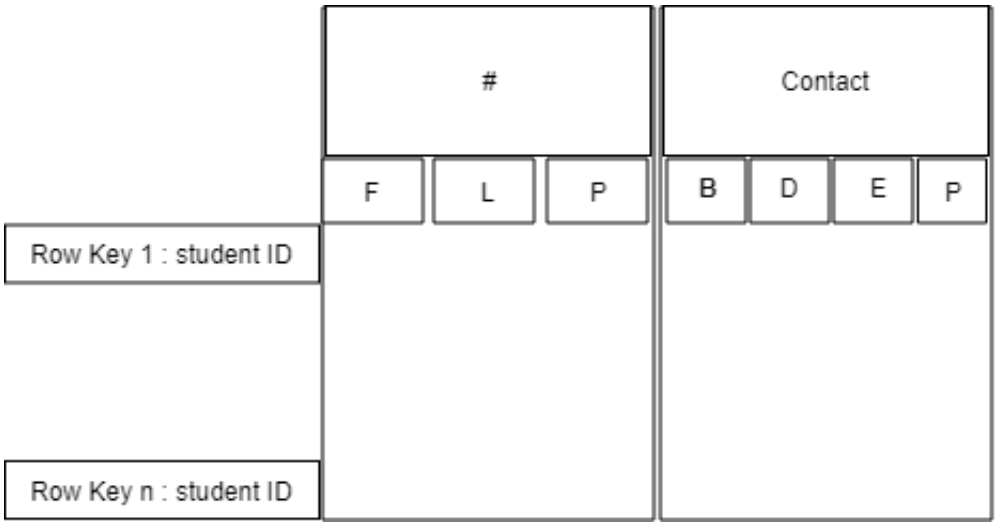

Figure 2.2.1-2: Student Table

2.2.2 The course table Design

The Course table C (for Course) stores course information including course name and instructor list (one course can be instructed by several instructors). This table contains a required column family and an instructor column family I (for Instructor) with the following columns:

- :N - Course name, for instance Big Data and Web Services.

- I:1 - Name of instructor 1 (required).
- I:2 - Name of instructor 2.
- I:n - Name of instructor n.

A course ID can be redistributed to different courses with respect to different years, so we consider a unique course ID associated with a year as row key, for instance S01A007/2015.

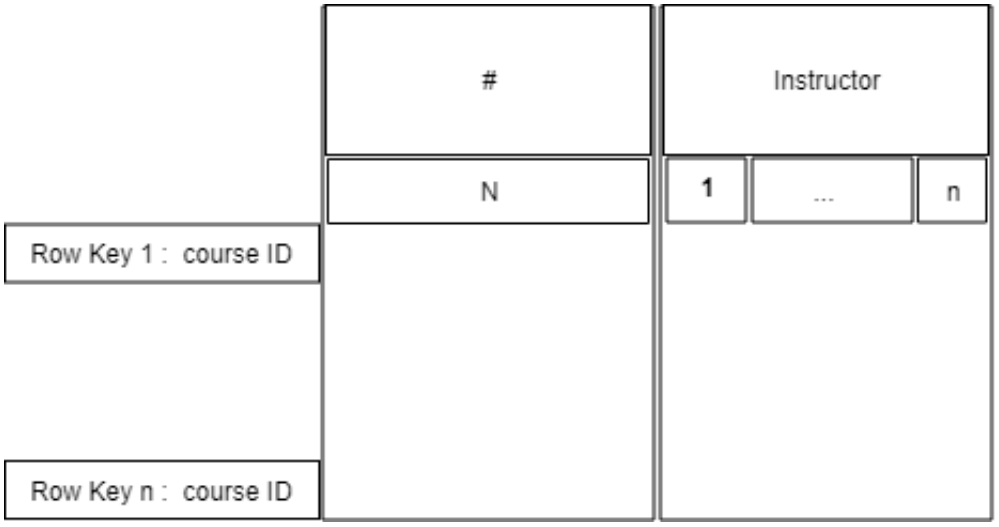

Figure 2.2.2-3: Course Table

2.2.3 The instructor table Design

The Instructor table I (for Instructor) stores all courses instructed by each instructor. Indeed, this table is considered as a helper table in order to make instructor related statistical tasks efficient. This table contains only one column family (for required information) with numbers as columns and with course IDs as values:

- :1 - Course 1 instructed by the instructor (required).
- :2 - Course 2 instructed by the instructor.
- :n - Course n instructed by the instructor.

Since the course list of each instructor is organized by year, we use NAME/YYYY as row key where NAME is the instructor's name (can contain space but cannot contain /) and YYYY is the concerned year, for instance Thomas Devogele/2015.

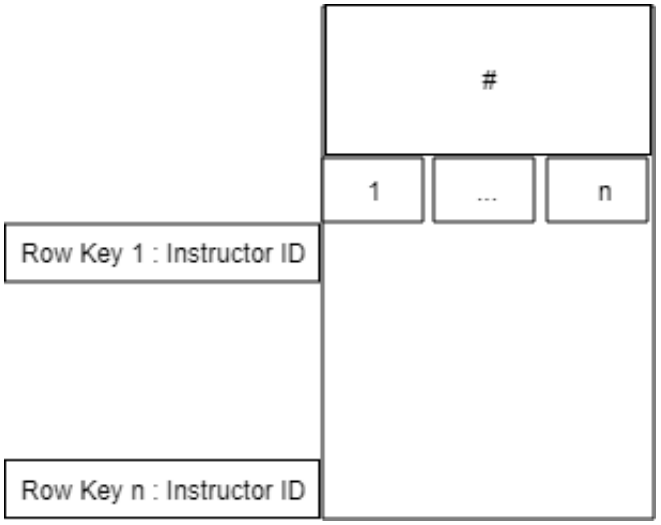

Figure 2.2.3-4: Instructor Table

2.2.4 The grade table Design

The Grade table G (for Grade) stores all student grades that is quite simple but will be filled by a huge increasing number of rows in considering the number of students and the number of courses per student.
This table contains only one column :G in the required column family for the grade of a course of a student. The row key of this table is consisted of different parts and is organized by year, semester, student and course, for instance 2015/S07/2012000123/S07A006.

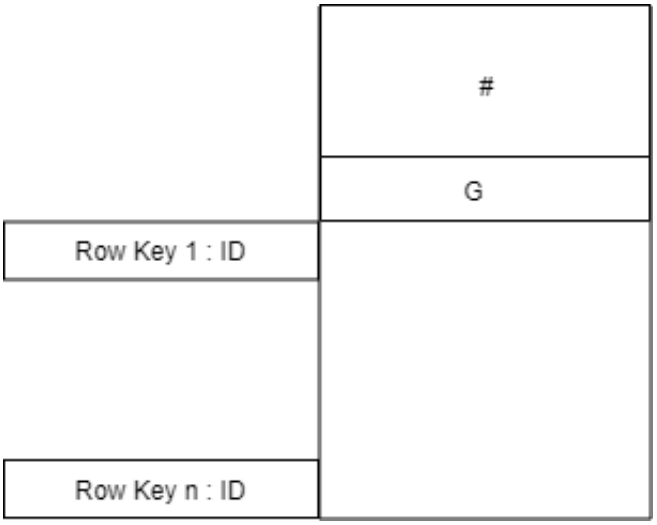

Figure 2.2.4-5: Grade Table

3 Workload

Our database workload is :

1. Return transcripts by school year.
2. Return the success rate of a given semester (from 1 to 10) according to the academic year.
3. Return the success rate of an EU since its establishment, in relation to its different names.
4. Return the success rate of an EU for a given school year.
5. Return the average marks of all the UEs of a promotion for a given school year.
6. Return the success rates for all UEs provided by a tutor .
7. Return the ranking of students in relation to their average scores according to the promotion and the school year.

4 Implementation

In this section , we show the implementation of mapreduce algorithms that answer every query in the workload and the rest api implementation.

4.1 Result Screenshots

In this section, we show screenshots for map reduce query after invoking the API.

4.1.1 Question1

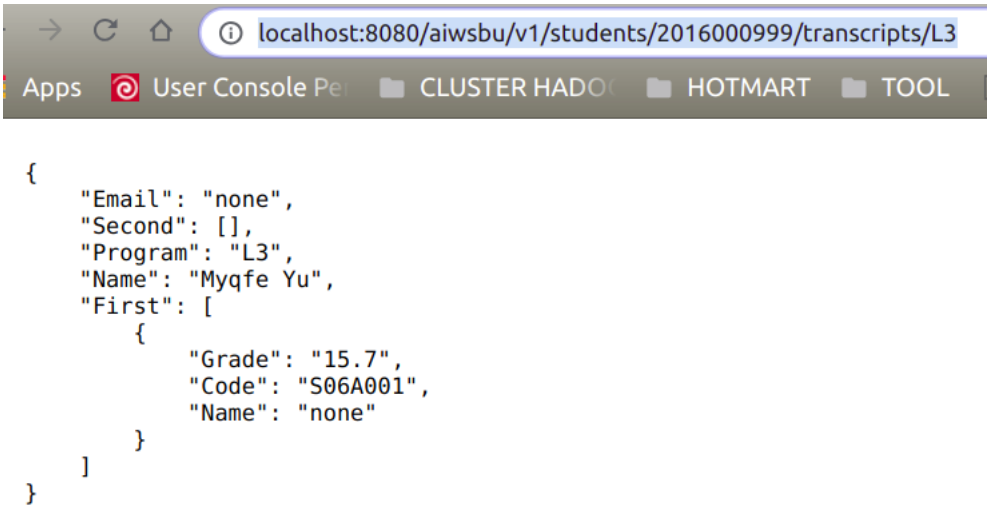

Figure 4.1.1-6: Screenshot for question1

4.1.2 Question2

Figure 4.1.2-7: Screenshot for question2

4.1.3 Question3

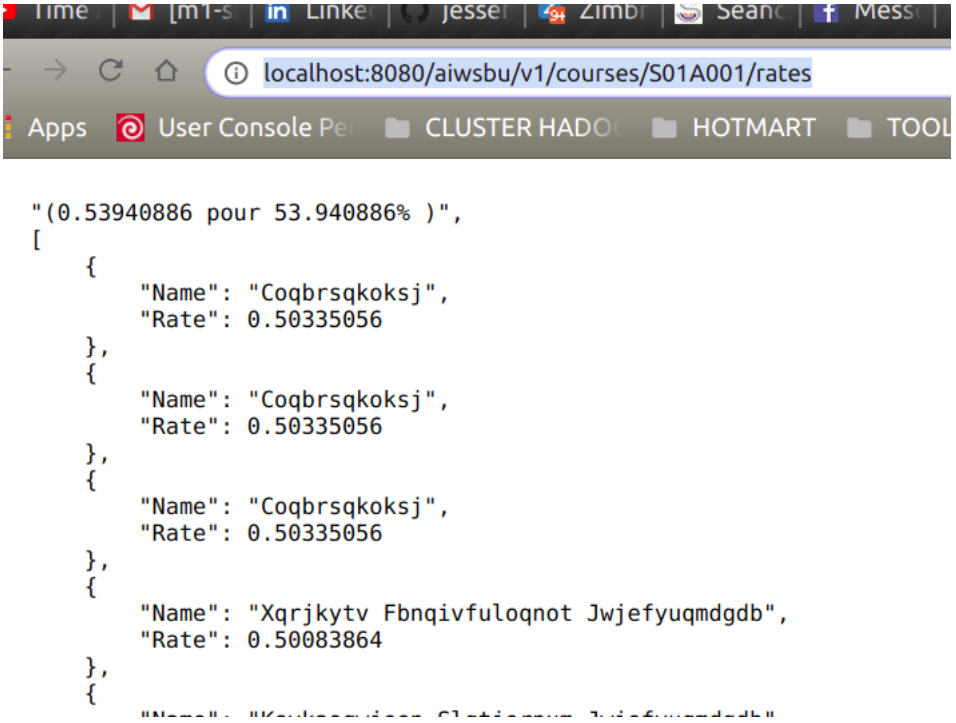

Figure 4.1.3-8: Screenshot for question3

4.1.4 Question4

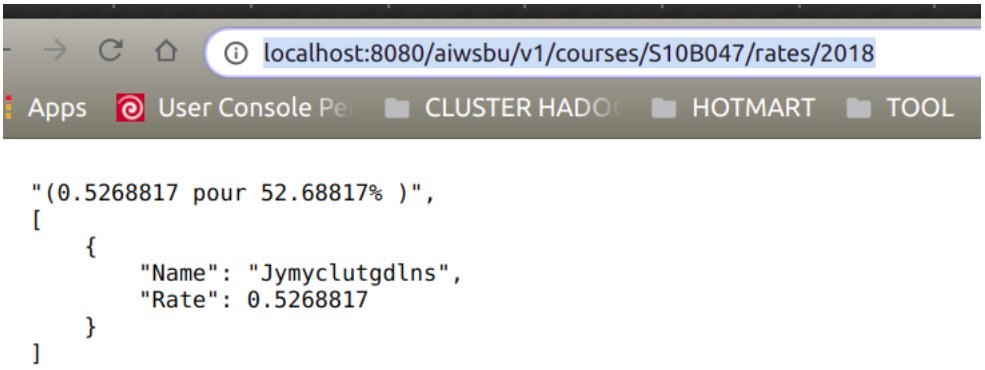

Figure 4.1.4-9: Screenshot for question4

4.1.5 Question5

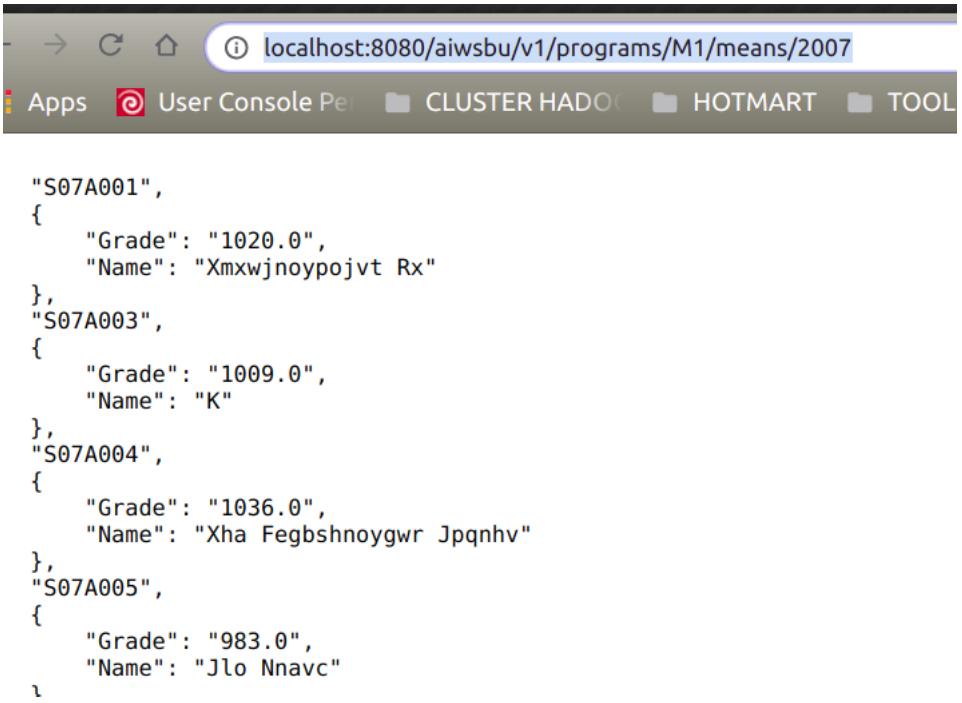

Figure 4.1.5-10: Screenshot for question5

4.1.6 Question6

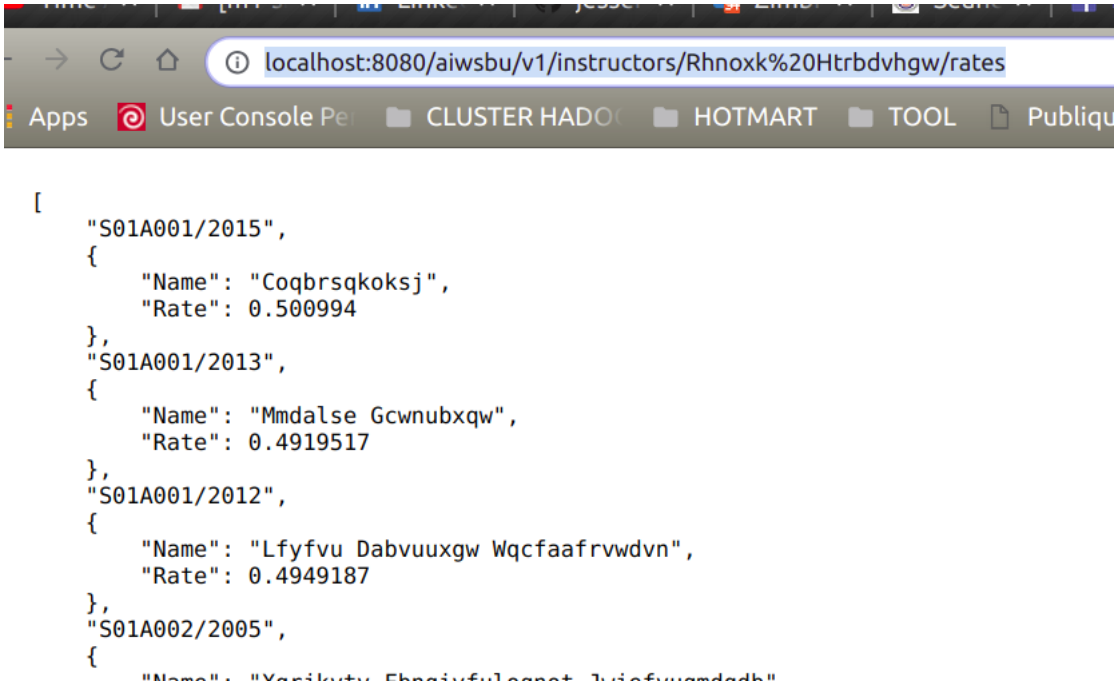

Figure 4.1.6-11: Screenshot for question6

4.1.7 Question7

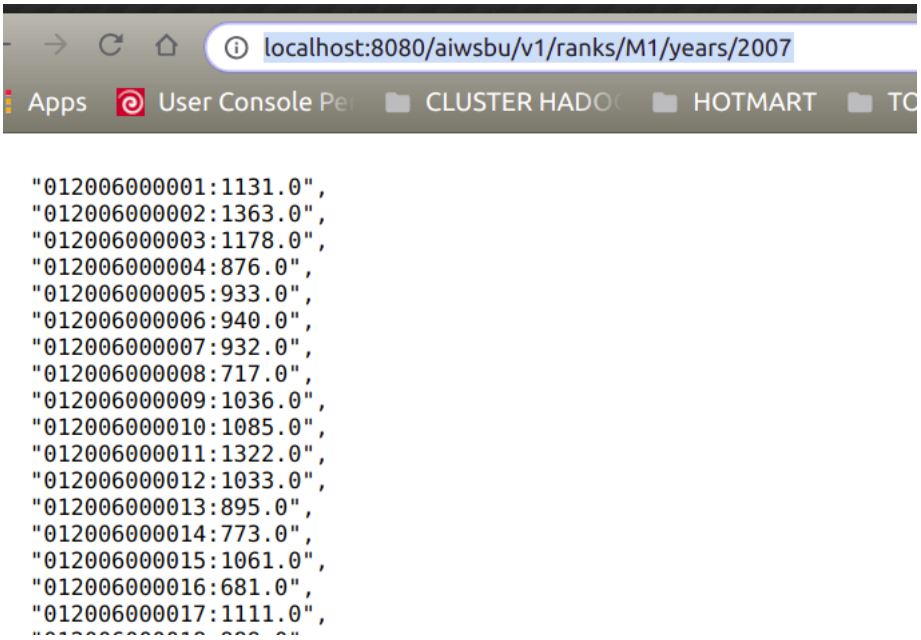

Figure 4.1.7-12: Screenshot for question7

4.2 Rest API

The rest API was implemented on a simple Python 2.7 WEB SERVER. Located into webserve_hb directory.

5 User guide

To execute this project on your own environment, make sure that you have successfully installed and setting the following technologies:

1. Java SDK 1.8;
2. Python 2.7
3. Hadoop version 2.8.4;
4. HBase version 1.2.6.1;
5. Eclipse+m2e;

6. Git;

Download this project from Celene or from our Github /jessefilho/bigdataproj

```
git clone https://github.com/jessefilho/bigdataproj
```

After installation of hadoop and hbase, execute commands lines below in a terminal linux for start the following services:

1. Hadoop services:

```
$ start-all.sh
```

2. Test if all hadoop services is running:

```
$ jps
```

3. HBase services:

```
$ start-hbase.sh
$ hbase-daemon.sh start thrift
```

And then to load HBase database:
$ java -cp ‘hbase classpath‘:bdproje_A.jar bdma.bigdata.aiwsbu.data.Setup

Then to running the jobs, it is recommended to use eclipse IDE, because we tried to use the proposed cluster, but the same show *java.lang.NoClassDefFoundError: org/apache/hadoop/hbase/HBaseConfiguration* when we execute with hadoop command such as *hadoop jar bdproje_A.jar bdma.bigdata.aiwsbu.mapreduce.Question1* at *dib-hadoop-master:21805893* directory.

We would like to remember that a cause of random data we used an interval in each job, that means that just data between this interval will be find on our REST API implemented with python.

Thus to execute our project on an eclipse IDE, first of all you need install m2e plugin into eclipse and maven on your environment to make sure that all dependencies is properly installed. Then at the same directory of our pom.xml execute *mvn install* and all mvn command needed to your environment.

After that you are able to running each question .java. It most be execute in the following order:

1. Question 1;

2. Question 2job1;
3. Question 2job2;
4. Question 3;
5. Question 4;
6. Question 5;
7. Question 6;
8. Question 7;

Since of you have all back-end services running and all jobs ended, the next step is to run our WEB SERVER locate on webserve.py file:

Run Python command WEB-SERVER:

```
[YOUR DIR PATH TO]/bigdata/webserve_hb: $ python webserve.py
```

If some message like *error: [Errno 32] Broken pipe* appears, please re-running the web-server, because it is a lean WS just for test propose, then it have a very short timeout count.

Our REST URI execute query from follow templates such as below:

Question 1:
http://localhost:8080/aiwsbu/v1/students/[ID]/transcripts/[program]
e.i:
http://localhost:8080/aiwsbu/v1/students/2016000999/transcripts/L3

Question 2:
http://localhost:8080/aiwsbu/v1/rates/[semester]
e.i:
http://localhost:8080/aiwsbu/v1/rates/S07

Question 3 API:
http://localhost:8080/aiwsbu/v1/courses/[id]/rates
e.i:
http://localhost:8080/aiwsbu/v1/courses/S01A001/rates

Question 4:
http://localhost:8080/aiwsbu/v1/courses/[id]/rates/[year]
e.i:
http://localhost:8080/aiwsbu/v1/courses/S10B047/rates/2018

Question 5:
http://localhost:8080/aiwsbu/v1/programs/[program]/means/[year]
e.i:
http://localhost:8080/aiwsbu/v1/programs/M1/means/2007

Question 6:
http://localhost:8080/aiwsbu/v1/instructors/[name]/rates
e.i:
http://localhost:8080/aiwsbu/v1/instructors/Rhnoxk%20Htrbdvhgw/rates

Question 7:
http://localhost:8080/aiwsbu/v1/ranks/[program]/years/[year]
e.i:
http://localhost:8080/aiwsbu/v1/ranks/M1/years/2007

6 Conclusion

This project was held during the first year master degree for the purpose of implementing a web server that access the scholar database of the university of Blois.

References