Goals:

This project aims to search for the best model to be used in order to investigate all the comments and the stories tables that found in Hacker News website. We have used Cassandra as one of the Big tools to create the database to store, clean, process and to retrieve the data. We also aiming to use the application programming interface (API) to facilitate loading the bulk data into Cassandra to visualize the result.

Objectives:

1. Download a dataset using the Big Query.
2. Develop NoSQL data base into Cassandra.
3. Implement the data model by using DDL and DML to perform a batch data loading into Cassandra.
4. Perform the read and write operations using specific queries.
5. Link the data base model using API and perform some queries.

Problem Statement:

This project focus on the challenge of how to load a bulk data into NoSQL tool, specifically using Cassandra, and to build a data model to handle, process and retrieve the data. Moreover, we need to investigate on how to link the Cassandra model with the API to perform the queries.

Data and Modelling

## Descriptive Analysis

The data was obtained from [www.kaggle.com](http://www.kaggle.com) website. It is unstructured data contains of all the comments and the stories of the Hacker News from 2006-2017.Each story consist of the story id, and the author who submitted it. The comments table consist of the id, the author name and the user who made the comment. It is obvious that there are lots of missing values, but no pre-processing or cleaning data has been done.

We are considering using Cassandra as NoSQL tool, to upload the data and to create all the queries.

The table below shows the description of the attributes:

|  |  |  |
| --- | --- | --- |
| Attributes | Description | Data Type |
| ID | Story ID | Unique id |
| By | Submitter username | Nominal |
| Score | Story Score | Numeric |
| Time | Unix Time | Numeric |
| Time stamp | Readable Time | Numeric |
| Title | Story Title | Nominal |
| URL | Story URL | Nominal |
| Text | Story text | Nominal |
| Parent | Parent ID | Numeric |
| Deleted | Is the post deleted? | Boolean |
| Dead | Is the post dead? | Boolean |
| Descendants | Number of story descendants | Numeric |
| Author | Author username | Nominal |

# Data Structure and Data Modelling:

1. Data Structure:

We have created two nodes, and connected them to each other, we have created three tables according to the main three tables in the website. The tables are stories, the comments and the full table.

CREATE TABLE project\_DB.stories ( id int PRIMARY KEY, by text, score int, time int, time\_st timestamp, title text, url text, text text, dead Boolean, deleted Boolean, descendants int, author text);  
   
CREATE TABLE project\_DB.comments ( id int PRIMARY KEY, by text, author text, time int, time\_ts timestamp, text text, parent int, dead boolean, deleted Boolean, ranking int);

CREATE TABLE project\_DB.full\_table ( id int PRIMARY KEY, by text, time int, timestamp timestamp, title text, type text, URL text, text text, parent int, deleted Boolean, dead Boolean, descendants int, ranking int );

Queries:

In Cassandra we have modelled five queries, for each we have created a different tables as below:

1. Search the stories with the word politics from 2015-2018:

|  |  |
| --- | --- |
| Column Name | Data Type |
| Year | Int |
| Story\_id | Int |
| Author\_username | Text |
| Source | Text |
| Title | text |

The PRIMARY KEY (year, story\_id)

1. Search the stories with the words, facebook, intel, Microsoft Show the average, maximum, minimum scores of each.

|  |  |
| --- | --- |
| Column Name | Data Type |
| Title | Text |
| Score | Int |
| year | Int |
| Author\_username | Text |
| Story\_id | Int |

PRIMARY KEY()

1. Search for the lowest score for the year between 2013 and 2018

|  |  |
| --- | --- |
| Column Name | Data Type |
| Score | Int |
| Title | Text |
| Author\_username | Text |
| Year | Int |
| Story\_id | int |

PRIMARY KEY()

1. Search for the author names with the count greater than 30

|  |  |
| --- | --- |
| Column Name | Data type |
| Author\_username | Text |
| Count | int |

1. Search for

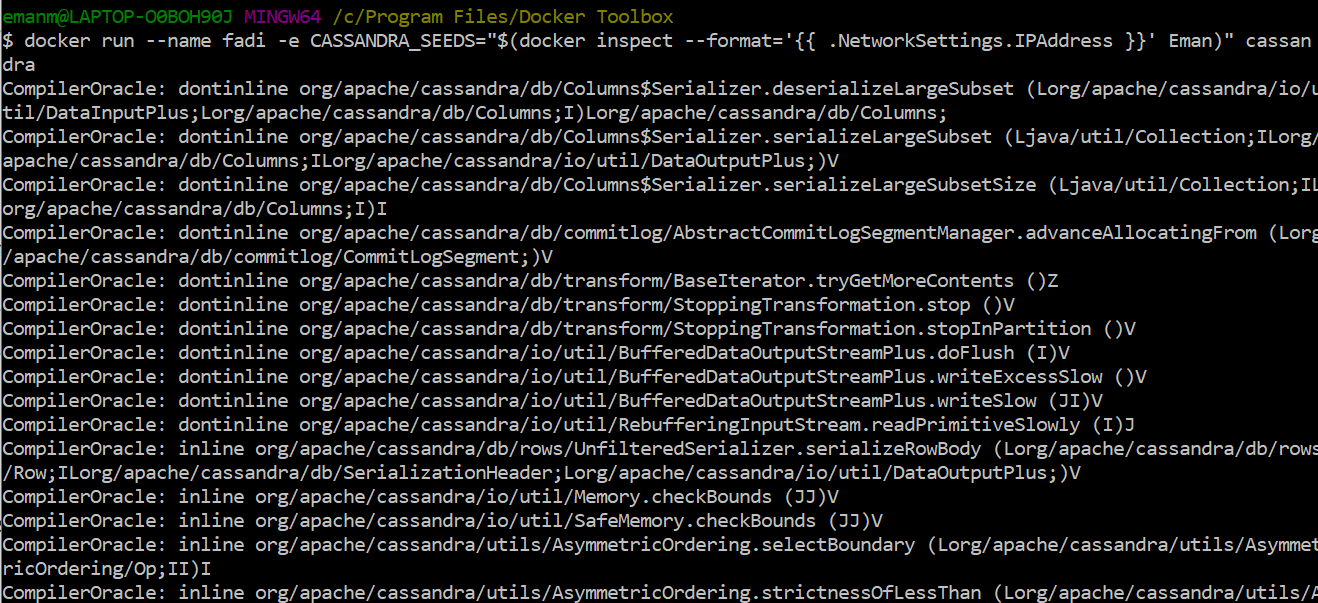
|  |  |
| --- | --- |
| Column Name | Data type |
| Title | text |
| Story\_id | Int |
| Authors\_username | Text |
| Year | Int |
| Quarter | Int |
| month | int |

PRIMARY KEY()

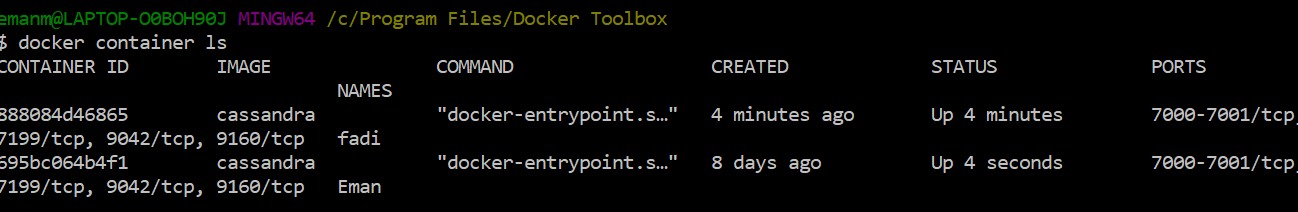
1.Creating the first cluster node (Eman)



2.Creating the second cluster node (Fadi), and connecting both of them as a master and slave to establish gossiping.



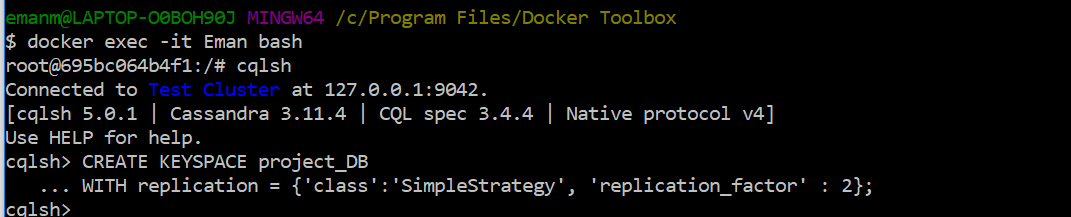
3.Showing the two nodes running



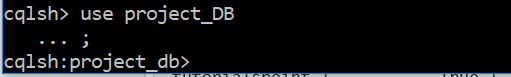
4.Creating a database for the hacker stories

CREATE KEYSPACE project\_DB

WITH replication = {'class':'SimpleStrategy', 'replication\_factor' : 2};



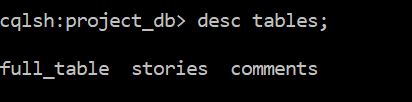
5.Using the project database



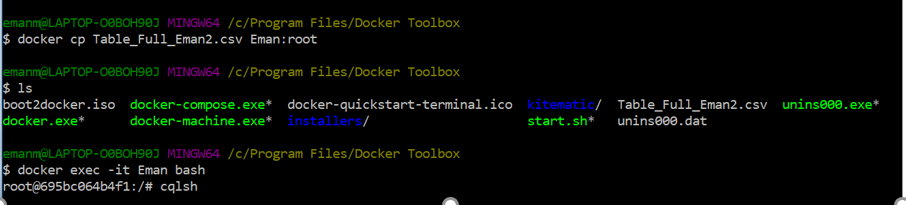
6.Creating the database tables to hold all needed data

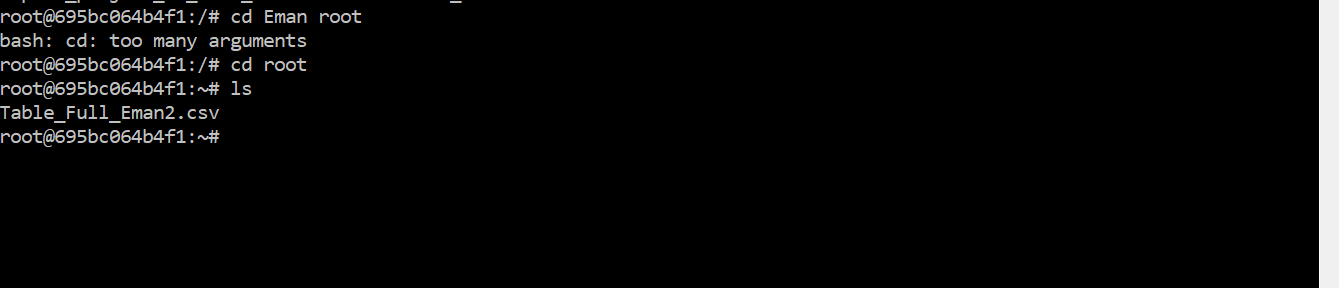
CREATE TABLE project\_DB.stories ( id int PRIMARY KEY, author text, by\_name text, comments text, dead Boolean, deleted Boolean, descendants int, score int, time int, time\_ts timestamp, title text, url text);  
   
CREATE TABLE project\_DB.comments ( id int PRIMARY KEY, author text, comment text, comment\_by text, dead boolean, deleted boolean, parent\_comment int, ranking smallint, time int, time\_ts timestamp);  
  
CREATE TABLE project\_DB.full\_table ( id int PRIMARY KEY, by\_name text, comments text, parent int, time int, timestamp timestamp, title text);

6.All tables created in Cassandra



7. we extracted a bulk of data using Big Query and we saves them as CSV file to load them in the created tables. Before loading we have coped the file into the docker root (container) so we could later load it in the tables as shown below





8. We loaded the data into the table (Full Table).

Data Modelling:

We have chosen 5 queries to run

Query1: Select all records that has a story title that contains "politics" by from 2015 to 2018. Show the source, author, story title and year of publication

Read: SELECT \* FROM politics\_by\_year WHERE year>2014 and year<2019 ALLOW FILTERING;

Problem: Cannot order in the right way

Outcome:A screenshot of a cell phone

Description automatically generated

Query2: Select all records that has a story title that contains "facebook", "intel", Microsoft". Show the average, maximum, minimum scores of each.

Read: SELECT title, year, min/max/avg(score) AS score, authors\_username, story\_id FROM name\_score WHERE name=1/2/3;

Problem: Cannot select the title by using command such as “like ‘%facebook%’” in sql.

Solution: add another column called ‘name’ and classified all the titles contain “Facebook” into number 1, “Intel” into number 2, “Microsoft” into number 3.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A picture containing indoor

Description automatically generated

Outcome:

Facebook

Max

A screenshot of a cell phone

Description automatically generated

MinA screenshot of a cell phone

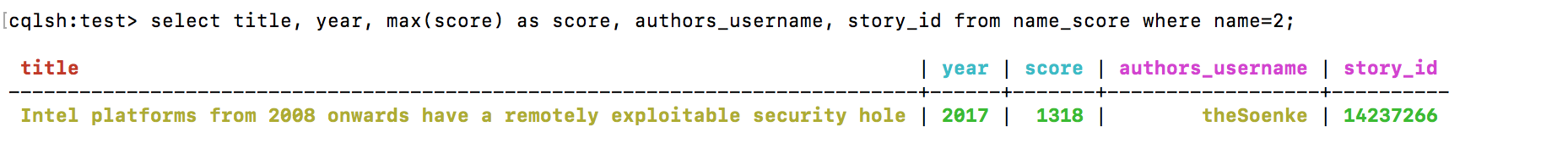
Description automatically generated

Avg

A screenshot of a cell phone

Description automatically generated

Intel

Max

MinA screenshot of a cell phone

Description automatically generated

Avg

A screenshot of a cell phone

Description automatically generated

Microsoft maxA screenshot of a cell phone

Description automatically generated

Min

A screenshot of a cell phone

Description automatically generated

avgA screenshot of a cell phone

Description automatically generated

Query3: Select the lowest scores of each year from 2013 to 2018 . Show the story title and author.

Read: SELECT MIN (Score) AS Score, Title, Author\_Username, Year, Story\_ID FROM lowest\_score\_by\_year WHERE Year=2013/2014/2015/2016/2017/2018 ALLOW FILTERING;

Problem: Data of 2009 and 2012 can not be loaded into Cassandra, so I changed the query into ‘Select the lowest scores of each year from 2013 to 2018’A close up of a piece of paper

Description automatically generated

Outcome:

A screenshot of a cell phone

Description automatically generated

Query4: Show all authors who created stories more than 30 times . Show the author.

Read: SELECT Authors\_username, Count FROM authors\_30 WHERE Count>30 ALLOW FILTERING;

Problem: Can not use Count() command to summarize the appearing time of each author. So I replaced the method with using Count() command in SQL then loading the data into Cassandra.

A screenshot of a cell phone

Description automatically generated

Outcome:

A screenshot of a social media post

Description automatically generated

Query5: Select the latest story created, show the date of publication, story\_id, title, and author

Read: SELECT Title, Story\_ID, Authors\_username, Year, Quarter, Month From latest\_story WHERE Year>2017 and Quarter>3 and Month>11 ALLOW FILTERING;

Outcome:A screenshot of a cell phone

Description automatically generated

Connecting Cassandra with Java API

Here is the Java code file:

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

This is the result after executing the java file and we got 12 errors which seems like the system can not import the datastax package.

A screenshot of text

Description automatically generated