lab 3 Column-oriented Databases

Goals

The objectives of this work are:

- Understand the fundamentals of databases where tables are stored by columns.
- Install and use an open source solution.
- Develop solutions for various use cases.

prior note

This module should preferably be developed in Linux. If you intend to use Windows, check the compatibility notes for the software you will be using.
Submit code/results/reports to elearning. Use a folder (1, 2, ..5) for each exercise, compressed into a single file.
Good job!

3.1 Cassandra – Command-line installation and exploitation

Cassandra is a *column-oriented database* initially developed by Facebook and currently supported by the “Apache Software Foundation”. It is an open source project, with “Apache License 2.0” license.

a) Install Apache Cassandra on your personal computer (http://cassandra.apache.org/) and run the server (cassandra -f).

b) Study the functioning of the system by testing the most used commands, through the command line (client program cqlsh).

Consult the slides provided for the course and websites with documentation about Cassandra. Some examples:

- *http://cassandra.apache.org/doc/latest/*
- *https://www.tutorialspoint.com/cassandra/* You should study concepts and features such as:

- Keyspace creation, description and use
- Creation and Description of Tables
- Write, Read, Edit, Delete (CRUD)
- Column Values - use of *nested tuples, collections, etc.*
- Time-to-live and Timestamp

c) Produce a report (CBD_L301_<NMEC>.TXT) with all iterations with thecqlsh. Comment on some of the operations.

3.2 Cassandra – Video Sharing System

It is intended to implement a database that supports a video sharing system using the CQL language (Cassandra Query Language).

a) Develop and implement the data model of the video sharing system, taking into account the specifics of the Cassandra database and respecting the following requirements:

1. User management: among others, register the username, name, email, registration timestamp on the platform;

2. Video management: among others, register the author of the share, video name, description, one or more tags and upload/share time stamp;

3. Management of comments: made for a video at a given moment in time and authored by a user;

4. Management of video followers: allow the registration of users who follow a certain video. In an information system, it will allow, for example, to notify all followers of a new comment inserted for the video;

5. Event recording: by video and user and can be of the play/pause/stop type, including the temporal recording of the event and the (temporal) moment in which it occurs in the video. For example, user XPTO played the YGZ video at 2:35:54 on October 2, 2017, 300 seconds into the video;

6. Video Rating: Integer value from 1-5, per video and does not require author registration.

7. Allow searching for all videos by a given author;

8. Allow the user to search for comments, sorted inversely by date;

9. Allow searching for comments by videos, sorted inversely by date;

10. Allow searching the average rating of a video and how many times it was voted;

b) Enter data in all created tables (10-30 insertions per entity). Their content must make sense, namely ensuring that we have data that allow us to respond to the various requested surveys. Then, and for each table, make a query that returns all records in JSON format. Must submit to script with CQL DML*statements*created, as well as the result of each query in a ".json" file.

c) If you have not already done so, implement the searches defined in paragraph *7*The *10*;

d) Implement the following searches.
Important notes : a) You should avoid using "allow filtering" in implemented queries; b) Some items/queries are not possible to perform in Cassandra; identify and justify them, based on the concepts of Cassandra's data model.

1. The last 3 comments entered for a video;

2. List of tags for a given video;

3. All videos tagged Aveiro;

4. The last 5 events of a given video made by a user;

5. Videos shared by a certain user (maria1987, for example) in a certain period of time (August 2017, for example);

6. The last 10 videos, sorted inversely by shared date;

7. All followers (followers) of a given video;

8. All comments (of videos) that a user is following (following);

9. The 5 highest rated videos;

10. A query that returns all the videos and that clearly shows the way in which they are ordered;

11. List with the existing Tags and the number of videos cataloged with each one of them;

12. .. 13. Describe 2 additional questions to the database (items 12 to 13), significantly different from the previous ones, and also present the research solution for each question.

3.3 Cassandra – Driver (Java)

For this exercise you should use the video sharing system's database, but now accessing it programmatically through a JAVA driver. To install the driver on your desktop (Java IDE), you must follow the recommendations available. For example in:

https://www.tutorialspoint.com/cassandra/cassandra_installation.htm https://docs.datastax.com/en/driver-matrix/doc/driver_matrix/javaDrivers.html https://github.com/datastax/java-driver

a) Develop a small Java application that demonstrates inserting, editing and searching records in the database.

b) Reimplement in Java, four *queries* your choice of exercise 3.2.

3.4 Free Themed Database

This exercise aims to develop a small database that takes advantage of Cassandra's data model and whose theme is free. However, you must meet the following requirements:

a) A keyspace with at least 4 tables;

b) Insertion of an average of 12 records per table;

c) Use of the following data structures (all):

- *set, list, map*

d) Definition of at least 2 secondary indices;

e) Use of at least 5 updates and 5 data deletes:

- *use non-trivial operations on data structures (from point c))*

f) Creation of 10 expressive queries of your domain of knowledge of the clause

SELECT:

- *use WHERE, ORDER BY, LIMIT, etc.*

3.5 Cassandra – Restaurant Database (Optional)

Refer to the Lab2 guide restaurant database.

a) Build a replica of that database using Cassandra now. The data model must take into account the specifics of this type of database.

b) Load the database with the contents of the file “restaurantes.json”.

c) Implement queries 1-5, 7, 8, 12, 14, 15, 17, 19, 27, and 30 from exercise 2.2, using the CQL Data Management Language (DML) instruction set. The query result must be presented/saved in JSON format.
Note that some of these queries may require additional client-side processing.