COMP-1835-M01-2022-23 Graph and Modern Databases

Coursework

001255924

Hadis Babakhani Roudbardeh

Part 1

Peer report-paper number 10

Summery

The performance of Big Data databases, storage systems, and NoSQL databases are all topics covered in this article. Many NoSQL database types and subcategories are taken into account, along with the benefits and drawbacks of each category. This study aims to discuss and contrast several NoSQL (Not Only Structured Query Language) solutions (using a variety of criteria). for high availability and excellent performance (both in terms of speed and size). Also, a functional comparison of the Cassandra, MongoDB, and Neo4j databases, which compares column-oriented databases, document data stores, and graph data stores. NoSQL was created to address the shortcomings of logical databases; at the time, it was necessary to handle non-structured data and do it more quickly.

Three NoSQL databases are compared in the article: Neo4J, MongoDB, and Apache Cassandra. Performance and other properties, including data storage, data modelling, query language, scalability, and availability, are compared.

Column-oriented databases like Cassandra, created and deployed at Facebook to maximise availability, scalability, and performance MongoDB is a document data store that prioritises scalability and ease of development but is not robust. Neo4J is a graph database that safeguards data integrity while providing quick read and write performance.

Each database has a unique query language, programming language, and data type. Neo4J utilises Cypher, Cassandra uses CQL, while MongoDB uses JavaScript. Cassandra does not enable data aggregation, although MongoDB does. Cassandra and MongoDB do not support full ACID operations, however Neo4J does.

Cassandra has the ability to scale either horizontally and vertically. Neo4J can scale out and up to handle big graphs, and MongoDB can scale out horizontally to hold large data volumes. Cassandra is better for real-time data analytics and reports whereas Neo4J is better for real-

time applications. Mobile apps and real-time personalisation both leverage MongoDB. Cross-reference use cases are best suited for graph databases, while IoT and user profile data are best stored in column-oriented databases, and MongoDB is a general-purpose distributed document data store.

The writing is easy and straightforward to read. The information is organised by the author. The work, however, lacks sufficient clarity and detail in a few places. The technical components should be expanded and clarified in order to guarantee that readers truly understand what the researchers looked into. For instance, the inventor of column-oriented databases talked about that in the first section, but in the second, I noticed that he seemed to be rejecting the first idea. There were a few mistakes, but not many. The author used a number of websites as sources, but a research paper is designed to rely on a wide range of sources, including books, journals, documents, and so on. and the absence of visuals was an additional issue.

Major issues:

The beginning of the essay is not well-developed, which makes it challenging for the reader to understand what the essay is about. This is one of the essay's main difficulties. The introduction should include a succinct summary of the topic, a list of the key ideas that will be covered, and a thesis statement that sums up the author's opinion. Without a strong introduction, the reader could find it difficult to comprehend the author's ideas.

The essay's confusing structure presents a further serious issue. Each paragraph of the essay should build upon the one before it in a logical and coherent way. This will make it easier for the reader to follow the author's reasoning and grasp the major points being raised. To ensure that the essay runs smoothly, the author should construct a clear framework and employ topic sentences and transitional phrases.

NoSQL databases are only skimmed throughout the text, which could lead readers who aren't familiar with the subject mislead. NoSQL databases and its key characteristics need to be explained in more detail by the author. For instance, the author claimed that NoSQL databases do not comply with ACID, but they should also mention that they frequently follow the BASE principles (essentially available, soft state, eventual consistency).

The absence of reliable sources to support the assumptions made in the article is another problem with it. The author should provide reliable evidence to support their views, such as scholarly publications, papers, and websites. It can be challenging for the reader to believe the information presented in the essay if unreliable sources are used.

The author needs to be more specific about the situations and uses for document-oriented databases in the section on those databases. The author mentioned that document-oriented databases are excellent for applications where data must be saved as a document, however they ought to also give examples of such applications. The author should also describe the special features of document-oriented databases and the conditions in which these characteristics make them preferable to other types of databases.

It would also be helpful to enhance the comparison of NoSQL databases. In order for the audience to understand the differences between the databases, the author should compare them based on particular features or categories. According to their data models, scalability, performance, and user-friendliness, for instance, the author could compare the databases. To further aid the reader in comprehending the differences between the databases, the author could potentially consider adding visual aids as tables or figures.

In conclusion, the essay can be made better by giving a clear introduction and thesis statement, creating a logical and coherent structure, using reliable sources, explaining NoSQL and document-oriented databases in considerable detail, making comparisons the databases based on particular features, and using visual aids to develop a good understanding.

Minor issues:

It is vital to pay attention to each database's unique aspects when comparing them, but it is also beneficial to note any similarities. In this instance, the author may have stated how horizontally scalable and schema-free MongoDB, Cassandra, Redis, and Neo4j are. The similarities and contrasts between these databases could then be made clearer by readers.

Databases have become an important topic recently, and NewSQL is an exciting subject connected to them. NewSQL is mentioned by the author as a future prediction, although more information and explanation would have been beneficial. Modern relational database management systems that try to tackle some of the scalability and performance concerns of traditional relational databases are referred to as "NewSQL" systems. Readers might have a

better understanding of the possible benefits of NewSQL if this subject was covered in more detail.

The methodology, findings, and study's conclusion should all be clearly summarised in the abstract. By doing this, readers could decide whether the article is appropriate to their interests and whether they should keep reading. As a result, the author could enhance the abstract by giving more in-depth explanations of these topics.

While the article does include some references, more might be added to increase the information's credibility and dependability. This would also provide readers the option to learn more about the subject if they are eager to do so.

The article's advantages and downsides section could use more clarification and examples. This would enable readers to comprehend the problems associated with adopting NoSQL databases in greater detail. Readers may be able to decide whether to use NoSQL databases for their projects by being given particular instances of when they are advantageous and when they are not.

Finally, using bullet points and subheadings to organise the content could make the text easier to read and more rationally structured. There are a number of NoSQL database properties to take into account, thus this is very pertinent. Longer sentences could be broken up into shorter ones to help the reader understand the information.

Presentation and style Comments:

To help the reader understand the text, the document may use a more organised layout with headings and subheadings. Readers can more clearly follow the author's thought process and understand how the content is organised by breaking the essay up into sections. Also, by emphasising and highlighting important ideas, the author can help the reader better understand the essay's main thesis.

To enhance the essay's presentation, the author can use short, clear paragraphs that are simple to read, graphs and charts to help the reader visualise, and lists with bullet points and numbers. Using shorter sentences and reducing longer ones into shorter ones helps the reader understand complex ideas easily.

references Comments: It's crucial to cite enough trustworthy information to back up the assumptions and arguments you make in your essay. It is recommended that using review scholarly publications, journals, or articles from reliable sources to guarantee the accuracy of the data presented.

Besides,The author must also ensure that every reference is correctly formatted and credited. This includes, among other things, the author's name, the work's title, and the publication date. A correctly formatted reference list can provide the work credibility and show readers that the author is an authority on the subject after much research and study.

Also, it's important to keep in mind that the references should be diverse and represent a variety of points of view on the subject. This can be used by the author to demonstrate that they were open to other ideas and viewpoints before making their own judgements. A range of references can also help the author provide the reader a more in-depth understanding of the subject.

Part 2

Virtual machines numbers for both graphdb and nosql-31

Redis:

```
[nosql@nosql Desktop]$ redis-cli
127.0.0.1:6379> select 0
OK
```

First, I created a database for a high school and named it 0.since we just can use numbers for the name of the database in Redis.

```
127.0.0.1:6379> set high_school_name "narjes"
OK
127.0.0.1:6379> set students_number 2000
OK
127.0.0.1:6379> set teachers_number 80
OK
127.0.0.1:6379> set location "{\"country\": \"Iran\", \"city\": \"Tehran\"}"
OK
127.0.0.1:6379> set founded_year 1997
OK
127.0.0.1:6379> set tuition_fee 500
OK
127.0.0.1:6379> set is_private_school True
OK
127.0.0.1:6379> sadd sports_teams "basketball" "football" "volleyball"
(integer) 3
127.0.0.1:6379> hmset principal_information first_name "hadis" last_name "babakhani" email "hadisbabakhani@yahoo.com"
OK
127.0.0.1:6379> sadd classes "math" "english" "persian" "history"
(integer) 4
```

I have used ‘set’ command to set a value into a key.here, I set values into high_school_name,student_number,teachers_number,location,founded_year,tuition_fee and is_private_school.

Sadd, is used for adding one or more members to a set.I have two sets here named, sports_teams and classes.

Hmset is a command to set multiple field-value into a key.here I used that just for principal_information.then I used get and hmget commands to retrieve values.

The outputs are below:

```
127.0.0.1:6379> get high_school_name
"narjes"
127.0.0.1:6379> get students_number
"2000"
127.0.0.1:6379> get teachers_number
"80"
127.0.0.1:6379> get location
"{\"country\": \"Iran\", \"city\": \"Tehran\"}"
127.0.0.1:6379> get tuition_fee
"500"
127.0.0.1:6379> smembers sports_teams
1) "football"
2) "basketball"
3) "volleyball"
127.0.0.1:6379> get is_private_school
"True"
```

And I implemented some queries here as well:

```
127.0.0.1:6379> hget principal_information
(error) ERR wrong number of arguments for 'hget' command
127.0.0.1:6379> hget principal_information first_name last_name email
(error) ERR wrong number of arguments for 'hget' command
127.0.0.1:6379> hmget principal_information first_name last_name email
1) "hadis"
2) "babakhani"
3) "hadisbabakhani@yahoo.com"
127.0.0.1:6379> incr students_number 10
(error) ERR wrong number of arguments for 'incr' command
127.0.0.1:6379> lpush students_list "hoodad farhad"
(integer) 1
127.0.0.1:6379> incr students_number
(integer) 2001
127.0.0.1:6379> sadd sport_teams "padel"
(integer) 1
127.0.0.1:6379> del sport_teams
(integer) 1
127.0.0.1:6379> sadd sports_teams "padel"
(integer) 1
127.0.0.1:6379> smembers sports_teams
1) "padel"
2) "football"
3) "basketball"
4) "volleyball"
127.0.0.1:6379> hset principal_information email "babakhanihadis24@gmail.com"
(integer) 0
127.0.0.1:6379> hget principal_information email
"babakhanihadis24@gmail.com"
```

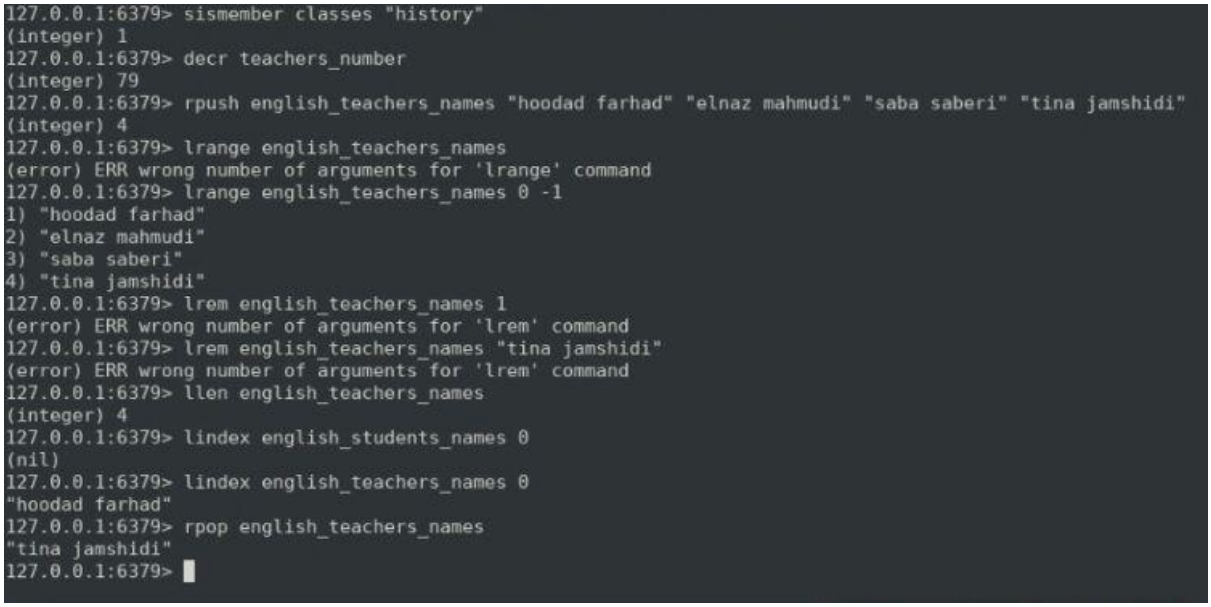
1-I created a student_list and added hoodad farhad as an example with lpush command.

2-I increased the number of students by incr command:2000 turned to 2001.

3-I added padel to sports_teams set.

4-with smembers command I get and show all values including the new one I added.

5,6-I set another email for principal_information key and used hget to retrieve that in the next line.



7-sismember is a command used to check if a given value is a member of a set or not.if the output is 1 it means that the value is a member of the set and here the output is 1 which means history is a member of our classes key.

8-decr is a command to decrease the number and here I used that to decrease teachers_number from 80 to 79.

9,10-I created a list of English_teachers_names with rpush command and retrieved them in the next line with lrange command.0 -1 is necessary to show that you wanted the whole list.

11-with llen I got the length of the list which is 4.

12-lindex command shows the value of the choosed index.here I wanted the value with index 0 which is hoodad farhad

13-with rpop command I removed tina jamshidi from the list.

Benefits and drawbacks:redis is a fast database because it stores data in memory instead of disk.it’s flexible and working with redis was easy.for drawbacks I can mention storage limitation and also as we see it’s not suitable for complex queries.besides, it is single threaded and it can handle just one request at a specific time.

Cassandra:

For this part, I created a database called airline and also a table called airline as well with 30 rows, with columns named:flight_numb,capacity,company,destination,flight_duration,origin and type_craft

```
cqlsh:airline> SELECT * FROM airline.airline;

 flight_numb | capacity | company          | destination | flight_duration | origin     | type_aircraft
-------------+----------+------------------+-------------+-----------------+------------+---------------
       QJ872 |      150 | british airways  |      London |               1 |      Paris |        airbus
       OL555 |      100 | turkish airline  |    Istanbul |               6 | manchester |        airbus
       LB260 |      200 |   qatar airways  |      Berlin |              10 |    Toronto |        boeing
       MN711 |      300 |   qatar airways  |      Madrid |              10 |    Toronto |        boeing
       CX812 |      160 |    fly emirates  |       Dubai |               2 |   Istanbul |        airbus
       QC009 |      150 |    fly emirates  |    Istanbul |               5 |      Paris |        airbus
       ZW002 |      350 |    fly emirates  |     Toronto |               9 |   Istanbul |        boeing
       OL055 |      100 | turkish airline  |      Tehran |               8 | manchester |        airbus
       ZH092 |      350 | british airways  |     Toronto |               7 |     London |        boeing
       KW805 |      180 | british airways  |      London |               1 | manchester |        airbus
       KW809 |      180 | british airways  |      London |               1 | manchester |        airbus
       LW000 |      170 |       lufthansa  |   Amsterdam |               2 |      Milan |        boeing
       HF160 |       60 |   qatar airways  |       Paris |               2 |      Milan |           jet
       PW722 |      300 | turkish airline  |      Lisbon |               3 |      Paris |        boeing
       PA722 |       70 | turkish airline  |      Tehran |               2 |   Istanbul |           jet
       LQ966 |      300 | turkish airline  |    Istanbul |               5 |     Tehran |        boeing
       LQ987 |      150 |    fly emirates  |    Istanbul |               5 |     Berlin |        airbus
       LW060 |      170 |       lufthansa  |   Amsterdam |               2 |     Berlin |        boeing
       DF160 |      250 |   qatar airways  |        Rome |               7 |     Tehran |        boeing
       ZW092 |      350 |    fly emirates  |     Toronto |               7 |     London |        boeing
       AG571 |      300 |   qatar airways  |      London |              10 |    Toronto |        boeing
       LK871 |      300 |        iran air  |      Tehran |               7 |     London |        boeing
       LE760 |      350 |   qatar airways  |      Tehran |               6 |      Paris |        boeing
       ZH792 |      350 | british airways  |      London |               7 |    Toronto |        boeing
        W501 |      200 |       Mahan Air  |      Tehran |               7 |     London |        airbus
       RQ009 |       80 | turkish airline  |    Istanbul |               5 |     London |           jet
       DF100 |      150 | british airways  |      London |               7 |     Tehran |        airbus
       LB060 |      200 |       lufthansa  |      Berlin |              10 |    Toronto |        boeing
       LM999 |      160 |    fly emirates  |       Dubai |               2 |     Tehran |        airbus
       LE260 |      200 |   qatar airways  |      London |               9 |    Toronto |        boeing
```

1-here with select * from airline.airline I got all values that I have.

```
(30 rows)
cqlsh:airline> SELECT * FROM airline WHERE origin='Tehran';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"
cqlsh:airline> SELECT * FROM airline WHERE origin='Tehran' ALLOW FILTERING;

 flight_numb | capacity | company         | destination | flight_duration | origin | type_aircraft
-------------+----------+-----------------+-------------+-----------------+--------+---------------
       LQ966 |      300 | turkish airline |    Istanbul |               5 | Tehran |        boeing
       DF160 |      250 |   qatar airways |        Rome |               7 | Tehran |        boeing
       DF100 |      150 | british airways |      London |               7 | Tehran |        airbus
       LM999 |      160 |    fly emirates |       Dubai |               2 | Tehran |        airbus

(4 rows)
cqlsh:airline> SELECT * FROM airline WHERE destination='Toronto' ALLOW FILTERING;

 flight_numb | capacity | company         | destination | flight_duration | origin   | type_aircraft
-------------+----------+-----------------+-------------+-----------------+----------+---------------
       ZW002 |      350 |    fly emirates |     Toronto |               9 | Istanbul |        boeing
       ZH092 |      350 | british airways |     Toronto |               7 |   London |        boeing
       ZW092 |      350 |    fly emirates |     Toronto |               7 |   London |        boeing

(3 rows)
cqlsh:airline> SELECT * FROM airline WHERE flight_duration>8 ALLOW FILTERING;

 flight_numb | capacity | company       | destination | flight_duration | origin   | type_aircraft
-------------+----------+---------------+-------------+-----------------+----------+---------------
       LB260 |      200 | qatar airways |      Berlin |              10 |  Toronto |        boeing
       MN711 |      300 | qatar airways |      Madrid |              10 |  Toronto |        boeing
       ZW002 |      350 |  fly emirates |     Toronto |               9 | Istanbul |        boeing
       AG571 |      300 | qatar airways |      London |              10 |  Toronto |        boeing
       LB060 |      200 |     lufthansa |      Berlin |              10 |  Toronto |        boeing
       LE260 |      200 | qatar airways |      London |               9 |  Toronto |        boeing

(6 rows)
cqlsh:airline> SELECT COUNT(*) FROM airline WHERE destination='London' ALLOW FILTERING;

 count
-------
     7
```

2,3,4,5,6-I wanted to get the flight with origin=Tehran but I got the ALLOW FILTERING error as it was non-indexed column which has been solved with this query next line:

Select * from airline where origin=’tehran’ ALLOW FILTERING

And in other queries I did this with destination=’toronto’ and flight_duration=8

```
(29 rows)
cqlsh:airline> CREATE INDEX on airline(origin);
cqlsh:airline> SELECT * FROM airline.airline;

 flight_numb | capacity | company         | destination | flight_duration | origin     | type_aircraft
-------------+----------+-----------------+-------------+-----------------+------------+---------------
       QJ872 |      150 | british airways |      London |               1 |      Paris |        airbus
       OL555 |      100 | turkish airline |    Istanbul |               6 | manchester |        airbus
       LB260 |      200 |   qatar airways |      Berlin |              10 |    Toronto |        boeing
       MN711 |      300 |   qatar airways |      Madrid |              10 |    Toronto |        boeing
       CX812 |      160 |    fly emirates |       Dubai |               2 |   Istanbul |        airbus
       QC009 |      150 |    fly emirates |    Istanbul |               5 |      Paris |        airbus
       ZW002 |      350 |    fly emirates |     Toronto |               9 |   Istanbul |        boeing
       OL055 |      100 | turkish airline |      Tehran |               8 | manchester |        airbus
       ZH092 |      350 | british airways |     Toronto |               7 |     London |        boeing
       KW805 |      180 | british airways |      London |               1 | manchester |        airbus
       KW809 |      180 | british airways |      London |               1 | manchester |        airbus
       LW000 |      170 |       lufthansa |   Amsterdam |               2 |      Milan |        boeing
       HF160 |       60 |   qatar airways |       Paris |               2 |      Milan |           jet
       PW722 |      300 | turkish airline |      Lisbon |               3 |      Paris |        boeing
       PA722 |       70 | turkish airline |      Tehran |               2 |   Istanbul |           jet
       LQ966 |      300 | turkish airline |    Istanbul |               5 |     Tehran |        boeing
       LO987 |      150 |    fly emirates |    Istanbul |               5 |     Berlin |        airbus
       LW060 |      170 |       lufthansa |   Amsterdam |               2 |     Berlin |        boeing
       DF160 |      250 |   qatar airways |        Rome |               7 |     Tehran |        boeing
       ZW092 |      350 |    fly emirates |     Toronto |               7 |     London |        boeing
       AG571 |      300 |   qatar airways |      London |              10 |    Toronto |        boeing
       LK871 |      300 |      iran air |      Tehran |               7 |     London |        boeing
       LE760 |      350 |   qatar airways |      Tehran |               6 |      Paris |        boeing
       ZH792 |      350 | british airways |      London |               7 |    Toronto |        boeing
        W501 |      200 |      Mahan Air |      Tehran |               7 |     London |        airbus
       RQ009 |       80 | turkish airline |    Istanbul |               5 |     London |           jet
       LB060 |      200 |       lufthansa |      Berlin |              10 |    Toronto |        boeing
       LM999 |      160 |    fly emirates |       Dubai |               2 |     Tehran |        airbus
       LE260 |      200 |   qatar airways |      London |               9 |    Toronto |        boeing
```

But here I created index for origin column so I didn’t get this error from next time that I used that for origin=’berlin’

7,8-

```
(29 rows)
cqlsh:airline> SELECT COUNT(*) FROM airline WHERE origin='Berlin';

 count
-------
     2

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:airline> SELECT * FROM airline WHERE origin='Berlin';

 flight_numb | capacity | company      | destination | flight_duration | origin | type_aircraft
-------------+----------+--------------+-------------+-----------------+--------+---------------
       LO987 |      150 | fly emirates |    Istanbul |               5 | Berlin |        airbus
       LW060 |      170 |    lufthansa |   Amsterdam |               2 | Berlin |        boeing

(2 rows)
cqlsh:airline>
```

I used count command to count the number of flight with origin=berlin which is just 2 and with next line query I used select * from airline where origin=berlin I showed the flight with origin berlin.

```
cqlsh:airline> ALTER TABLE airline ADD delay INT;
cqlsh:airline> SELECT * FROM airline.airline;

 flight_numb | capacity | company          | delay | destination | flight_duration | origin     | type_aircraft
-------------+----------+------------------+-------+-------------+-----------------+------------+--------------
       QJ872 |      150 | british airways  |  null |      London |               1 |      Paris |       airbus
       OL555 |      100 | turkish airline  |  null |    Istanbul |               6 | manchester |       airbus
       LB260 |      200 |   qatar airways  |  null |      Berlin |              10 |    Toronto |       boeing
       MN711 |      300 |   qatar airways  |  null |      Madrid |              10 |    Toronto |       boeing
       CX812 |      160 |    fly emirates  |  null |       Dubai |               2 |   Istanbul |       airbus
       QC009 |      150 |    fly emirates  |  null |    Istanbul |               5 |      Paris |       airbus
       ZW002 |      350 |    fly emirates  |  null |     Toronto |               9 |   Istanbul |       boeing
       OL055 |      100 | turkish airline  |  null |      Tehran |               8 | manchester |       airbus
       ZH092 |      350 | british airways  |  null |     Toronto |               7 |     London |       boeing
       KW805 |      180 | british airways  |  null |      London |               1 | manchester |       airbus
       KW809 |      180 | british airways  |  null |      London |               1 | manchester |       airbus
       LW000 |      170 |       lufthansa  |  null |   Amsterdam |               2 |      Milan |       boeing
       HF160 |       60 |   qatar airways  |  null |       Paris |               2 |      Milan |          jet
       PW722 |      300 | turkish airline  |  null |      Lisbon |               3 |      Paris |       boeing
       PA722 |       70 | turkish airline  |  null |      Tehran |               2 |   Istanbul |          jet
       LQ966 |      300 | turkish airline  |  null |    Istanbul |               5 |     Tehran |       boeing
       LO987 |      150 |    fly emirates  |  null |    Istanbul |               5 |     Berlin |       airbus
       LW060 |      170 |       lufthansa  |  null |   Amsterdam |               2 |     Berlin |       boeing
       DF160 |      250 |   qatar airways  |  null |        Rome |               7 |     Tehran |       boeing
       ZW092 |      350 |    fly emirates  |  null |     Toronto |               7 |     London |       boeing
```

```
cqlsh:airline> UPDATE airline SET delay=4 WHERE flight_numb='HF160';
cqlsh:airline> SELECT * FROM airline.airline;

 flight_numb | capacity | company          | delay | destination | flight_duration | origin     | type_aircraft
-------------+----------+------------------+-------+-------------+-----------------+------------+--------------
       QJ872 |      150 | british airways  |  null |      London |               1 |      Paris |       airbus
       OL555 |      100 | turkish airline  |  null |    Istanbul |               6 | manchester |       airbus
       LB260 |      200 |   qatar airways  |  null |      Berlin |              10 |    Toronto |       boeing
       MN711 |      300 |   qatar airways  |  null |      Madrid |              10 |    Toronto |       boeing
       CX812 |      160 |    fly emirates  |  null |       Dubai |               2 |   Istanbul |       airbus
       QC009 |      150 |    fly emirates  |  null |    Istanbul |               5 |      Paris |       airbus
       ZW002 |      350 |    fly emirates  |  null |     Toronto |               9 |   Istanbul |       boeing
       OL055 |      100 | turkish airline  |  null |      Tehran |               8 | manchester |       airbus
       ZH092 |      350 | british airways  |  null |     Toronto |               7 |     London |       boeing
       KW805 |      180 | british airways  |  null |      London |               1 | manchester |       airbus
       KW809 |      180 | british airways  |  null |      London |               1 | manchester |       airbus
       LW000 |      170 |       lufthansa  |  null |   Amsterdam |               2 |      Milan |       boeing
       HF160 |       60 |   qatar airways  |     4 |       Paris |               2 |      Milan |          jet
       PW722 |      300 | turkish airline  |  null |      Lisbon |               3 |      Paris |       boeing
       PA722 |       70 | turkish airline  |  null |      Tehran |               2 |   Istanbul |          jet
       LQ966 |      300 | turkish airline  |  null |    Istanbul |               5 |     Tehran |       boeing
       LO987 |      150 |    fly emirates  |  null |    Istanbul |               5 |     Berlin |       airbus
       LW060 |      170 |       lufthansa  |  null |   Amsterdam |               2 |     Berlin |       boeing
       DF160 |      250 |   qatar airways  |  null |        Rome |               7 |     Tehran |       boeing
       ZW092 |      350 |    fly emirates  |  null |     Toronto |               7 |     London |       boeing
       AG571 |      300 |   qatar airways  |  null |      London |              10 |    Toronto |       boeing
       LK871 |      300 |        iran air  |  null |      Tehran |               7 |     London |       boeing
       LE760 |      350 |   qatar airways  |  null |      Tehran |               6 |      Paris |       boeing
       ZH792 |      350 | british airways  |  null |      London |               7 |    Toronto |       boeing
        W501 |      200 |       Mahan Air  |  null |      Tehran |               7 |     London |       airbus
       RQ009 |       80 | turkish airline  |  null |    Istanbul |               5 |     London |          jet
       DF100 |      150 | british airways  |  null |      London |               7 |     Tehran |       airbus
       LB060 |      200 |       lufthansa  |  null |      Berlin |              10 |    Toronto |       boeing
       LM999 |      160 |    fly emirates  |  null |       Dubai |               2 |     Tehran |       airbus
       LE260 |      200 |   qatar airways  |  null |      London |               9 |    Toronto |       boeing

(30 rows)
```

9,10- I updated table with adding a delay column and set 4 hours delay for one of the flights.and show that with the next line query.

11-

```
cqlsh:airline> DELETE FROM airline WHERE flight_numb='DF100';
cqlsh:airline> SELECT * FROM airline.airline;

 flight_numb | capacity | company         | delay | destination | flight_duration | origin     | type_aircraft
-------------+----------+-----------------+-------+-------------+-----------------+------------+---------------
       QJ872 |      150 | british airways |  null |      London |               1 |      Paris |        airbus
       OL555 |      100 | turkish airline |  null |    Istanbul |               6 | manchester |        airbus
       LB260 |      200 |   qatar airways |  null |      Berlin |              10 |    Toronto |        boeing
       MN711 |      300 |   qatar airways |  null |      Madrid |              10 |    Toronto |        boeing
       CX812 |      160 |    fly emirates |  null |       Dubai |               2 |   Istanbul |        airbus
       QC009 |      150 |    fly emirates |  null |    Istanbul |               5 |      Paris |        airbus
       ZW002 |      350 |    fly emirates |  null |     Toronto |               9 |   Istanbul |        boeing
       OL055 |      100 | turkish airline |  null |      Tehran |               8 | manchester |        airbus
       ZH092 |      350 | british airways |  null |     Toronto |               7 |     London |        boeing
       KW805 |      180 | british airways |  null |      London |               1 | manchester |        airbus
       KW809 |      180 | british airways |  null |      London |               1 | manchester |        airbus
       LW000 |      170 |       lufthansa |  null |   Amsterdam |               2 |      Milan |        boeing
       HF160 |       60 |   qatar airways |     4 |       Paris |               2 |      Milan |           jet
       PW722 |      300 | turkish airline |  null |      Lisbon |               3 |      Paris |        boeing
       PA722 |       70 | turkish airline |  null |      Tehran |               2 |   Istanbul |           jet
       LQ966 |      300 | turkish airline |  null |    Istanbul |               5 |     Tehran |        boeing
       LO987 |      150 |    fly emirates |  null |    Istanbul |               5 |     Berlin |        airbus
       LW060 |      170 |       lufthansa |  null |   Amsterdam |               2 |     Berlin |        boeing
       DF160 |      250 |   qatar airways |  null |        Rome |               7 |     Tehran |        boeing
       ZW092 |      350 |    fly emirates |  null |     Toronto |               7 |     London |        boeing
       AG571 |      300 |   qatar airways |  null |      London |              10 |    Toronto |        boeing
       LK871 |      300 |        iran air |  null |      Tehran |               7 |     London |        boeing
       LE760 |      350 |   qatar airways |  null |      Tehran |               6 |      Paris |        boeing
       ZH792 |      350 | british airways |  null |      London |               7 |    Toronto |        boeing
        W501 |      200 |       Mahan Air |  null |      Tehran |               7 |     London |        airbus
       RQ009 |       80 | turkish airline |  null |    Istanbul |               5 |     London |           jet
       LB060 |      200 |       lufthansa |  null |      Berlin |              10 |    Toronto |        boeing
       LM999 |      160 |    fly emirates |  null |       Dubai |               2 |     Tehran |        airbus
       LE260 |      200 |   qatar airways |  null |      London |               9 |    Toronto |        boeing

(29 rows)
```

Here I deleted a row with flight_numb=’DF100’

And my rows from 30 turned to 29

12-here I dropped the delay column:

```
(29 rows)
cqlsh:airline> ALTER TABLE airline DROP delay;
cqlsh:airline> SELECT * FROM airline.airline;

 flight_numb | capacity | company         | destination | flight_duration | origin     | type_aircraft
-------------+----------+-----------------+-------------+-----------------+------------+---------------
       QJ872 |      150 | british airways |      London |               1 |      Paris |        airbus
       OL555 |      100 | turkish airline |    Istanbul |               6 | manchester |        airbus
       LB260 |      200 |   qatar airways |      Berlin |              10 |    Toronto |        boeing
       MN711 |      300 |   qatar airways |      Madrid |              10 |    Toronto |        boeing
       CX812 |      160 |    fly emirates |       Dubai |               2 |   Istanbul |        airbus
       QC009 |      150 |    fly emirates |    Istanbul |               5 |      Paris |        airbus
       ZW002 |      350 |    fly emirates |     Toronto |               9 |   Istanbul |        boeing
       OL055 |      100 | turkish airline |      Tehran |               8 | manchester |        airbus
       ZH092 |      350 | british airways |     Toronto |               7 |     London |        boeing
       KW805 |      180 | british airways |      London |               1 | manchester |        airbus
       KW809 |      180 | british airways |      London |               1 | manchester |        airbus
       LW000 |      170 |       lufthansa |   Amsterdam |               2 |      Milan |        boeing
       HF160 |       60 |   qatar airways |       Paris |               2 |      Milan |           jet
       PW722 |      300 | turkish airline |      Lisbon |               3 |      Paris |        boeing
       PA722 |       70 | turkish airline |      Tehran |               2 |   Istanbul |           jet
       LQ966 |      300 | turkish airline |    Istanbul |               5 |     Tehran |        boeing
       LO987 |      150 |    fly emirates |    Istanbul |               5 |     Berlin |        airbus
       LW060 |      170 |       lufthansa |   Amsterdam |               2 |     Berlin |        boeing
       DF160 |      250 |   qatar airways |        Rome |               7 |     Tehran |        boeing
       ZW092 |      350 |    fly emirates |     Toronto |               7 |     London |        boeing
       AG571 |      300 |   qatar airways |      London |              10 |    Toronto |        boeing
       LK871 |      300 |        iran air |      Tehran |               7 |     London |        boeing
       LE760 |      350 |   qatar airways |      Tehran |               6 |      Paris |        boeing
       ZH792 |      350 | british airways |      London |               7 |    Toronto |        boeing
        W501 |      200 |       Mahan Air |      Tehran |               7 |     London |        airbus
       RQ009 |       80 | turkish airline |    Istanbul |               5 |     London |           jet
       LB060 |      200 |       lufthansa |      Berlin |              10 |    Toronto |        boeing
       LM999 |      160 |    fly emirates |       Dubai |               2 |     Tehran |        airbus
       LE260 |      200 |   qatar airways |      London |               9 |    Toronto |        boeing
```

Benefits and drawbacks: Cassandra is a scalable and highly available.it is also open source and we can access to that for free.besides flexibility.

In terms of drawbacks I can mention complexity and the query language which is Cql and we need to learn that as it is different from other databases.it takes some time to learn that.it doesn’t support joins and it’s not full ACID transaction.

Json

For this part,since I work part time at pizzaexpress, I decided to use that as my database.

Here is my json document code:


```
{
  "name": "pizzaexpress",
  "location": {
    "address": "bluewater shopping centre plaza",
    "city": "greenhithe",
    "county": "kent",
    "postcode": "DA99XU"
  },
  "phone": "00446748512",
  "website": "https://www.pizzaexpress.com",
  "hours": {
    "Monday": "Closed",
    "Tuesday": "5pm-10pm",
    "Wednesday": "5pm-10pm",
    "Thursday": "5pm-10pm",
    "Friday": "5pm-11pm",
    "Saturday": "10am-11pm",
    "Sunday": "10am-9pm"
  },
  "menu": [
    {
      "name": "Appetizers",
      "items": [
        {
          "name": "garlic bread",
          "description": "garlic butter with bread.",
          "price": 8.99
        },
        {
          "name": "Bruschetta",
          "description": "Toasted bread topped with fresh tomatoes, basil, and garlic.",
          "price": 10.00
        }
      ]
    },
```

```
   {
    "name": "meals",
    "items": [
     {
      "name": "american hot pizza",
      "description": "peperoni with papricorn.",
      "price": 16.75
     },
     {
      "name": "pasta carbonara",
      "description": "Pasta tossed with eggs, Parmesan cheese, and pancetta.",
      "price": 12.96
     },
     {
      "name": "chicken spicy pizza",
      "description": "chicken with some different pepper.",
      "price": 14.99
     }
    ]
   },
   {
    "name": "Desserts",
    "items": [
     {
      "name": "Chocolate  Cake",
      "description": " simple chocolate cake.",
      "price": 3.55
     },
     {
      "name": "cinnamon roll",
      "description": "Layers of cinnamon with white chocolate syrup.",
      "price": 6.30
     }
    ]
   }
  ]
}
```


name: The restaurant's name is 'pizzaexpress.
location: This object includes data about the restaurant's location, such as the address, city, county, and postcode.
Phone: "00446748512": This is the restaurant's number.

Website: "https://www.pizzaexpress.com" This is the restaurant's webpage.
hours:This object holds details on the restaurant's hours of operation, including the hours for every day of the week.
menu:This array of objects stands in for the many menu item categories at the restaurant. Each object has two properties: a "name" property (for example, "Appetizers") and a "items"

property, which is an array of objects that each represent a specific item in the category. Each item object has three properties: "name," "description," and "price.
this JSON document offers structured data on a restaurant, including the name, address, phone number, operating hours, and menu items with costs and descriptions.

Schema file:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "name": {"type": "string"},
    "location": {
      "type": "object",
      "properties": {
        "address": {"type": "string"},
        "city": {"type": "string"},
        "county": {"type": "string"},
        "postcode": {"type": "string"}
      },
      "required": ["address", "city", "county", "postcode"]
    },
    "phone": {"type": "string"},
    "website": {"type": "string"},
    "hours": {
      "type": "object",
      "properties": {
        "Monday": {"type": "string"},
        "Tuesday": {"type": "string"},
        "Wednesday": {"type": "string"},
        "Thursday": {"type": "string"},
        "Friday": {"type": "string"},
        "Saturday": {"type": "string"},
```

```
        "Sunday": {"type": "string"}
      },
      "required": ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"]
    },
    "menu": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "name": {"type": "string"},
          "items": {
            "type": "array",
            "items": {
              "type": "object",
              "properties": {
                "name": {"type": "string"},
                "description": {"type": "string"},
                "price": {"type": "number"}
              },
              "required": ["name", "description", "price"]
            }
          }
        },
        "required": ["name", "items"]
      }
    }
  },
  "required": ["name", "location", "phone", "hours", "menu"]
}
```

The version of the JSON Schema that was used in this file is indicated by the line "$schema": "http://json-schema.org/draft-07/schema#".

type: "object": This line shows that the JSON document should be an object.

properties: This object defines the properties that are anticipated in the JSON document.

name: This line specifies that the "name" property should be a string.

location: This object specifies the "location" property, which should be an object containing the values "address," "city," "county," and "postcode," each of which should be a string.

phone:This line specifies that the "phone" property should be a string.

website: This line specifies that the "website" property should be a string.

hours: This object defines the "hours" property, which must be an object with properties for every day of the week (e.g., "Monday," "Tuesday," etc.), each of which must be a string.

Menu: This object defines the "menu" attribute, which is an array of other objects. A string property called "name" and an array property called "items" should both be present on every object. A string "name," a string "description," and a string "price" property should each be included in an item object.

required: This array lists the properties in the JSON document that must be present.

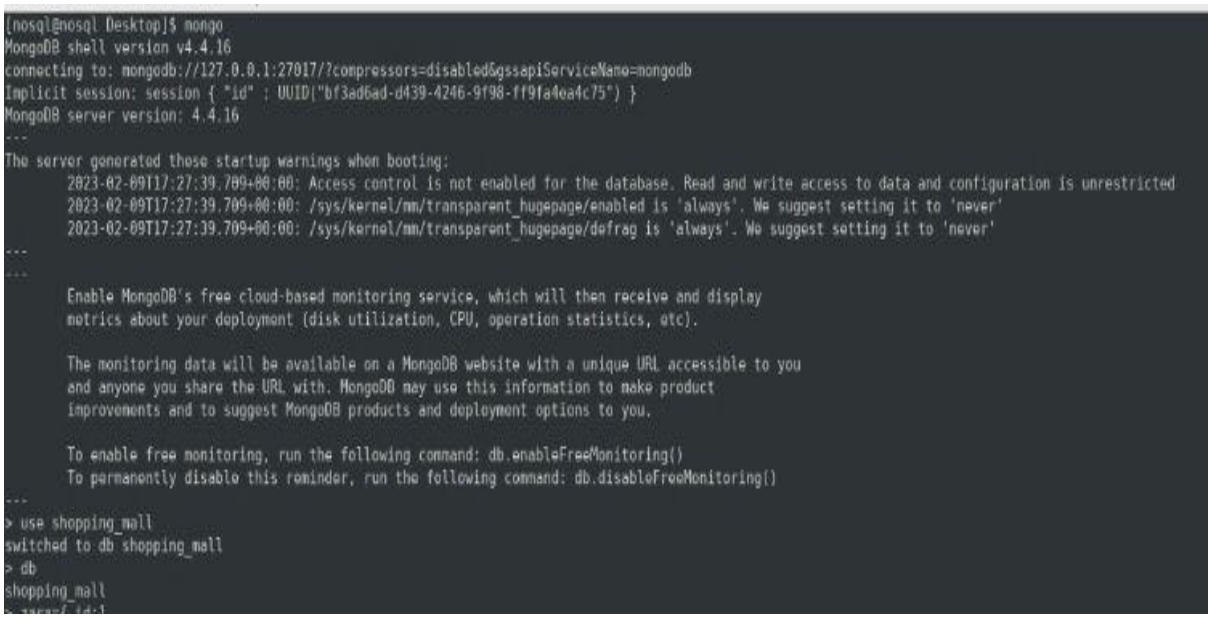
This JSON Schema file describes the structure and predicted data types for a JSON document for a restaurant and may be used to verify that a JSON document meets with this schema.

I added the files as appendix in the end.

Benefits and drawbacks: easy to read and write and also easy to parse.it’s supported by most programming languages and framework. About drawbacks I can say it will be a challenge for large data as we have to do data validation and verification manually.also there is no query language for that.

mongoDB

For mongoDB I created a database about shopping mall.



I’ve created this database with command:

use shopping_mall

db

then, I insert some values into different stores. The keys are _id,category,location,price_range,country of origin and foundation year.

```
shopping_mall
> zara={_id:1,
... category:"clothing",
... location:"bluewater,second floor",
... price_range:"not bad",
... country:"spain",
... year:1975
... }
{
        "_id" : 1,
        "category" : "clothing",
        "location" : "bluewater,second floor",
        "price_range" : "not bad",
        "country" : "spain",
        "year" : 1975
}
```

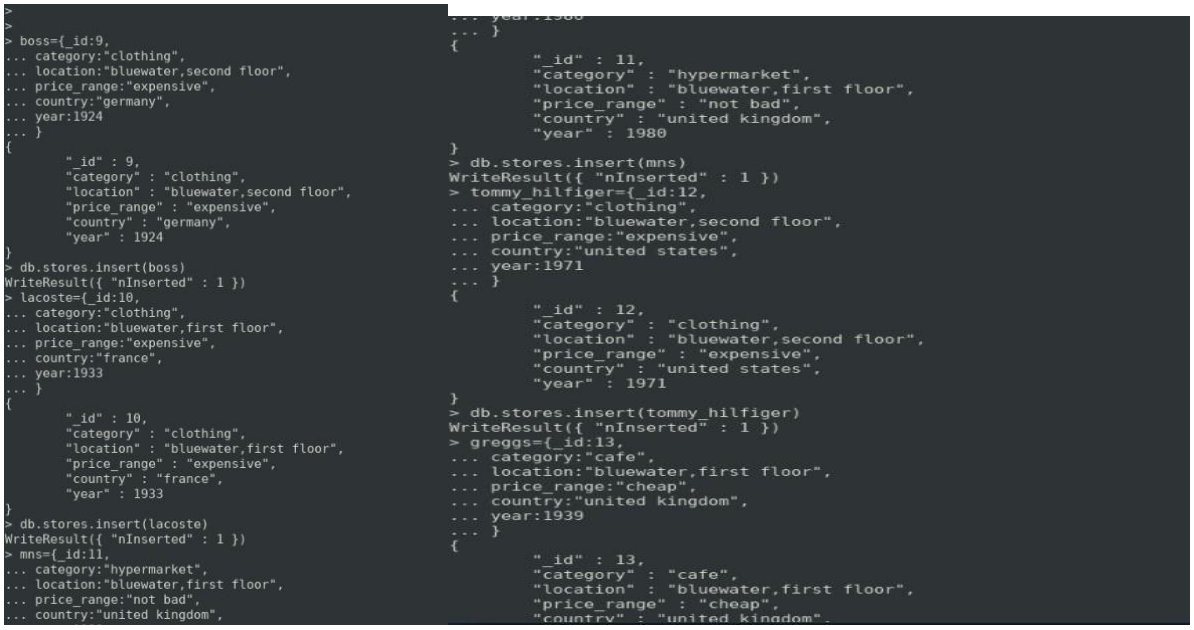
```
> db.stores.insert(zara)
WriteResult({ "nInserted" : 1 })
> db.stores.find()
{ "_id" : 1, "category" : "clothing", "location" : "bluewater,second floor", "price_range" : "not bad", "country" : "spain", "year" : 1975 }
> mango={_id:2,
... category:"clothing",
... location:"bluewater,second floor",
... price_range:"expensive",
... country:"spain",
... year:1984
... }
{
        "_id" : 2,
        "category" : "clothing",
        "location" : "bluewater,second floor",
        "price_range" : "expensive",
        "country" : "spain",
        "year" : 1984
}
> db.stores.insert(mango)
WriteResult({ "nInserted" : 1 })
> db.stores.find()
{ "_id" : 1, "category" : "clothing", "location" : "bluewater,second floor", "price_range" : "not bad", "country" : "spain", "year" : 1975 }
{ "_id" : 2, "category" : "clothing", "location" : "bluewater,second floor", "price_range" : "expensive", "country" : "spain", "year" : 1984 }
> apple={_id:3,
... category:"tech",
... location:"bluewater,first floor",
... price_range:"expansive",
... country:"united states",
... year:1976
... }
{
        "_id" : 3,
        "category" : "tech",
        "location" : "bluewater,first floor",
        "price_range" : "expensive",
        "country" : "united states",
        "year" : 1976
}
> db.stores.insert(apple)
WriteResult({ "nInserted" : 1 })
```

```
}
> db.stores.insert(pret)
WriteResult({ "nInserted" : 1 })
> starbucks={_id:19,
... category:"cafe",
... location:"bluewater,first floor",
... price_range:"expensive",
... country:"united states",
... year:1866
... }
{
        "_id" : 19,
        "category" : "cafe",
        "location" : "bluewater,first floor",
        "price_range" : "expensive",
        "country" : "united states",
        "year" : 1866
}
> db.stores.insert(starbucks)
WriteResult({ "nInserted" : 1 })
> ck={_id:20,
... category:"clothing",
... location:"bluewater second floor",
... price_range:"expensive",
... country:"united states",
... year:1942
... }
{
        "_id" : 20,
        "category" : "clothing",
        "location" : "bluewater second floor",
        "price_range" : "expensive",
        "country" : "united states",
        "year" : 1942
}
> db.stores.insert(ck)
WriteResult({ "nInserted" : 1 })
> db.stores.find()
{ "_id" : 1, "category" : "clothing", "location" : "bluewater,second floor", "price_range" : "not bad", "country" : "spain", "year" : 1975 }
{ "_id" : 2, "category" : "clothing", "location" : "bluewater,second floor", "price_range" : "expensive", "country" : "spain", "year" : 1984 }
{ "_id" : 3, "category" : "tech", "location" : "bluewater,first floor", "price_range" : "expensive", "country" : "united states", "year" : 1976 }
```

```
{
        "_id" : 16,
        "category" : "clothing",
        "location" : "bluewater,first floor",
        "price_range" : "not bad",
        "country" : "united kingdom",
        "year" : 1855
}
> db.stores.insert(newlook)
WriteResult({ "nInserted" : 1 })
> next={_id:17,
... category:"clothing",
... location:"bluewater,second floor",
... price_range:"not bad",
... country:"united kingdom",
... year:1989
... }
{
        "_id" : 17,
        "category" : "clothing",
        "location" : "bluewater,second floor",
        "price_range" : "not bad",
        "country" : "united kingdom",
        "year" : 1989
}
> db.stores.insert(next)
WriteResult({ "nInserted" : 1 })
> pret={_id:18,
... category:"cafe",
... location:"bluewater,first floor",
... price_range:"not bad",
... country:"united kingdom",
... year:1976
... }
{
        "_id" : 18,
        "category" : "cafe",
        "location" : "bluewater,first floor",
        "price_range" : "not bad",
        "country" : "united kingdom",
        "year" : 1976
```

I created 20 documents and show them with this query:

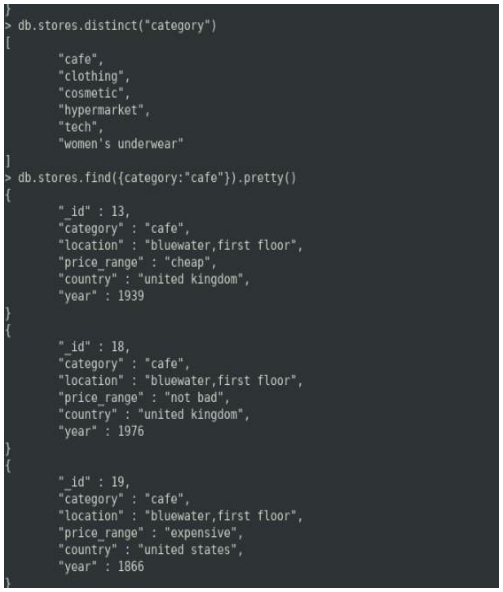
Db.stores.find()



```
>
>
> boss={_id:9,
... category:"clothing",
... location:"bluewater,second floor",
... price_range:"expensive",
... country:"germany",
... year:1924
... }
{
        "_id" : 9,
        "category" : "clothing",
        "location" : "bluewater,second floor",
        "price_range" : "expensive",
        "country" : "germany",
        "year" : 1924
}
> db.stores.insert(boss)
WriteResult({ "nInserted" : 1 })
> lacoste={_id:10,
... category:"clothing",
... location:"bluewater,first floor",
... price_range:"expensive",
... country:"france",
... year:1933
... }
{
        "_id" : 10,
        "category" : "clothing",
        "location" : "bluewater,first floor",
        "price_range" : "expensive",
        "country" : "france",
        "year" : 1933
}
> db.stores.insert(lacoste)
WriteResult({ "nInserted" : 1 })
> mns={_id:11,
... category:"hypermarket",
... location:"bluewater,first floor",
... price_range:"not bad",
... country:"united kingdom",
... year:1980
... }
{
        "_id" : 11,
        "category" : "hypermarket",
        "location" : "bluewater,first floor",
        "price_range" : "not bad",
        "country" : "united kingdom",
        "year" : 1980
}
> db.stores.insert(mns)
WriteResult({ "nInserted" : 1 })
> tommy_hilfiger={_id:12,
... category:"clothing",
... location:"bluewater,second floor",
... price_range:"expensive",
... country:"united states",
... year:1971
... }
{
        "_id" : 12,
        "category" : "clothing",
        "location" : "bluewater,second floor",
        "price_range" : "expensive",
        "country" : "united states",
        "year" : 1971
}
> db.stores.insert(tommy_hilfiger)
WriteResult({ "nInserted" : 1 })
> greggs={_id:13,
... category:"cafe",
... location:"bluewater,first floor",
... price_range:"cheap",
... country:"united kingdom",
... year:1939
... }
{
        "_id" : 13,
        "category" : "cafe",
        "location" : "bluewater,first floor",
        "price_range" : "cheap",
        "country" : "united kingdom",
```

```
        "price_range" : "expensive",
        "country" : "germany",
        "year" : 1892
}
> db.stores.insert(diesel)
WriteResult({ "nInserted" : 1 })
> victorias_secret={_id:8,
... category:"women's underwear",
... location:"bluewater,second floor",
... price_range:"expensive",
... country:"united states",
... year:1997
... }
{
        "_id" : 8,
        "category" : "women's underwear",
        "location" : "bluewater,second floor",
        "price_range" : "expensive",
        "country" : "united states",
        "year" : 1997
}
> db.stores.insert(victorias_secret)
WriteResult({ "nInserted" : 1 })
> boss={_id=9,
... category:"clothing",
... location:"bluewater,second floor",
... price_range:"expensive",
... country:"germany",
... year:1924
... }
uncaught exception: SyntaxError: missing : after property id :
@(shell):1:9
> boss={_id:9,
... boss={_id:9, category:"clothing", location:"bluewater,second floor", price_range:"expensive",
... db.stores.insert(boss)
... ;
... db.stores.find()
... db.stores.find();
```

And then I implemented some queries on those data

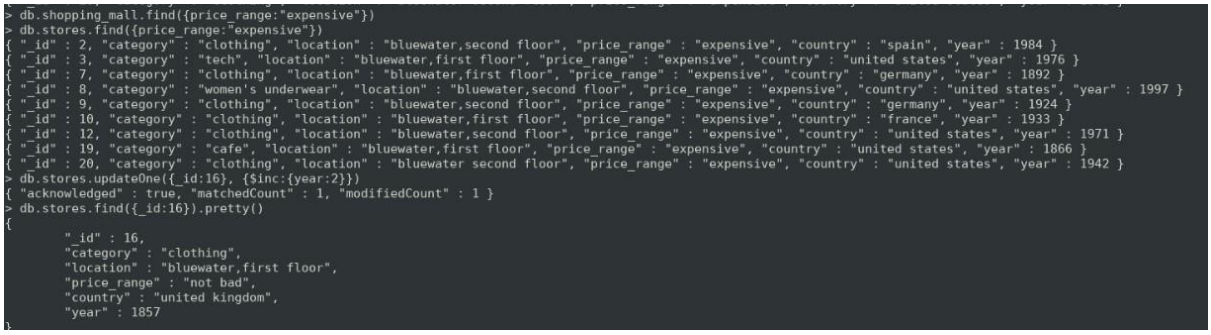


1,2- with db.stores.distinct(“category”), I obtained all categories that I used for the database.

In this database I used café,clothing,cosmetic,hypermarket,tech and women’s underwear as the category.

Then with db.stores.find({category:”café”}).pretty() I got all documents containing café as the category.

Pretty() command show the documents better and more user friendly.



3,4-here I used two other queries,

Db.stores.find({price_range:”expensive”}) gave me all documents which their price range were expensive.

Db.stores.find({_id:16}).pretty

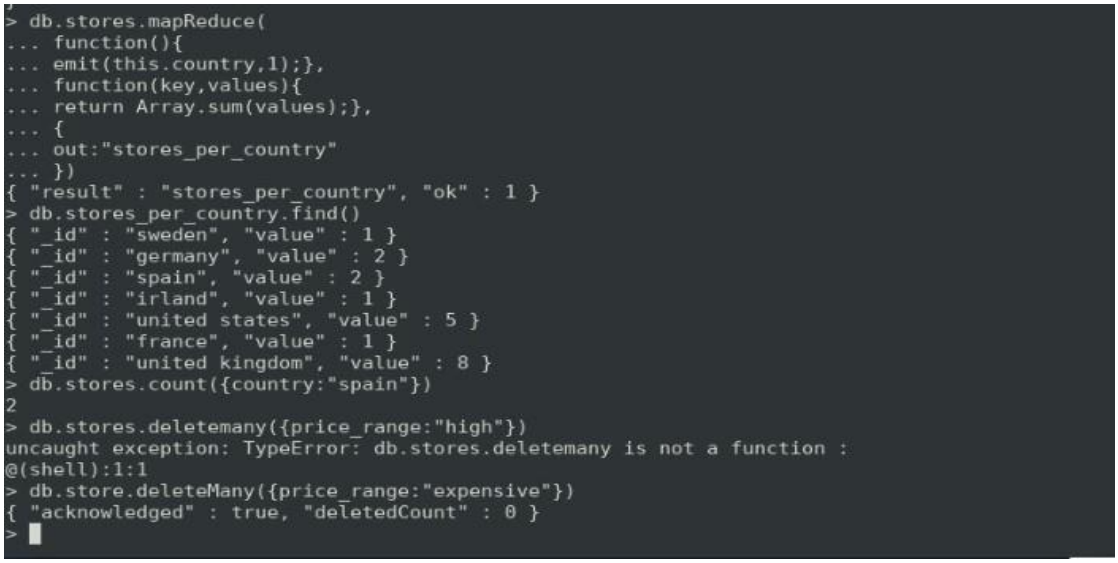
Showed the document with id=16.

5,6,7-I implemented 3 queries here.

The first one which is aggregate command, I got the average of the all years I used in this database which is 1941.1

The next query, I used sort({year:-1}).limit(1) which obtained the nearest year(2007)

And another query is pretty() which showed the all documents in a prettier way.



8,9,10-

Here again I used 3 other queries

The first query is mapreduce.

MongoDB uses MapReduce, a data processing method, to analyze big datasets. It includes converting key-value pairs of data from a collection into a summary value.

Here I used this command to show for each country how many stores I have in my database.

We see that the output of the mapreduce here showed us the value for each country. For example we had 5 stores originated from united states.

The next query counted the stores originated from spain which is 2.

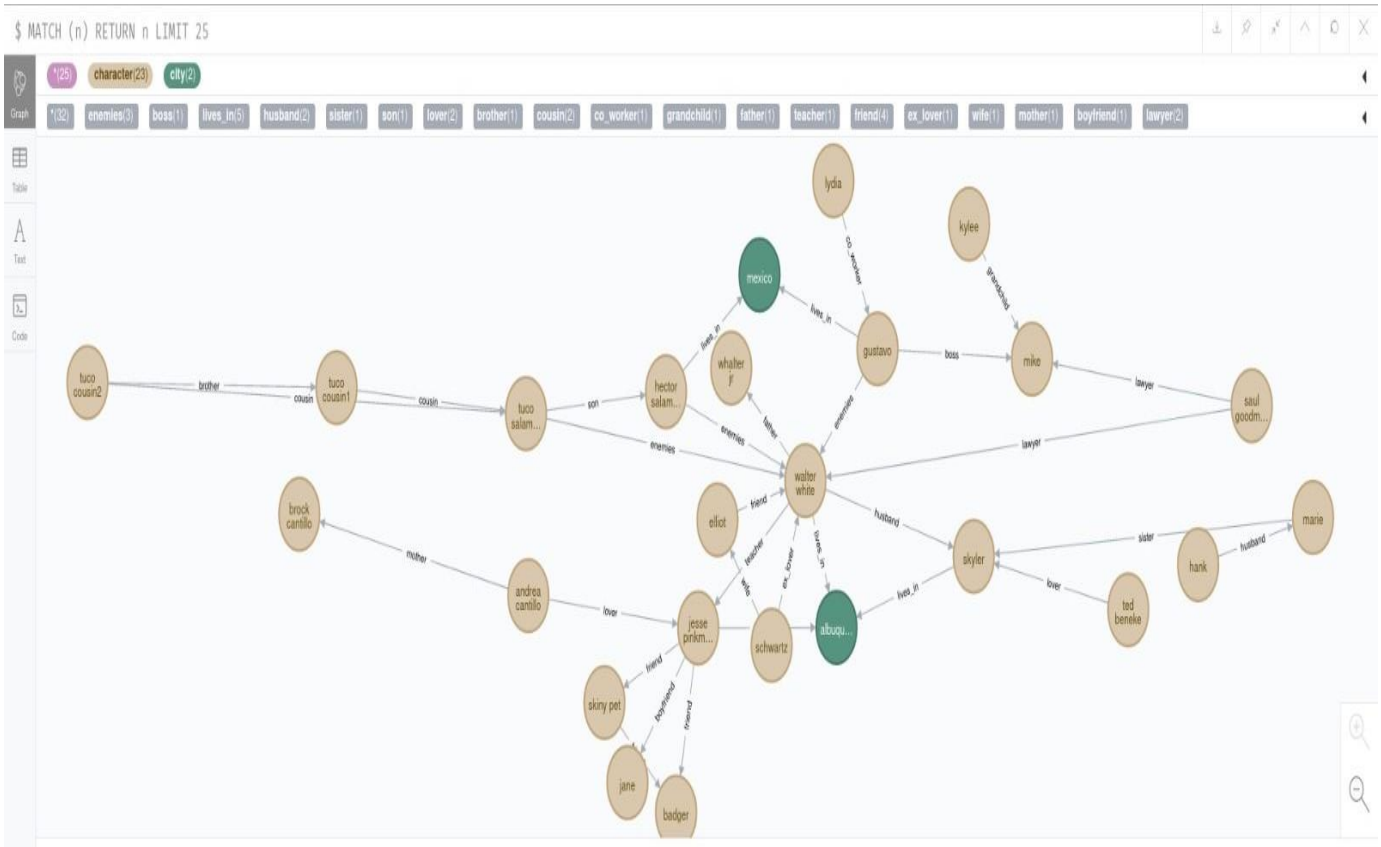
The last query deleted all documents with price range=expensive.

Benefits and drawbacks: it is flexible the query language is so rich and strong it envolves map-reduce function.as drawbacks I can say that it doesn't offer full ACID compliance.limited data security and because of spliting data into small chunks sometimes it leads to slow query performance.

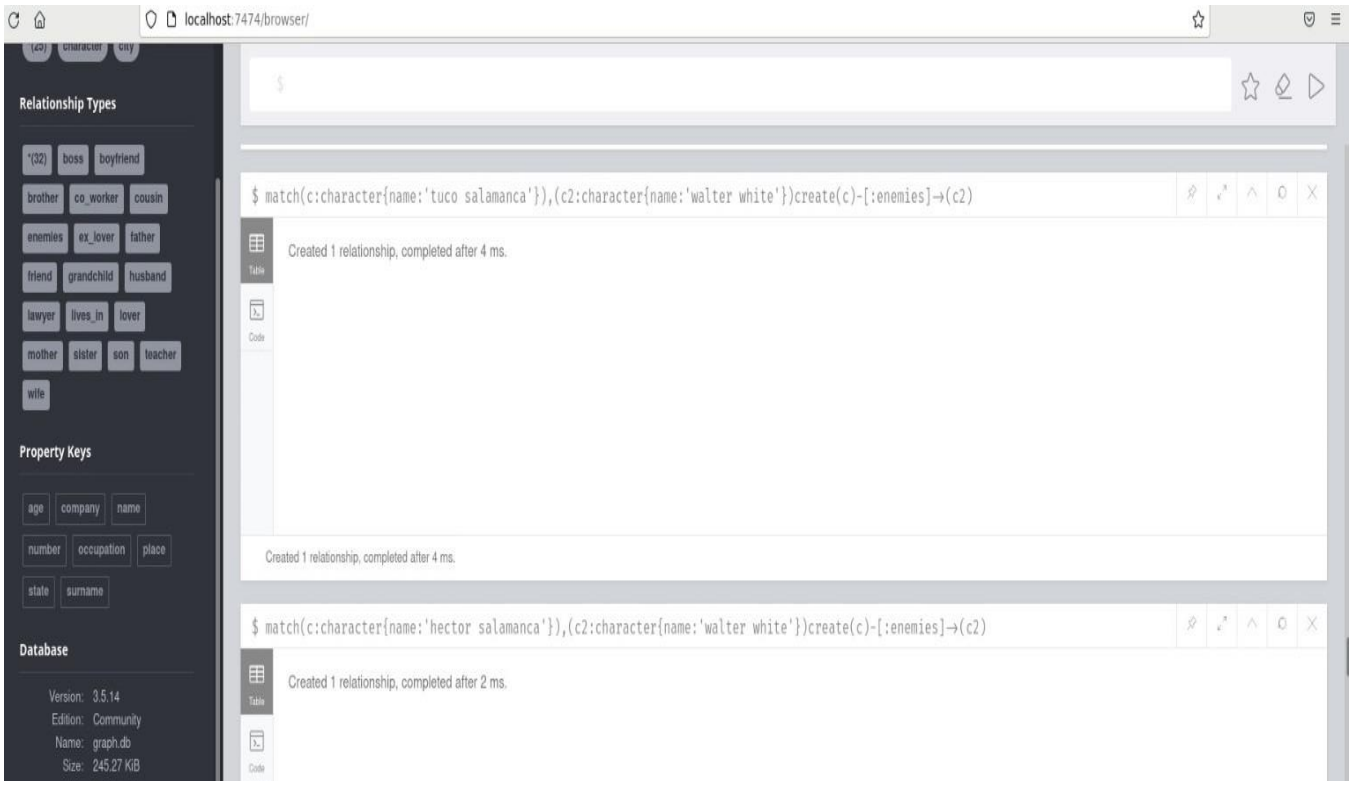
Neo4j

For this part, I used my favourite series as a sample database. This is a graph for breaking bad series which is the best series I have ever seen.

Here is my complete graph:



I created 25 nodes with relationships between characters in the series and two cities which are Mexico and Albuquerque in new Mexico state.



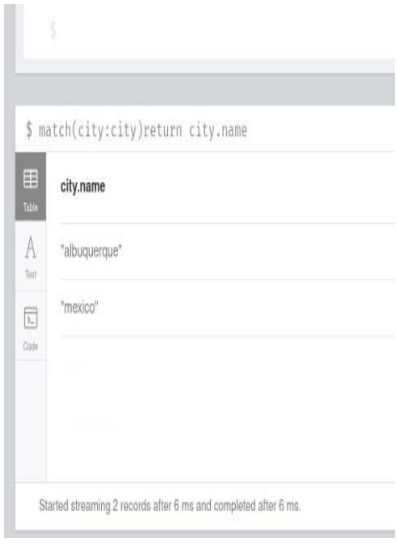
I created the relationships between character with this query:

Match(c:character{name: }),(c2:character{name: })created(c)-[:]->(c2)

Here I used this query to obtain all character I created for the graph.

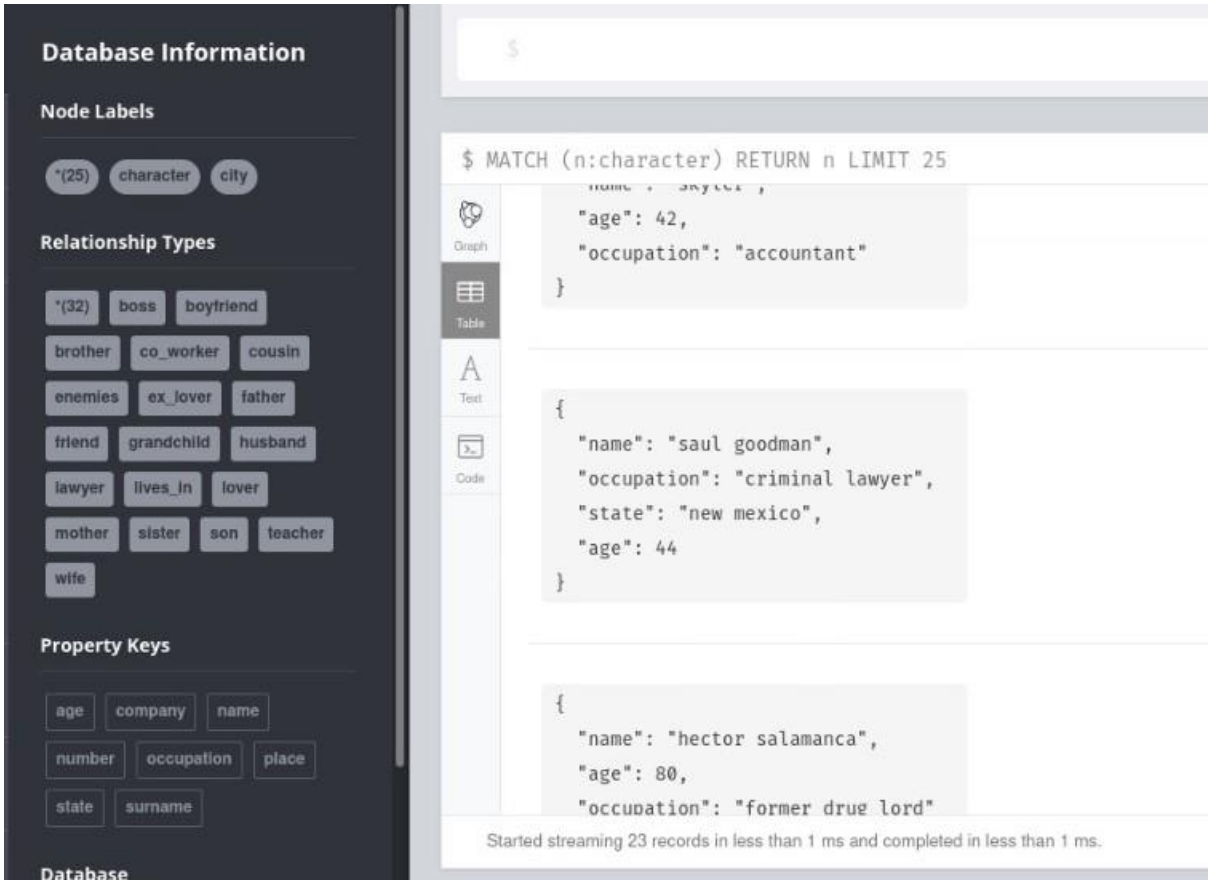
Match(c:character)return c.name

And here I obtained the cities which I used as nodes.



The next query I implemented is a query which shows all characters who lives in Albuquerque.i created a lives_in relationship between these characters below and city:Albuquerque before.

Also for one of my character , saul goodman I added state label as well



As we can see here I just added state feature for saul goodman and others have just name, occupation and age.

Benefits and drawbacks:it is easy to query and have graph processing. We can see what is happening In our database.also it is parallel processing.for drawbacks ,I can mention that cypher is easy for simple queries but for complicated and more complex data using cypher can be hard.

APPENDIX: for json part