****

**School of Computing Science and Digital Media**

Faculty of Design and Technology

**Coursework Assignment**

|  |  |
| --- | --- |
| Surname | Tait |
| First name | Andrew |
| Matriculation Number | 1504693 |
| Contact phone number | - |
| Course + Year | MSc Data Science |
| Module Co-ordinator | Rob Lothian |
| Module Number + Name | CMM534 Big Data Analytics and Visualization |
| Coursework Title | CMM534 - Coursework Assignment |
| Coursework Part | - |
| Due Date | 4th May 2018 |

**Declaration** \*\* *This* ***must*** *be affirmed by signing below*

**I confirm**

* **that the work undertaken for this assignment is entirely my own and that I have not made use of any unauthorised assistance**
* **that the sources of all reference material has been properly acknowledged.**

|  |  |
| --- | --- |
| Student Signature |  |
| Date Submitted | 4th May 2018 |

|  |  |
| --- | --- |
| **Marker's Comments** | |
| **Marker** | **Grade** |

\*\* An extract from the University Regulations

6. **Academic Misconduct**

Refer also to Schedule 3.3 of this Regulation for guidance on this procedure.

6.1 **Academic Misconduct** is defined as any attempt by students to gain an unfair advantage in assessments and examinations. Examples of academic misconduct include plagiarism, cheating, falsifying data, collusion, bribery or attempted bribery, personation or any other activity intended to provide an unfair advantage.

(i)**Plagiarism** is the practice of presenting the thoughts or writings of another or others as original, without acknowledgement of their source(s). All material used to support a piece of work should be carefully referenced and should not normally be copied directly unless as an acknowledged quote. Text translated into the words of the individual student should in all cases acknowledge the source.

(ii)**Cheating** includes:

1. the taking of any unauthorised material into an examination;
2. obtaining copy of “unseen” papers in advance of an examination;
3. communicating or attempting to communicate in any way with another student during an examination;
4. copying or attempting to copy from another student during an examination or in the production of coursework;
5. wilful deception in any element of an examination or assessment.

(iii) **Falsification of data** consists of the misrepresentation of the results of experimental work or the presentation of results from fictitious work.

(iv) **Collusion** is the representation of unauthorised group work as that of an individual student.

(v) **Bribery** is the paying, offering or attempted exchange of an inducement for information or material intended to advantage the recipient in an examination or assessment.

(vi) **Personation** consists of a substitute taking the place of a student in an examination.

**A student who aids and abets a fellow student to commit academic misconduct shall be deemed to have committed academic misconduct and will be dealt with accordingly.**

Table of Contents

[Task 2 – Neo4j 2](#_Toc513206918)

[Database Design 2](#_Toc513206919)

[Nodes 2](#_Toc513206920)

[Relationships 2](#_Toc513206921)

[Future work and Scalability. 3](#_Toc513206922)

[Task 3 – Technical Paper 4](#_Toc513206923)

[Redis 4](#_Toc513206924)

[MongoDB 5](#_Toc513206925)

[Comparison between Redis and MongoDB 5](#_Toc513206926)

[References 6](#_Toc513206927)

Table of Figures

[Figure 1 Neo4j Graph DB Design (Hierarchy) 2](#_Toc513206928)

# Task 2 – Neo4j

Database Design

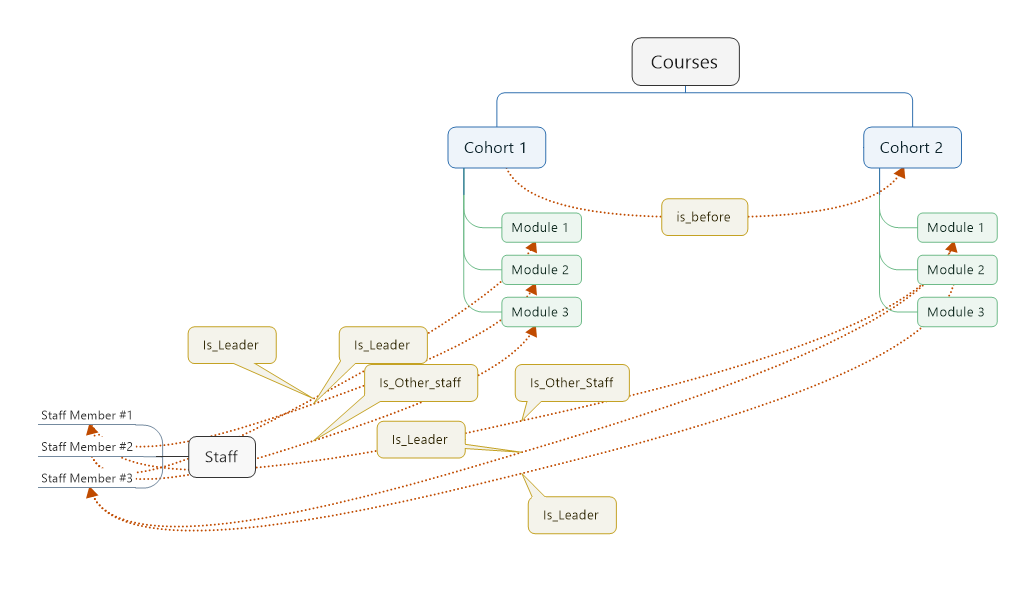


Figure 1 Neo4j Graph DB Design (Hierarchy)

In order to design the graph database, I first did a rough hierarchy design shown in figure 1 above.

It starts with the courses at the top of the hierarchy, followed by cohorts which in turn can have many modules taught in then. Finally, the staff members can be either the leader or other staff on any module.

### Nodes

* Courses
  + The courses consist of the course ID and title.
* Cohorts
  + Attributes: cohort ID, name, start date and total number of students.
* Modules
  + Attributes: module ID, code and title.
* Staff
  + Attributes: staff name.

### Relationships

* belongs\_to
  + This is used to connect the cohorts to their courses.
  + For example
    - Data science semester 1 2017/2018 -> Data Science
* is\_before
  + This is used to connect different cohorts together.
  + For instance, Data Science Semester 1 is before Data Science Semester 2.
    - This also allows for the first semester to share modules with the second semester.
* taught\_in
  + This links the modules with the cohorts.
  + It also has an attribute, semester which defines which semester in the course structure it is.
  + For example, (M004)-[:taught\_in{semester:'SM001'}]->(CH001) will indicate that module M004 is taught in the first semester for Cohort 1.
* Is\_leader
  + Defines if a staff member is the lead lecturer for a module.
* is\_other\_staff
  + Defines if a staff member is an assistant lecturer for a module.

### Future work and Scalability.

The graph database was designed with scalability and future additions in mind.

The process of adding new content to the graph database, would go similar to the following:

A new course is will be added first e.g. ‘Application Software Development’ then the cohorts for each semester (September/January intakes) are inserted and linked to the new course. Modules are inheritably linked to each cohort to make up the course structure.

Finally, staff members are linked to the modules, defining if they are the lead or assistant lecturer.

New modules and staff members can be added with ease without changing the structure defined above. For instance, when a new module is added it will then be associated with the cohorts and staff members respectively.

When it came to deciding on how the semesters would be setup, I went with putting the semester value in the :taught\_in relationship as an attribute. Since the cohort have their start date as an attribute too and they are in turn linked to each other with the :is\_before relationship. So, it will tell that each module is taught in a certain cohort within semester 1 for example.

The alterative route to this would have been to add the Semesters as an additional node and linking the modules/cohorts with a separate relationship. However, this proven to complicate the sample queries so I reverted to the latter approach.

# Task 3 – Technical Paper

### Redis

Redis is a key value type store, it is built as in-memory noSQL database. It supports a variety of data structures including strings, lists, sets, hashes and sorted sets.

**Use cases**

There are wide variety of use cases for Redis, as outlined below. [4]

Very Large Datasets, due to the on-going demand for processing large datasets with quick results. Redis allows this feature to be a reality to its operations running 90% of its data in flash and 10% in ram. The performance boosts allow Redis to perform 3 million operations a second with less than 1 millisecond of latency.

Caching with Redis, allows applications to run a very high speeds due to it’s easier horizontal scaling of the application stack, bellow 1 millisecond latency and very low number of instances as possible.

Spark and Redis, by combine both technologies together the performance increases are very noticeable with up 10 ten times less latency compared to the traditional HDFS framework.

**Performances/Features**

Redis has several performance and feature enhancements which make it very competitive compared to its traditional SQL databases counterparts. This is down it’s high performance of queries, scalability of data and flexibility of its schema. It also uses a primary key with some value as its database structure. [1]

**Limitations**

As Redis is a key-value store, it’s best to avoid if we want to use any of the following: composite key, joins or derived table operations on the database. [1]

### MongoDB

MongoDB is a Document Store Database (a higher version of key-value store), where the values are saved as documents in which that the data has complex structure such as JSON. It’s one of the most known NoSQL databases of this type.

**Use cases**

There are a variety of use case for MongoDB, a few examples are outlined below. [3]

Operational Intelligence use cases include the store and handing of real time data, such as storing log data and Hierarchical Aggregation.

Produce Data Management, due to the flexible scheme of MongoDB this makes it particularly useful in storing product data. Such as product catalogue, inventory management and category hierarchy.

Content Management systems, MongoDB can also be used in familiar situations that traditional SQL databases are typically used for. For instance, storing comments on a website, metadata and assessed management.

**Performances/Features**

MongoDB has several performance advantages on it’s traditional SQL counterparts, for instance it has a flexible schema structure, fast performance of queries, high scalability of data and uses JSON in form of a tree as the database structure. It supports sharing and partitioning with order preserving, horizonal scaling (adding additional servers for load balancing/redundancy) and uses a master slave hierarchy for replication.

**Limitations**

As MongoDB is a document store type database, it’s best to avoid if we want to use any of the following: denormalization, unordered key, composite key, composite aggregation, joins or derived table operations on the database [1]

### Comparison between Redis and MongoDB

While there over 200 different kinds of NoSQL database solutions out there, however I will only compare two different systems: Redis and MongoDB. As outlined in the two sections above, are the characteristics and uses cases of each system.

Now it’s time to compare both Redis and MongoDB, to do this I’ll be referencing the “Performing Comparisons Between Five NoSQL Databases” paper [2]

Each of the tests carried out were on 5 virtual machines with the same hardware specifications (Ubuntu Server).

The first test was to compare the loading time for 100,000 records. Redis was the fastest with 7931 milliseconds and MongoDB followed with 10332 milliseconds.

The second test was checking executing time for 100,000 records. To do these three separate workloads were carried out: workload A - Update heavy wit 50/50 read/update operations, workload C, 100% read only and workload H 100% Update only operations.

Redis performed rather well in these tests with an average execution time of 1000 milli seconds across each workload, except for workload H were it slightly increased. Followed again with MongoDB with an average execution time of 1900 milliseconds across all workloads, while slightly decreasing in workload C.

The final test was record which database had the highest throughout of loading data with an increasing record count. Redis was again the highest performing out of the 5 databases with a throughput (operations/per second) of 20000. While MongoDB didn’t performance very well in comparison with an average throughput value of less than 10000.

While the tests above conclude that Redis was the highest performing database out of the two, it’s worth noting that in terms of scalability MongoDB has the advantage of working with extremely large datasets while Redis is limited to it’s performance of running in memory/flash storage.

References

[1] A. Gupta, S. Tyagi, N. Panwar, S. Sachdeva and U. Saxena, "NoSQL databases: Critical analysis and comparison," 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), Gurgaon, 2017, pp. 293-299.  
doi: 10.1109/IC3TSN.2017.8284494

URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8284494&isnumber=8284439>

[2] E. Tang and Y. Fan, "Performance Comparison between Five NoSQL Databases," 2016 7th International Conference on Cloud Computing and Big Data (CCBD), Macau, 2016, pp. 105-109.  
doi: 10.1109/CCBD.2016.030

URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7979888&isnumber=7979858>

[3]Use Cases¶. (MongoDB). Retrieved May 3, 2018, from <https://docs.mongodb.com/ecosystem/use-cases/>

[4] “Redis Enterprise Use Cases.” Retrieved May 3, 2018 *Redis Labs*, <https://redislabs.com/solutions/use-cases/>