.

School of Computing, Engineering and Built Environment

Big Data Platforms

Level: M

Module Code: MMI227050

Coursework 1

Issue: 26th February 2022
This coursework comprises 50% of the overall mark for the module.

Hand-in date: 14th March 2022

Attention is drawn to the university regulations on plagiarism. Whilst discussion of the coursework between individual students is encouraged, the actual work has to be undertaken individually. Collusion may result in a zero mark being recorded for the coursework for all concerned and may result in further action being taken.

PART A: CREATING A DOCUMENT DATABASE

In this task you are given an application scenario and some use cases associated with that scenario, and you are required to design a data model for a document store to support these use cases and to implement this model as a MongoDB database. Note that you are not required to design or implement the application, just the data model and data store.

Scenario – online auction

A data store is required for an online auction application. It is proposed that MongoDB will be used due to its capability to support massive numbers of users and activity in the application.

In the application, a user will be able to create an auction, specifying the item to be sold which should have a name, category and description. The auction will also have a closing date and the starting bid price. The auction status will initially be "open" and change to "ended" when the end date is reached. Any user can make a bid on an item, specifying the bidder, the date/time and the bid amount. When a bid is made, the current price for the auction is updated. User information will include user name, email address and location.

The data store should be optimized around the following key use cases:

1. Find all auctions for items in a specified category
2. Find all auctions created by a specified user
3. Make a new bid on an auction
4. Show all the details of an auction including bids and status

A1. Creating the data model (10 marks)
Your documentation should specify the names of the collections in your data model design, and illustrate your schema with an example document for each collection, represented in JSON form. The properties of the collections should correspond to the data referred to in the scenario above. Your documentation should justify the choice of collections and the use of linking, embedding or any other data modeling approach you have chosen.

A2. Creating the database (10 marks)
Create a MongoDB database called *cw1parta*

Use MongoDB database methods, e.g. *db.<collection>.insertOne*, to create collections, based on your data model, and insert test data in the form of documents to represent the following:

- at least 5 users
- at least 10 auctions – there should be at least 3 different categories of items within these auctions
- at least 2 bids for each auction – do not have exactly the same number of bids for each

A3. Queries (10 marks)
You should devise and run a MongoDB query corresponding to each of the key use cases.
You should also devise and run a MongoDB aggregation pipeline query to find the average current price for each item category

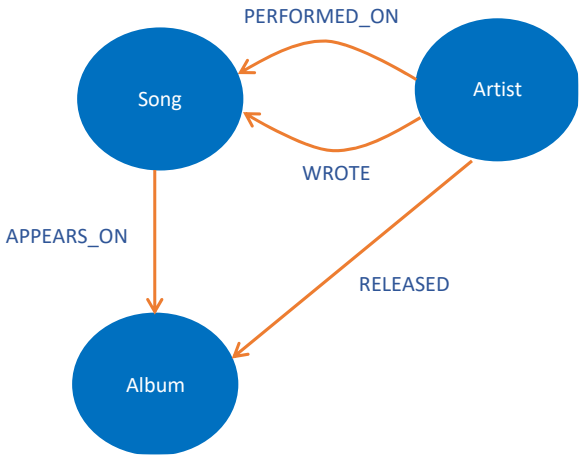
PART B: CREATING A GRAPH DATABASE

In this task you are given a graph data model. You are required to create and populate a Neo4j database which follows that model. You are also required to devise and run Cypher queries to retrieve information from that database.

Scenario – Music library data model

A graph database is required to represent items in a music library and the artists and musicians credited on these, in order to explore connections within the data.

The following diagram illustrates the node and relationship labels which should used in the Graph database that you implement:



The following table shows properties of nodes, and of edges where appropriate. Edge labels in the diagram that are not included in the table can be assumed to have no properties.

Node or edge label	Properties
Song	title, length
Album	title
Artist	name
PERFORMED_ON	performance_type

Further notes on this model:

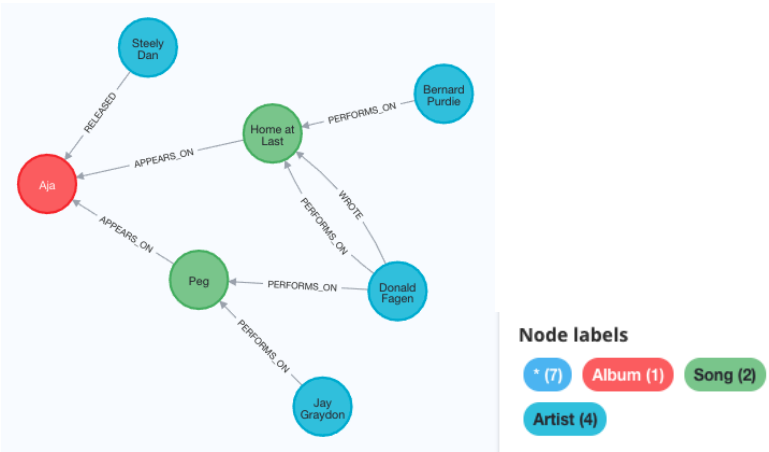
- A **Song** is a specific recording of a piece of music
- A collection of Songs by an **Artist** appear on an **Album**, which is released by that Artist.
- An Artist can also be involved with a Song by performing on it, or writing (or co-writing) the Song
- When an Artist performs on a song, the relationship may have a property to specify the *performance type*, i.e. "guitar", "vocals"

B1. Creating the database (10 marks)
For this task you can use the Neo4J Sandbox or Neo4J Desktop. Create a Neo4J database called *cw1partb.*

Create nodes and edges, based on your data model, to represent the following:

- 2 albums, each with at least 3 songs – the artist for each of the 2 albums should be different.
- 3 artists for each song, one of whom has written the song:
    - you can include some repetition, e.g. where the same artist performs on more than one song, but do not have exactly the same set of musicians for every song on an album
    - include cases where an artist has both performed on and written the song,
    - make sure the performance type is included in the appropriate relationships

As an example, the following screenshot from the Neo4j browser shows a graph with 1 album and 2 songs. The name property of each node is shown in this view, but the other properties are not. You should choose your own data (which can be real or made-up).



You should use Cypher statements to create nodes and edges, and record each statement that you use by copying and pasting the code into your report.

B2. Queries (10 marks)
You should devise and run a Cypher query to return each of the following data from the graph:

1. all the nodes and edges in the database
2. all the songs on a specified album
3. all the artists who have both written and performed on any song
4. all the artists who have played a specified instrument on a specified album
5. the total length of a specified album

Record each query by copying and pasting the code into your report. For each query also copy the query results in text format and paste into your report. It is suggested that you use a monospaced font such as Courier to display the results. For query #1, also take a screenshot of the result in graph mode, and insert this image in your report.

DELIVERABLES AND MARKING SCHEME

Report

You must submit a report, as a PDF or Word document, containing the following:

Part A. Document data			
A1	Data model	• Schema design, illustrated by an example document from each collection • Justification for design • *There is no single correct solution. Marks will be awarded for completeness and validity of design and for justification relevant to the presented design.*	10 marks
A2	Creating the database	• Paste the database method calls you use to create and populated the MongoDB database, and the output from running each of these, into your report. • *Marks will be awarded for successful creation of the collections in the data model, with suitable and sufficient test data.*	10 marks
A3	Queries	• For each of the four use cases and the aggregate query, paste your query code and the output from the queries into your report • *Marks will be awarded for valid design and successful execution for each of the 5 specified queries*	10 marks
Part B. Graph Data			
B1	Creating the database	• Paste the Cypher statements you used to add data to the Neo4j database, and the output from running these into your report. • *Marks will be awarded for successful creation of suitable and sufficient test data that adheres to the given data model.*	10 marks
B2	Queries	• For each of the specified queries, paste your Cypher query and the results in graph format and text format (query #1), or text format only (all others) • *Marks will be awarded for valid design and successful execution for each of the 5 specified queries*	10 marks
Total			50 marks

Data

You must also submit the contents of the MongoDB database you create in Part A. Marks will not be awarded for the data but this will be used for confirmation of part A2.

Use *mongoexport* to create a JSON document for each collection in your MongoDBdatabase in Part A. Name the output file for each collection as follows: *<collectionname>.json.* Combine these into a single ZIP file and submit this ZIP file. You can find documentation on using *mongoexport* at https://docs.mongodb.com/database-tools/mongoexport/.

(End of assignment)