CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

## Comp7037 : NoSQL Data Architectures

Assignment 2: **NoSQL Data Architectures** with MongoDB and Neo4j

Complete the assignment in groups of 2. Each member of the group must be familiar with all work produced by the group. Both group members of the group will be interviewed on their work in the labs during week 11 and week 12. No marks will be given for the assignment to any student who has not done an interview with the lecturer on the work.

### 1. Background.

Rachel Allen is one of the most famous Cork-based Irish chefs, well known for her work on television and as a writer. Let's suppose that, after years saving, Rachel has decided to follow one of her dreams: Open her own new restaurant in New York City!

She has been deep in thought about how to make this project successful: *What are New Yorkers taste like? Do her famous recipes fit into the new market, or should they be adapted somehow?*

During her research Rachel has found out a json file with information on restaurants in New York. This will enable Rachel to set up a database with details on ten of thousands of restaurants.

Rachel is hoping the data might contain some insights and answer the following questions:

1. What kind of cuisine do New Yorkers prefer?
2. Which area represents the biggest market opportunity for opening a new restaurant of this kind of cuisine?
3. Who are the biggest competitors in this area?

You are to set up a collection in a Mongodb database. The mongo database must be distributed (or sharded) on a cluster consisting of 15 servers placed on 4 small data centres here in Ireland. The Mongo database collection is to be replicated on a Neo4j database using mongo connector.

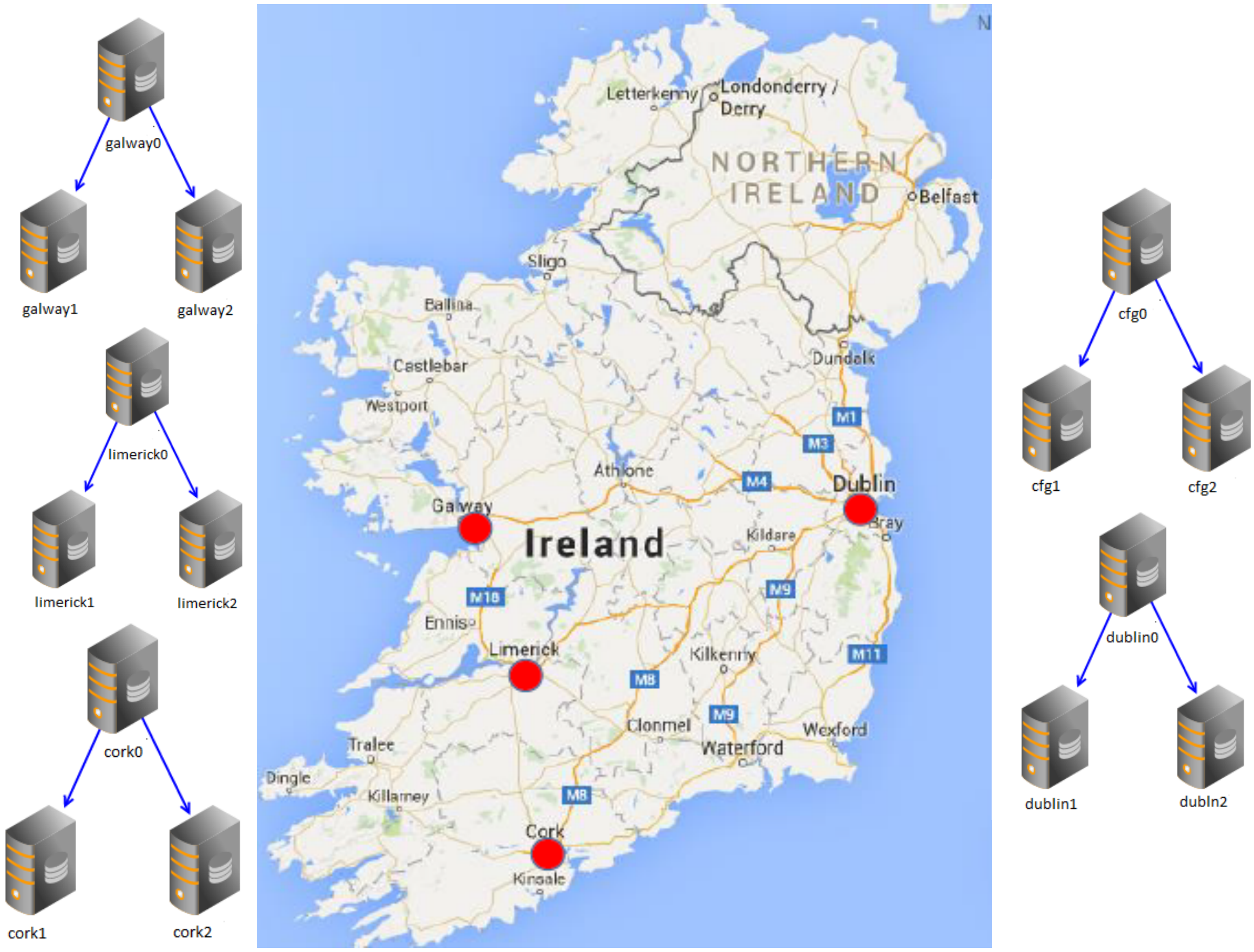
Your goals for this assignment are

1. Set up the database and collection in Mongodb distributed across 4 shards (**use scripts provided however you will need to improve and comment them**).

2. Create a database in Neo4j from the data in mongodb to implement polyglot persistance.
3. Find the answer to Rachel's questions on both databases
4. Write a report to recommend what database configuration Rachel should consider going forward and specify the advantages and disadvantages of your proposal (minimum 2 A4 pages, max 4 pages)

## 2. The MongoDB Cluster.

Your MongoDB cluster will be based on the one we have looked at in the lectures to demonstrate sharding. It consists of 15 nodes (or independent servers), distributed over 4 small data centres: 6 nodes in Dublin and 9 nodes in Cork, Limerick and Galway (3 nodes each). The following picture represents the data centres and their associated nodes.



Each group of 3 nodes { *city0*, *city1*, *city2* } represents a shard of the cluster. The nodes are established as a truly replica set, in which one node acts as the primary node and the other two as secondary nodes (originally, *city0* as primary and *city1* and *city2* as secondary, but this setting is susceptible to be changed during the session).

The group of 3 nodes { cfg0, cfg1, cfg2 } represents the configuration servers. The nodes are established as well as a truly replica set, and they contain the metadata of each database and collection (i.e., which is the primary shard for each unsharded collection, and what is the distribution of the chunks among the shards for each sharded collection).

Finally, the cluster is interfaced to the potential clients via lightweight processes. On the one hand, these processes abstract the data queried from the clients from the cluster complexity for storing it. The processes present the entire cluster as a single logical node, abstracting the actual physical distribution of the data among the shards (the 12 *city** nodes).

On the other hand, these processes provide also access to the metadata (or configuration databases contained in the 3 cfg* nodes). This metadata provides a detailed description of the shards and the concrete data hosted by each of them.

### 3.The Cluster Setup.

The cluster “is simulated”, in the sense that it is built up on top of a single physical machine. A script-based procedure is provided to allow you to build up the cluster on your own physical machine. It consists on the following files (resources folder on blackboard) :

- 1.create_nodes.bat:
- 2.setup_cluster.bat:
- 3.insert_collection.bat:
- 4.shard_collection.bat:
- 5.remove_cluster.bat

Important Notes :-

1. If you are setting up in the lab change the path in the scripts to point to your student folder on the c drive – there is sufficient space for set up there.
2. Use version 3.0 of mongodb for set up of cluster if not already installed download and install on c drive in lab machines

### The MongoDB Collection.

The MongoDB collection of restaurants used as dataset for the assignment is in the zip file for the set up of the cluster in the resources folder on blackboard.

Each document represents 1 restaurant, and provides the following information:

- The name of the restaurant (together with a unique identifier).
- The borough of the city in which the restaurant is placed, together with its complete address (including its zipcode, street, building and coordinates).

- The kind of cuisine offered in the restaurant.
- A list of reviews from customers, including the date, grade and score of each review.

The following JSON object (used as an example) is one of the documents of the collection:

```
{
        "address": {
                "building": "1007",
                "coord": [-73.856077, 40.848447],
                "street": "Morris Park Ave",
                "zipcode": "10462"
        },
        "borough": "Bronx",
        "cuisine": "Bakery",
        "grades": [
                { "date": { "$date": 1393804800000 },  "grade": "A",  "score": 2 },
                { "date": { "$date": 1378857600000 },  "grade": "A",  "score": 6 },
                { "date": { "$date": 1358985600000 },  "grade": "A",  "score": 10 },
                { "date": { "$date": 1322006400000 },  "grade": "A",  "score": 9 },
                { "date": { "$date": 1299715200000 },  "grade": "B",  "score": 14 }
                 ],
        "name": "Morris Park Bake Shop",
        "restaurant_id": "30075445"
}
```

## 4. Neo4j – Polyglot persistance with MongoDB

Mongo-connector provides a mechanism for listening for all update operations and facilitates mirroring those updates to Neo4j using DocManager to interface with MongoDb.

APOC procedures in neo4j provide methods for connecting to and retrieving data to enable equivalent nodes to be created in neo4j

Important Notes :-

1. Cluster configuration is very resource intensive to avoid issues created by this I have given you options to simplify the synchronisation of the databases detailed in points 2 and 3
2. Alernative A

- Use DocManager – running mongodb version 3.4
- Set up single replication set eg Dublin…. Guidelines in hints document posted on blackboard
- Point executable Docmanager to this replication set
- Populate Collection for restaurants
- Do queries on resultant synchronised neo4j database and mongodb collection
- Avoids difficulties dealing with multiple shards and replication sets in cluster
- Simplifies integration as using latest versions of software

3. Alernative B
   - Use APOC procedures – running mongodb version 3.4
   - Set up guidelines in hints document posted on blackboard
   - Run single mongod server, on which you will populate a database with the restaurants collection
   - Connect to mongodb using APOC procedures and populate neo4j database with information returned.
   - Populate Collection for restaurants
   - Do queries on resultant synchronised neo4j database and mongodb collection
   - Avoids difficulties dealing with multiple shards and replication sets in cluster, connecting with single mongo server
   - Simplifies integration as using latest versions of software

4. **Queries for the Assignment.**

**Note : Although not querying a cluster it is an absolute requirement to develop mongo queries as Aggregation Pipeline. The requirement for communicating with a cluster has been removed because of resource constraints however as the complexities of interface are abstracted by mongos, essential to continue as if cluster there to achieve learning goals of assignment.**

As mentioned before, Rachel would like to apply some data analytics on the restaurants collection, as the data might contain some insights answering the following questions:

1. *What kind of cuisine do New Yorkers prefer?*
2. *Which area represents the biggest market opportunity for opening a new restaurant of this kind of cuisine?*
3. *Who are the biggest competitors in this area?*

She will make her final decisions based on the following approach:

*i.* Instead of opening an Irish-based restaurant in which she will directly apply her famous recipes, she will adapt them to whatever style New Yorkers like the most. Thus, she wants to know the kind of cuisine with higher number of restaurants in the city. Whatever this cuisine-style is, she will open a restaurant of this kind.

*ii.* Once the kind of cuisine is fixed, she wants to decide the borough in which she will open the new restaurant. For this reason, she wants to know the ratio (percentage) of restaurants of this kind of cuisine per borough. Her approach would be to pick the borough with smaller ratio (i.e., the one with less competence).

Of course this approach might not be ideal. For example, if Manhattan has a 50% of restaurants of type A and Brooklyn has only a 20%, perhaps is because people in Brooklyn are not that much interested in restaurants of type A. In other words, there is a risk. But, come on, there is going to be a risk whatever the approach being followed. So that's why we are doing data analytics, to get some insights helping she *to maximise the chances* of making good decisions.

*iii.* Once the kind of cuisine and the borough are fixed, she wants to follow the same approach for the zipcode of the borough in which she will open the restaurant. She will only consider the 5 best zipcodes (i.e., the 5 zipcodes of the borough with higher number of restaurants in total). For these 5 zipcodes, she wants to know the ratio (percentage) of restaurants of this kind of cuisine. Her approach would be to pick the zipcode with smaller ratio (i.e., the one with less competence).

*iv.* Finally, once the kind of cuisine, borough and zipcode for the new restaurant are fixed, she wants to know which ones are the 3 biggest competitors of the zipcode. That is, which are the best 3 restaurants of this very same kind of cuisine, placed in this very same borough and zipcode. Her approach would be to visit them all to see how to differentiate her new restaurant from them. To select the three restaurants to visit, she wants to follow the reviews available in the collection. She wants to consider only restaurants with, at least, 4 customer reviews, and then select the 3 with best (higher) average review scores.

## 6. Exercise – MongoDB and Neo4j analysis

The Python file **mongo_data_analysis.py** is to be developed, at template is availabe on blackboard to perform the data analytics answering Rachel's questions in the MongoDB database. It uses the pymongo library to connect a mongo.exe client to the cluster (via the interface provided by one of the mongos.exe processes) and query the restaurants collection. As mentioned, the mongos.exe interface abstracts the complexity of the cluster to the client, so you can think as if the entire cluster was a single logical node.

Complete the functions **(NB. when coding the functions, minimise the amount of data being transferred from the cluster to the Python program using MongoDB aggregation framework)** :

i. **most_popular_cuisine**:
  - Method which receives the name of the test database of the cluster in which the restaurants collection is.
  - The method returns the name of the kind of cuisine with higher number of restaurants in New York and its ratio (percentage).

ii. **ratio_per_borough_and_cuisine**:

- The method receives the name of the test database and the kind of cuisine we are interested in.
- The method returns the name of the borough with smaller percentage of restaurants of this kind of cuisine. It also returns the proper percentage.

iii. **ratio_per_zipcode**:

- The method receives the name of the test database, the kind of cuisine and the borough we are interested in.
- The method returns the name of the zipcode with smaller percentage of restaurants of this kind of cuisine. It also returns the proper percentage.

iv. **best_restaurants**:

- The method receives the name of the test database, the kind of cuisine, borough and zipcode we are interested in.
- The method returns the names of the 3 best restaurants (the ones with, at least, 4 reviews and higher average scores). It also returns the proper average scores.

A file neo4j_data_analysis.cql is to be created. The file should contain the queries required to :-

i. **Find the most_popular_cuisine**:

- returns the name of the kind of cuisine with higher number of restaurants in New York and its ratio (percentage).

ii. **Find the ratio_per_borough_and_cuisine**:

- returns the name of the borough with smaller percentage of restaurants of the kind of cuisine from (i). It also returns the proper percentage.

iii. **Find the ratio_per_zipcode**:

- returns the name of the zipcode with smaller percentage of restaurants of a particular kind of cuisine from (i) and (ii). It also returns the proper percentage.

iv. **Find the best_restaurants**:

- returns the names of the 3 best restaurants (the ones with, at least, 4 reviews and higher average scores). It also returns the proper average scores.

**Important.**

One possible solution in mongodb would be to gather all the documents of the collection and bring them to the Python program (so as to process them within it). **DO NOT** do this, as the amount of data to be transferred represents an unaffordable bottleneck! Instead, use the MongoDB aggregation framework. That is, within the Python program create the pipeline of steps to process the query. Then, connect to the cluster to trigger there the entire pipeline there. In this context, the cluster will only transfer back to the Python program the final list of documents achieved as a result of executing the entire pipeline.

**Marking Scheme and Submission Date.**

- Total marks in assessment 30%.
  - 12% of marks for setup of mongodb cluster with data
  - 24% of marks for  creation of Neo4j graph database with connector
  - 26% of marks for python and mongo queries
  - 26% neo4j queries
  - 12% of marks for report
- Submission: Upload to Blackboard all scripts, code files and the report before Sunday 11pm week 10, each student group will demo their work to the lecturer in the labs in week 11 and week 12.

**NB : STUDENT WORK WILL NOT BE MARKED WITHOUT REVIEWING WITH LECTURER IN LAB**