FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

NOSQL ASSIGNMENT

Database Technologies 2020/2021

Group G:

André Mori - up201700493@edu.fe.up.pt
Diogo Silva - up201405742@edu.fe.up.pt
Jiahao Zhang - up202010263@edu.fe.up.pt
Pasit Khantigul - up202010272@edu.fe.up.pt

Index

Introduction

The main goal of this project is to develop two NoSQL databases, a document database (Mongo) and a graph database (Neo4j), and compare their performance, modeling and easiness with the relational approach (SQL).

Model

For the MongoDB model, the group decided to create two collections: *Facilities* and *Municipalities*. The decision was made based on trying to group all the tables from the relational model into the least amount of collections, so less “joins” would have to be used.

Below are model examples of the Mongo collections:

```
// Facilities
{
    "_id" : NumberInt(602),
    "name" : "RESTAURANTE TÍPICO A TIPÓIA",
    "capacity" : NumberInt(60),
    "roomtype" : "Casa de fado",
    "address" : "RUA DO NORTE 100-102",
    "municipality" : {
        "designation" : "Lisboa",
        "district" : {
            "designation" : "Lisboa",
            "region" : "",
            "_id" : NumberInt(11)
        },
        "region" : {
            "designation" : "Lisboa",
            "nut1" : "Continente",
            "_id" : NumberInt(3)
        },
        "_id" : NumberInt(1106)
    },
    "activities" : [
        "música"
    ]
}

// Municipalities
{
    "_id" : NumberInt(101),
    "designation" : "Águeda",
    "district" : {
        "designation" : "Aveiro",
        "region" : "",
```

```
        "_id" : NumberInt(1)
    },
    "region" : {
        "designation" : "Centro",
        "nut1" : "Continente",
        "_id" : NumberInt(2)
    }
}
```

For the Neo4j model, the group decided to create Nodes for all the tables of the relational model, except for the *Uses* table. The decision was based on creating relationships between the different node types to eliminate the joins from the SQL model. The *Uses* table was just necessary to create relationships between *Facilities* and all of its *Activities*, but not to create its own node.

Below are the nodes and relationships created in Neo4j:

Nodes:

- Activities
- Districts
- Facilities
- Municipalities
- Regions
- Roomtypes

Relationships:

- DISTRICT_REGION
- FACILITY_ACTIVITY
- FACILITY_MUNICIPALITY
- FACILITY_ROOMTYPE
- MUNICIPALITY_DISTRICT
- MUNICIPALITY_REGION

Populate

To populate the Mongo database, the group chose to create a Python script. The script uses JSON data exported from each table of the relational database on the Oracle SQL developer and creates/populates collections as defined in the Model section.

To populate the Neo4j database, the group exported the tables from the relational model as .CSV files and added them to the project import folder. After that, a script is run to populate the nodes by reading the .CSV files.

More instructions on how to create and populate the databases are available in the **README.md** file delivered.

Questions

Question a)

Which are the facilities where the room type description contains 'touros' and have 'teatro' as one of their activities? Show the id, name, description and activity

Mongo

Query

```
db.facilities.find(
    {
        "roomtype": { $regex: "touros" },
        "activities": "teatro"
    },
    {
        "_id": 1,
        "name": 1,
        "roomtype": 1,
        "activities": 1
    }
);
```

Result

```
{
    "_id" : NumberInt(916),
    "name" : "COLISEU JOSÉ RONDÃO DE ALMEIDA-EX PRAÇA DE TOIROS",
    "roomtype" : "Praça de touros multiusos",
    "activities" : [
        "dança",
        "música",
        "tauromaquia",
        "teatro"
    ]
}
{
    "_id" : NumberInt(940),
    "name" : "ARENA DE ÉVORA - EX PRAÇA DE TOIROS",
    "roomtype" : "Praça de touros multiusos",
    "activities" : [
        "dança",
        "música",
```

```
        "tauromaquia",
        "teatro"
    ]
}
{
    "_id" : NumberInt(957),
    "name" : "COLISEU DE REDONDO - EX PRAÇA DE TOIROS",
    "roomtype" : "Praça de touros multiusos",
    "activities" : [
        "dança",
        "música",
        "tauromaquia",
        "teatro"
    ]
}
```

Neo4j

Query

```
MATCH (f:Facilities)-[:FACILITY_ROOMTYPE]-(rt:Roomtypes)
MATCH (f)-[:FACILITY_ACTIVITY]-(a:Activities)
WHERE rt.DESCRIPTION CONTAINS 'touros' AND a.ACTIVITY = 'teatro'
RETURN f.ID as ID, f.NAME as NAME, rt.DESCRIPTION as DESCRIPTION, a.ACTIVITY as
ACTIVITY;
```

Result

ID	NAME	DESCRIPTION	ACTIVITY
940	"COLISEU DE REDONDO - EX PRAÇA DE TOIROS"	"Praça de touros multiusos"	"teatro"
957	"ARENA DE ÉVORA - EX PRAÇA DE TOIROS”	"Praça de touros multiusos"	"teatro"
916	"COLISEU JOSÉ RONDÃO DE ALMEIDA-EX PRAÇA DE TOIROS"	"Praça de touros multiusos"	"teatro"

SQL

Query

```
SELECT id, name, description, activity
FROM facilities NATURAL JOIN roomtypes NATURAL JOIN xuses NATURAL JOIN
activities
WHERE description LIKE '%touros%' AND activity = 'teatro';
```

Result

ID	NAME	DESCRIPTION	ACTIVITY
916	COLISEU JOSÉ RONDÃO DE ALMEIDA–EX PRAÇA DE TOIROS	Praça de touros multiusos	teatro
940	ARENA DE ÉVORA – EX PRAÇA DE TOIROS	Praça de touros multiusos	teatro
957	COLISEU DE REDONDO – EX PRAÇA DE TOIROS	Praça de touros multiusos	teatro

Question b)

How many facilities with 'touros' in the room type description are there in each region?

Mongo

Query

```
db.facilities.aggregate([
    {
        $match:
        {
            "roomtype": { $regex: "touros" }
        }
    },
    {
        $group:
        {
            "_id": "$municipality.region.designation",
            "n_facilities": { $sum: 1 }
        }
    }
]);
```

Result


```
{
    "_id" : "Algarve",
    "n_facilities" : 1.0
}
{
    "_id" : "Lisboa",
    "n_facilities" : 6.0
}
{
    "_id" : "Alentejo",
    "n_facilities" : 43.0
}
{
    "_id" : "Norte",
    "n_facilities" : 3.0
}
{
    "_id" : "Centro",
    "n_facilities" : 11.0
}
```


Neo4j

Query

```
MATCH (f:Facilities)-[:FACILITY_MUNICIPALITY]->(m:Municipalities)
MATCH (m)-[:MUNICIPALITY_REGION]->(r:Regions)
MATCH (f)-[:FACILITY_ROOMTYPE]->(rt:Roomtypes)
WHERE rt.DESCRIPTION CONTAINS 'touros'
RETURN r.DESIGNATION AS DESIGNATION,count(r.DESIGNATION) AS N_FACILITIES;
```

Result

DESIGNATION	N_FACILITIES
"Alentejo"	43
"Lisboa"	6
"Centro”	11
"Norte"	3
"Algarve"	1

SQL

Query

```sql
SELECT regions.designation, count(*)
FROM facilities NATURAL JOIN roomtypes INNER JOIN municipalities
ON municipalities.cod = facilities.municipality
INNER JOIN regions ON municipalities.region = regions.cod
WHERE description  LIKE '%touros%'
GROUP BY regions.designation;
```

Result

DESIGNATION	COUNT(*)
Lisboa	6
Norte	3
Centro	11
Alentejo	43
Algarve	1

Question c)

How many municipalities do not have any facility with an activity of 'cinema'?

Mongo

Query

```javascript
db.municipalities.count() - db.facilities.aggregate(
    {
        $match: {
            "activities": { $in: ["cinema"] }
        }
    },
    {
        $group: {
            "_id": "$municipality.designation",
            "n_facilities": { $sum: 1 }
        }
    }
).toArray().length;
```

Result

```
100
```

Neo4j

Query

```
MATCH (f:Facilities)-[:FACILITY_ACTIVITY]->(a:Activities)
WHERE a.ACTIVITY = 'cinema'
MATCH (f)-[:FACILITY_MUNICIPALITY]->(m:Municipalities)
WITH COUNT(DISTINCT m.COD) AS municipalitiesWithCinema
MATCH (totalMunicipalities:Municipalities)
RETURN COUNT(totalMunicipalities) - municipalitiesWithCinema AS Total;
```

Result

Total
100

SQL

Query

```
SELECT count(*) FROM
municipalities
WHERE cod NOT IN
(SELECT municipality
FROM facilities NATURAL JOIN uses NATURAL JOIN activities
INNER JOIN
municipalities ON municipalities.cod = facilities.municipality
WHERE activity = 'cinema');
```

Result

COUNT(*)
100

Question d)

Which is the municipality with more facilities engaged in each of the six kinds of activities? Show the activity, the municipality name and the corresponding number of facilities.

Mongo

Query

```
db.facilities.aggregate([
    {
        $unwind: "$activities"
    },
    {
        $group:
        {
            "_id": {municipality: "$municipality.designation", activity:
"$activities"},
            "total": { $sum: 1}
        }
    },
    {
        $sort:
        { "total": -1 }
    },
    {
        $group:
        {
            "_id": "$_id.activity",
            "municipality": {"$first": "$_id.municipality"},
            "n_facilities": {$max: "$total"}
        }
    },
    {
        $sort: { "_id": 1 }
    },

]);
```

Result


```
{
    "_id" : "cinema",
    "municipality" : "Lisboa",
    "n_facilities" : 96.0
}
{
    "_id" : "circo",
    "municipality" : "Lisboa",
    "n_facilities" : 2.0
}
{
    "_id" : "dança",
    "municipality" : "Lisboa",
    "n_facilities" : 47.0
}
{
    "_id" : "música",
    "municipality" : "Lisboa",
    "n_facilities" : 77.0
}
{
    "_id" : "tauromaquia",
    "municipality" : "Moura",
    "n_facilities" : 4.0
}
{
    "_id" : "teatro",
    "municipality" : "Lisboa",
    "n_facilities" : 66.0
}
```


Neo4j

Query

```
MATCH (f:Facilities)-[:FACILITY_ACTIVITY]->(a:Activities)
MATCH (f)-[:FACILITY_MUNICIPALITY]->(m:Municipalities)
WITH m.DESIGNATION AS DESIGNATION, a.ACTIVITY AS ACTIVITY,  COUNT(a) AS TOTAL
WITH COLLECT({municipality: DESIGNATION, activity: ACTIVITY, total: TOTAL}) AS
ROWS, MAX(TOTAL) AS MAX, ACTIVITY AS activity
UNWIND [ROW IN ROWS WHERE ROW.activity = activity AND ROW.total = MAX] AS ROW
RETURN  ROW.activity AS ACTIVITY, ROW.municipality AS MUNICIPALITY, ROW.total AS
N_FACILITIES;
```

Result

ACTIVITY	MUNICIPALITY	N_FACILITIES
"dança"	"Lisboa"	47
"cinema"	"Lisboa"	96
"teatro"	"Lisboa"	66
"música"	"Lisboa"	77
"tauromaquia"	"Moura"	4
"circo"	"Lisboa"	2

SQL

Query

```
SELECT q2.designation, q1.activity, q1.max_facilities
FROM (SELECT activity, max(tot_facilities) as max_facilities
      FROM (SELECT municipality, activity, count(*) as
tot_facilities
      FROM facilities NATURAL JOIN uses NATURAL JOIN activities
      GROUP BY municipality, activity)
GROUP BY activity) q1 LEFT JOIN
     (SELECT designation, activity, count(*) as tot_facilities
      FROM municipalities INNER JOIN facilities NATURAL JOIN uses
NATURAL JOIN activities
      ON municipalities.cod = facilities.municipality
      GROUP BY designation, activity) q2 ON q2.activity = q1.activity AND
q2.tot_facilities = q1.max_facilities;
```

Result

DESIGNATION	ACTIVITY	FACILITIES
Lisboa	circo	2
Lisboa	dança	47
Moura	tauromaquia	4
Lisboa	música	77
Lisboa	teatro	66
Lisboa	cinema	96

Question e)

Which are the codes and designations of the districts with facilities in all the municipalities?

Mongo

Query

```
db.municipalities.aggregate([
    {
        $lookup:
        {
            from: 'facilities',
            localField: '_id',
            foreignField: 'municipality._id',
            as: 'facilities'
        }
    },
    {
        $group:
        {
            "_id": { _id: "$district._id", designation: "$district.designation"
},
            "municipalities": { $push: { "nome": "$designation",
"hasFacilities": {
                $gt: [{ $size: "$facilities" }, 0]
            } } }
        },
    },
    {
        $match: { "municipalities": { "$not": { "$elemMatch": { "hasFacilities":
false } } } }
    },
    {
        $group: { "_id": "$_id._id", "designation": { $first: "$_id.designation"
} }
    }
]);
```

Result

```
{
    "_id" : NumberInt(11),
    "designation" : "Lisboa"
}
{
    "_id" : NumberInt(7),
    "designation" : "Évora"
}
{
    "_id" : NumberInt(12),
    "designation" : "Portalegre"
}
{
    "_id" : NumberInt(15),
    "designation" : "Setúbal"
}
```

Neo4j

Query

```
MATCH (m)-[:MUNICIPALITY_DISTRICT]-(d:Districts)
OPTIONAL MATCH (f)-[:FACILITY_MUNICIPALITY]->(m)
WITH d as District, m.COD as M, f.ID as Facility
WHERE Facility is NULL
MATCH (d:Districts)
WITH COLLECT(DISTINCT District) AS withoutAllFacilities, COLLECT(DISTINCT d) AS
allDistricts
WITH [n IN allDistricts WHERE NOT n IN withoutAllFacilities] AS
withAllFacilities
UNWIND withAllFacilities as resDistricts
RETURN resDistricts.COD AS COD, resDistricts.DESIGNATION AS DESIGNATION;
```

Result

COD	DESIGNATION
7	Évora
11	Lisboa
12	Portalegre
15	Setúbal

SQL

Query

```
SELECT cod, designation
FROM districts WHERE cod NOT IN
(
SELECT districts.cod
FROM municipalities INNER JOIN districts ON municipalities.district =
districts.cod
LEFT OUTER JOIN facilities on facilities.municipality = municipalities.cod
WHERE id is null
);
```

Result

COD	DESIGNATION
15	Setúbal
7	Évora
11	Lisboa
12	Portalegre

Question f)

Ask the database a query you think is interesting:
"For each district, find the average capacity of its facilities with a precision of 2 decimal numbers."

Mongo

Query

```
db.facilities.aggregate([
{
    $group:
    {
        "_id": "$municipality.district._id",
        "designation": { $first: "$municipality.district.designation"},
        "capacity": { $avg: "$capacity"}
    }
},
{
    $addFields:
    {
        "avg": {$divide:[{$subtract:[{$multiply:['$capacity',100]},
               {$mod:[{$multiply:['$capacity',100]}, 1]}]},100]}
    }
},
{
    $group:
    {
        "_id": "$_id",
        "designation": { $first: "$designation"},
        "avgCapacityPerDistrict": {$first: "$avg"}
    }
},
{
    $sort: { "_id": 1 }
},
]);
```

Result

```
{ "_id" : NumberInt(1), "designation" : "Aveiro", "avgCapacityPerDistrict" : 332.34 }
```

```
{ "_id" : NumberInt(2), "designation" : "Beja", "avgCapacityPerDistrict" : 794.0 }
{ "_id" : NumberInt(3), "designation" : "Braga", "avgCapacityPerDistrict" : 552.01 }
{ "_id" : NumberInt(4), "designation" : "Bragança", "avgCapacityPerDistrict" : 688.72 }
{ "_id" : NumberInt(5), "designation" : "Castelo Branco", "avgCapacityPerDistrict" : 316.09 }
{ "_id" : NumberInt(6), "designation" : "Coimbra", "avgCapacityPerDistrict" : 383.25 }
{ "_id" : NumberInt(7), "designation" : "Évora", "avgCapacityPerDistrict" : 887.64 }
{ "_id" : NumberInt(8), "designation" : "Faro", "avgCapacityPerDistrict" : 451.38 }
{ "_id" : NumberInt(9), "designation" : "Guarda", "avgCapacityPerDistrict" : 431.4 }
{ "_id" : NumberInt(10), "designation" : "Leiria", "avgCapacityPerDistrict" : 417.22 }
{ "_id" : NumberInt(11), "designation" : "Lisboa", "avgCapacityPerDistrict" : 397.72 }
{ "_id" : NumberInt(12), "designation" : "Portalegre", "avgCapacityPerDistrict" : 1019.97 }
{ "_id" : NumberInt(13), "designation" : "Porto", "avgCapacityPerDistrict" : 359.38 }
{ "_id" : NumberInt(14), "designation" : "Santarém", "avgCapacityPerDistrict" : 1080.0 }
{ "_id" : NumberInt(15), "designation" : "Setúbal", "avgCapacityPerDistrict" : 528.69 }
{ "_id" : NumberInt(16), "designation" : "Viana do Castelo", "avgCapacityPerDistrict" : 396.13 }
{ "_id" : NumberInt(17), "designation" : "Vila Real", "avgCapacityPerDistrict" : 246.85 }
{ "_id" : NumberInt(18), "designation" : "Viseu", "avgCapacityPerDistrict" : 274.76 }
```

Neo4j

Query

```
MATCH (f:Facilities)-[:FACILITY_MUNICIPALITY]->(m:Municipalities)
MATCH (m)-[:MUNICIPALITY_DISTRICT]->(d:Districts)
RETURN d.COD AS COD, d.DESIGNATION as DESIGNATION,
ROUND(100*AVG(f.CAPACITY))/100 AS avgCapacityPerDistrict
ORDER BY d.COD;
```

Result

COD	DESIGNATION	avgCapacityPerDistrict
1	Aveiro	332.34
2	Beja	794.00
3	Braga	552.02
4	Bragança	688.73
5	Castelo Branco	316.10
6	Coimbra	383.26
7	Évora	887.65
8	Faro	451.39
9	Guarda	431.41

10	Leiria	417.23
11	Lisboa	397.73
12	Portalegre	1019.97
13	Porto	359.38
14	Santarém	1080.00
15	Setúbal	528.69
16	Viana do Castelo	396.13
17	Vila Real	246.86
18	Viseu	274.76

SQL

Query

```
SELECT d.cod,d.designation, ROUND(AVG(f.capacity),2) AS Avg_Cap_Per_District
FROM districts d, municipalities m, facilities f
WHERE m.district = d.cod AND f.municipality = m.cod
GROUP BY d.cod,d.designation;
```

Result

COD	DESIGNATION	AVG_CAP_PER_DISTRICT
1	Aveiro	332.34
2	Beja	794
3	Braga	552.02
4	Bragança	688.73
5	Castelo Branco	316.1
6	Coimbra	383.26
7	Évora	887.65
8	Faro	451.39
9	Guarda	431.41
10	Leiria	417.23
11	Lisboa	397.73
12	Portalegre	1019.97
13	Porto	359.38
14	Santarém	1080
15	Setúbal	528.69
16	Viana do Castelo	396.13
17	Vila Real	246.86
18	Viseu	274.76

Comparisons

During the development of the queries, we see that although SQL gives a lot of possibilities, Neo4j's model can be similar to the relational database, but with its relationships it removes the needs to perform JOIN operations (which requires a longer time) and allows almost direct access from one node to another. Besides that, it's syntax is not so complicated and it also provides a visual representation, which is very helpful to check how the nodes are interacting with each other.

Although MongoDB was easy to populate since it doesn't require defining a specific type, it's queries are slightly harder to implement since it has its own syntax and it also requires a very different modeling when converting it from the relational model, which required a lot of thinking and time to do so.

Execution times:

Question	MongoDB	Neo4j	SQL
A)	15 ms	5 ms	35 ms
B)	16 ms	5 ms	30 ms
C)	55 ms	10 ms	26 ms
D)	18 ms	33 ms	37 ms
E)	187 ms	37 ms	23 ms
F)	18 ms	12 ms	31 ms

We could see that for most of the queries analysed for this assignment, the Neo4j environment gives us better performance regarding the execution time. Mongo would come in second place, being faster than the SQL model.

For harder queries (for example the question E which has inside 2 queries), the execution time would be slightly higher in Neo4j and Mongo than in the SQL environment, but the general performance of them is better than the SQL.

For question E) in Mongo, the time is way higher since it needs to execute a query and also count the number of rows in the Municipalities collection, which takes a longer time.

Sizes:

Mongo	Neo4j	SQL
868 352 B	1 413 939 B	111 547 B

Note: the Neo4j size was obtained measuring the databases/neo4j folder, which contains all the .db files.

We can conclude that the SQL model occupies way less space than the other databases. This happens since the NoSQL model has duplicate data stored and the Neo4j has the nodes and relationships stored which also requires a large amount of data.

Conclusions

The group successfully modeled, populated and developed the required queries for all databases. With the development of the project, the group also managed to learn the differences between different kinds of databases and know more about their advantages and disadvantages. In conclusion, SQL relational database is still the smallest one in terms of space, but this comes with a cost of slower performance than in the other models, mainly because it requires the JOIN operations.