TBDA

Cultural Facilities

Group members:

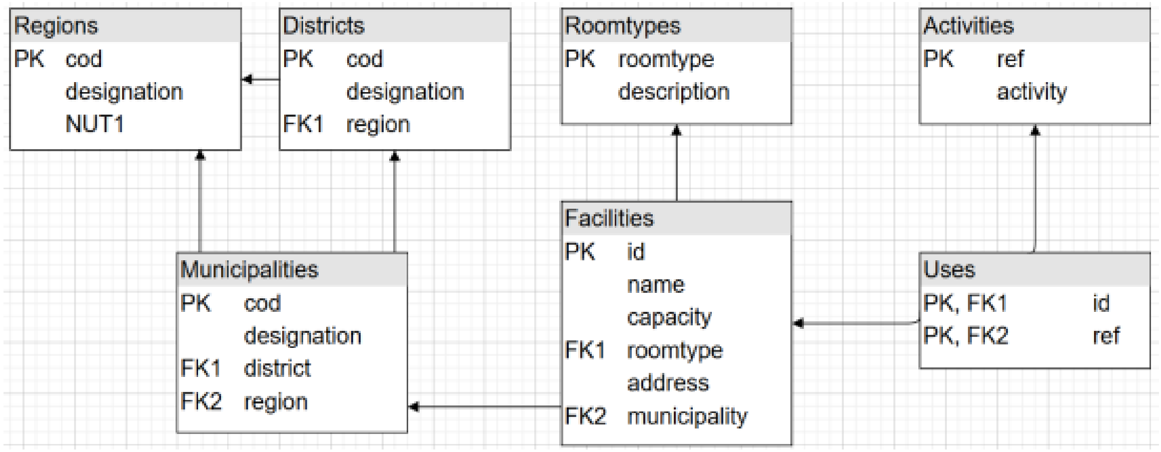
Juliane Marubayashi (up201800175)
Ricardo Carvalho (up201806791)
Rodrigo Reis (up201806534)

Faculdade de Engenharia do Porto
Jun, 12th of 2022, Porto

Index

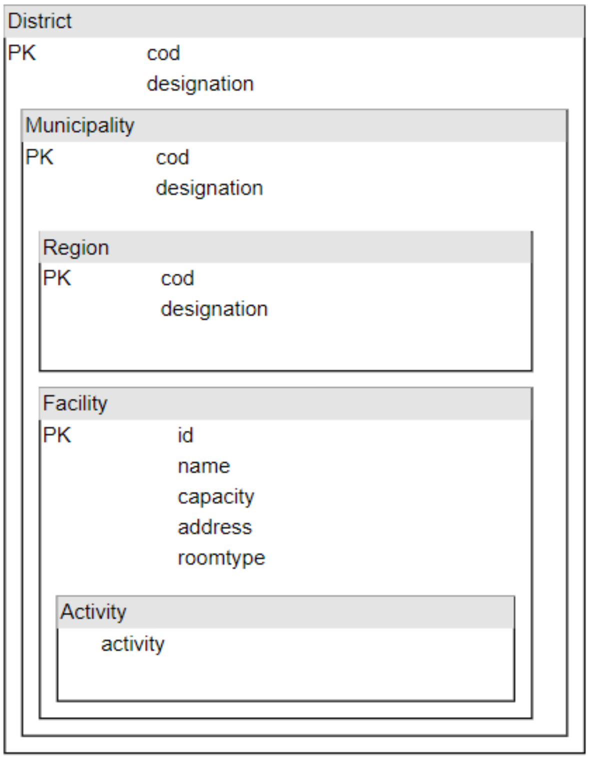
1) Context

This project regards the databases for the Cultural Facilities case. The facilities are the local where cultural activities takes place and these are located in a municipality. In each place, there may be activities of more than one category.



2) NoSQL model (MongoDB)

The group chose to make a unique collection (cultural_facilities), in which each document represents a district (total of 20 documents: 18 continent + 2 islands). We think that this format speeds up the queries that we'll do and handles missing information, a district may not have a region, but a municipality will always have a region and a district.



```
{
    "_id": NumberInt(15),
    "designation": "Setúbal",
    "municipalities": [
        {
            "_id": NumberInt(1501),
            "designation": "Alcácer do Sal",
            "region": {
                "_id": NumberInt(4),
                "designation": "Alentejo",
                "nut1": "Continente"
            },
            "facilities": [
                {
                    "_id": NumberInt(887),
                    "name": "PRAÇA DE TOIROS JOÃO BRANCO NUNCIO - ALCÁCER SAL",
                    "capacity": NumberInt(3791),
                    "roomtype": "Praça de touros",
                    "address": "ALCÁCER DO SAL",
                    "activities": [
                        "tauromaquia"
                    ]
                },
                {
                    "_id": NumberInt(888),
                    "name": "AUDITÓRIO MUNICIPAL DE ALCÁCER DO SAL",
                    "capacity": NumberInt(204),
                    "roomtype": "Auditório",
                    "address": "RUA MANUEL MARIA BARBOSA DU BOCAGE - BAIRRO DO
MORGADINHO",
                    "activities": [
                        "cinema",
                        "dança",
                        "música",
                        "teatro"
                    ]
                }
            ]
        }
    ]
}
```

3) NoSQL migration

To migrate the tables with the NoSQL model in mind, firstly we have copied the tables from the user GTD8.

```
create table activities as select * from gtd8.activities;
create table districts as select * from gtd8.districts;
create table facilities as select * from gtd8.facilities;
create table municipalities as select * from gtd8.municipalities;
create table regions as select * from gtd8.regions;
create table roomtypes as select * from gtd8.roomtypes;
create table uses as select * from gtd8.uses;
```

After copying the tables, we have created a SQL script which translates the SQL database into a `JSON` array according to the NoSQL model.

```
-- GET ACTIVITES
create or replace function get_activities(facility_id varchar2) return clob is
    jo clob;
begin
    select json_arrayagg(a.activity returning clob) into jo
    from facilities f, activities a, uses u
    where f.id = facility_id and u.id = facility_id and a.ref = u.ref;
    return jo;
end get_activities;
/
-- GET FACILITY ROOM TYPE
create or replace function get_roomtype(roomtype_id varchar2) return clob is
    c clob;
begin
    select r.description into c
    from roomtypes r
    where r.roomtype = roomtype_id;
    return c;
end get_roomtype;

/
-- GET FACITILIES
create or replace function get_facilities(municipality_id varchar2) return clob is
    jo clob;
begin
    select json_arrayagg(
        json_object(
            '_id'           value f.id,
            'name'          value f.name,
            'capacity'      value f.capacity,
            'roomtype'      value get_roomtype(f.roomtype),
            'address'       value f.address,
            'activities'    value get_activities(f.id)
        format json returning clob)
    format json returning clob) into jo
    from facilities f
    where f.municipality = municipality_id;
    return jo;
end get_facilities;
/

-- GET REGION OF A MUNICIPALITY
create or replace function get_region(region_id varchar2) return clob is
    jo clob;
begin
    select json_object(
        '_id'           value r.cod,
        'designation'   value r.designation,
        'nut1'          value r.nut1
        returning clob) into jo
    from regions r
    where region_id = r.cod;
    return jo;
end get_region;
/


-- GET THE MUNICIPALITIES
create or replace function get_municipalities(district_id varchar2) return clob
is
    jo clob;    -- store the result
begin
    select json_arrayagg(json_object(
        '_id'           value m.cod,
        'designation'   value m.designation,
        'region'        value get_region(m.region) format json,
        'facilities'    value get_facilities(m.cod) format json
        returning clob) format json returning clob) into jo
    from municipalities m
    join districts d on d.cod = m.district
    where d.cod = district_id;
    return jo;
end get_municipalities;

/
select json_arrayagg(
    json_object(
        '_id'               value cod,
        'designation'       value designation,
        'municipalities'    value get_municipalities(cod) format json returning
clob)
    format json returning clob)
from districts;
```

Notice that the code is mainly composed by functions that follows an hierarchy of calls:

- The last query in the code above, describes the districts and calls the `get_municipalities(cod)` to add the municipalities sub-documents.
- The same logic applies to the function `get_municipalities(cod)`, it describes the municipalities and adds sub-documents that describe the region and the facilities.
- The other functions follow the same pattern.

Still is valid to mention that it was necessary to return a `clob` type from the `json_object` and `json_arrayagg`. Since the string returned by the JSON might not be small enough to fit into a `VARCHAR2` type, it is necessary to use clob to store large strings.

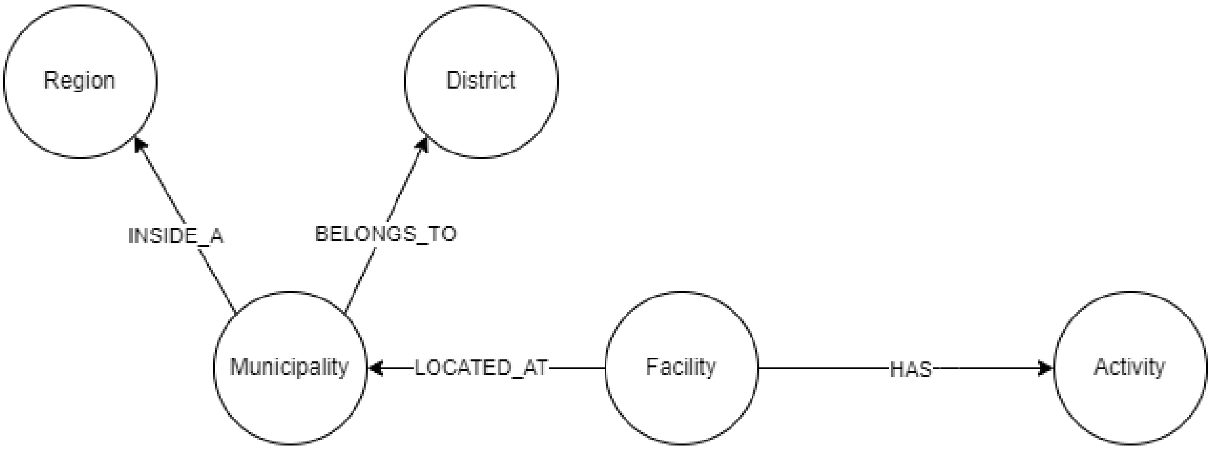
The result of the query above is returned and stored in a document. To populate the database with the recently created document, a the following piece of code in javascript was created:

```
var MongoClient = require('mongodb').MongoClient;
const data = require("./data/data.json");   // json data.
const databaseName = "tbda";
const collectionName = "cultural_facilities";
const url =
"mongodb://tbda:grupoa@vdbase.inesctec.pt:27017/?authSource=admin&readPreference
=primary&appname=MongoDB%20Compass&ssl=false";

MongoClient.connect(url, function (err, db) {
  if (err) throw err;
  const dbo = db.db(databaseName);
  createCollection(dbo);
  cleanCollection(dbo);
  populateCollection(dbo);
  db.close();
});

const createCollection = (dbo) => {
  dbo.createCollection(collectionName, function (err, res) {
    if (err) console.log("Collection already created!");
    console.log("Collection created!");
  });
}
const cleanCollection = (dbo) => {
  dbo.collection(collectionName).remove({});
}

const populateCollection = (dbo) => {
  dbo.collection(collectionName).insertMany(data, (err, res) => {
    if (err) throw err;
    console.log("Database populated!");
  });
}
```

The script is responsible for getting the result from the query and storing it in a variable called `data`, after that it connects to the mongoDB database, creates the collection if it still doesn't exist, clean it and populate it with the content of data.

4) Neo4J model

The Neo4J model represents the relation between data in a simplified way:



The `Roomtypes` was merged into `Facility`, since there's no advantage in creating a connection between these two types, since the amount of data stored would be almost the same and also would slow down the query time. Let's review the affirmation in concrete terms:

- The average size of a `Roomtype` description is 19B, according to the SQLDeveloper statistics;
- According to the Neo4J official website [1], the size that a relationship occupies is 34B.

From this analysis, we have concluded that storing the `Roomtype` inside the Facility is beneficial in terms of space and speed.

Although the relationship between `Facility` and `Activity` is many to many, we've chosen to create just a connection `Facility->Activity`, since queries of types "which activities…" (`Activity->Facility`) are non frequent. The same applies to the relationship `Region->Districts.`

5) Neo4J migration

Firstly, it was necessary to export the data from the SQL database to CSV. This can be done by using the SQLDeveloper interface.

After that created the following *cypher* to populate the database. This script is splitted in two parts:

- The first one creates the nodes: regions, districts, municipalities, facilities, roomtype and activities.
- The second part creates the relationships and deletes the unnecessary nodes (i.e roomtype).

```
// POPULATE ==================================================
// Regions

load csv with headers from 'file:///regions.csv' as line
create (region: Region {
    cod: toInteger(line.COD),
    designation: line.DESIGNATION,
    nut1: line.NUT1})
return region;

// Districts

load csv with headers from 'file:///districts.csv' as line
create (d: District {
    cod: toInteger(line.COD),
    designation: line.DESIGNATION,
    region: toInteger(line.REGION)
})
return d;

// Municipalities

load csv with headers from 'file:///municipalities.csv' as line
create (m: Municipality {
    cod: toInteger(line.COD),
    designation: line.DESIGNATION,
    region: toInteger(line.REGION),
    district: toInteger(line.DISTRICT)
})
return m;

// Facilities

load csv with headers from 'file:///facilities.csv' as line
create (f: Facility{
    id:line.ID,
    name: line.NAME,
    capacity: toInteger(line.CAPACITY),
    roomtype: line.ROOMTYPE,
    address: line.ADDRESS,
    municipality: toInteger(line.MUNICIPALITY)
})
return f;

// Roomtype
```

```
load csv with headers from 'file:///roomtypes.csv' as line
create (r: Roomtype{
    roomtype: line.ROOMTYPE,
    description: line.DESCRIPTION
})
return r;

// Activities

load csv with headers from 'file:///activities.csv' as line
create (a: Activity{
    ref: line.REF,
    activity: line.ACTIVITY
})
return a;


// RELATIONS =======================================================

// Acitvity-USES->Facility ----

load csv with headers from 'file:///uses.csv' as line
match (a: Activity {ref: line.REF})
match (f: Facility {id: line.ID})
// merge (a)-[:USES]->(f);
merge (f)-[:HAS]->(a);

// Add roomtype to facilities directly ----

MATCH (r:Roomtype), (f:Facility {roomtype:r.roomtype})
set f.roomtype=r.description;

// Deleting roomtype nodes
match (a:Roomtype) delete a;


// Facility-LOCATED_AT->Municipality

match (f:Facility)
match (m:Municipality {cod: f.municipality})
merge (f)-[:LOCATED_AT]->(m);

// Municipality-BELONGS_TO->District
match (m: Municipality)
match (d: District {cod: m.district})
merge (m)-[:BELONGS_TO]->(d);

// Municipality-INSIDE_A->Region
match (m: Municipality)
match (r:Region {cod: m.region})
```

```
merge (m)-[:INSIDE_A]->(r);
```

6) Queries

6.1) Query 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.

6.1.1) Query - mongoDB

```
db.cultural_facilities.aggregate(
    {"$unwind": "$municipalities"},
    {"$unwind": "$municipalities.facilities"},
    {"$match":{"municipalities.facilities.roomtype":/touros/,
    "municipalities.facilities.activities":"teatro"}},
    {"$project":{
        "FacilityId": "$municipalities.facilities._id",
        "FacilityName": "$municipalities.facilities.name",
        "Activities": "$municipalities.facilities.activities",
        "Roomtype": "$municipalities.facilities.roomtype",
        "_id":0}})
```

6.1.2) Result - mongoDB

```
{ FacilityId: 940,
  FacilityName: 'ARENA DE ÉVORA - EX PRAÇA DE TOIROS',
  Activities: [ 'dança', 'música', 'tauromaquia', 'teatro' ],
  Roomtype: 'Praça de touros multiusos' }
{ FacilityId: 957,
  FacilityName: 'COLISEU DE REDONDO - EX PRAÇA DE TOIROS',
  Activities: [ 'dança', 'música', 'tauromaquia', 'teatro' ],
  Roomtype: 'Praça de touros multiusos' }
{ FacilityId: 916,
  FacilityName: 'COLISEU JOSÉ RONDÃO DE ALMEIDA-EX PRAÇA DE TOIROS',
  Activities: [ 'dança', 'música', 'tauromaquia', 'teatro' ],
  Roomtype: 'Praça de touros multiusos' }
```

6.1.3) Query - Neo4J

```
match(f:Facility)-[:HAS]->(a:Activity)
where f.roomtype contains 'touros' and a.activity = 'teatro'
```

```
return f.id, f.name, f.roomtype as description, a.activity;
```

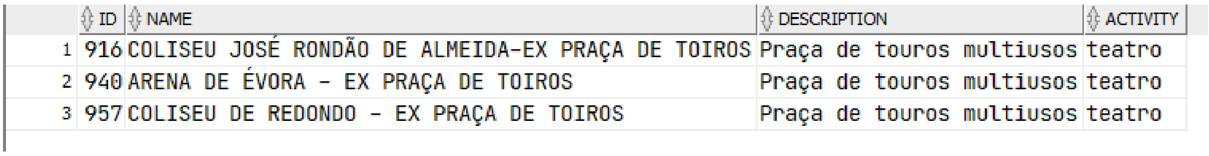
6.1.4) Result Neo4J

```
╒════════╤══════════════════════════════════════╤═══════════════════════════════
═══╤════════════════╕
│"f.id"│"f.name"                     │"description"              │"a.activity"│
╞════════╪══════════════════════════════════════╪═══════════════════════════════
═══╪════════════════╡
│"940" │"ARENA DE ÉVORA - EX PRAÇA DE│"Praça de touros multiusos"│"teatro"    │
│      │ TOIROS"                     │                           │            │
├────────┼──────────────────────────────────────┼───────────────────────────────
───┼────────────────┤
│"916" │"COLISEU JOSÉ RONDÃO DE ALMEI│"Praça de touros multiusos"│"teatro"    │
│      │DA-EX PRAÇA DE TOIROS"       │                           │            │
├────────┼──────────────────────────────────────┼───────────────────────────────
───┼────────────────┤
│"957" │"COLISEU DE REDONDO - EX PRAÇ│"Praça de touros multiusos"│"teatro"    │
│      │A DE TOIROS"                 │                           │            │
└────────┴──────────────────────────────────────┴───────────────────────────────
───┴────────────────┘
```

6.1.5) Query - SQL

```
select f.id, f.name, r.description, a.activity
from facilities f
inner join roomtypes r on f.roomtype = r.roomtype
inner join uses u on u.id = f.id
inner join activities a on a.ref = u.ref
where r.description like '%touros%' and a.activity='teatro';
```

6.1.6) Result SQL



	ID	NAME	DESCRIPTION	ACTIVITY
1	916	COLISEU JOSÉ RONDÃO DE ALMEIDA-EX PRAÇA DE TOIROS	Praça de touros multiusos	teatro
2	940	ARENA DE ÉVORA - EX PRAÇA DE TOIROS	Praça de touros multiusos	teatro
3	957	COLISEU DE REDONDO - EX PRAÇA DE TOIROS	Praça de touros multiusos	teatro

6.1.7) Analysis

The query times are:

- **MongoDB : 0.068 sec**
- **Neo4J: 0.025 sec**

- **SQL: 0.09 sec**

6.2) Query b

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

6.2.1) Query - mongoDB

```
db.cultural_facilities.aggregate(
    {$unwind: "$municipalities"},
    {$unwind: "$municipalities.facilities"},
    {$match:{"municipalities.facilities.roomtype":/touros/}},
    {$group : {_id:"$municipalities.region", count:{$sum:1}}},
    {$project: {
        _id: 0,
        "Region": "$_id.designation",
        "NumFaciltes": "$count"
    }})
```

6.2.2) Result - mongoDB

```
{ Region: 'Lisboa', NumFaciltes: 6 }
{ Region: 'Norte', NumFaciltes: 3 }
{ Region: 'Alentejo', NumFaciltes: 43 }
{ Region: 'Centro', NumFaciltes: 11 }
{ Region: 'Algarve', NumFaciltes: 1 }
```

6.2.3) Query - Neo4J

```
match(f:Facility)-[:LOCATED_AT]->(m:Municipality)-[:INSIDE_A]->(r:Region)
where f.roomtype contains "touro"
return r.designation as Region, count(f) as Num_of_facilities;
```

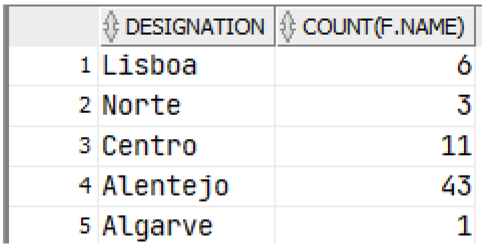
6.2.4) Result Neo4J

```
╒══════════╤═══════════════════╕
│"Region"  │"Num_of_facilities"│
╞══════════╪═══════════════════╡
│"Centro"  │11                 │
├──────────┼───────────────────┤
│"Lisboa"  │6                  │
├──────────┼───────────────────┤
│"Alentejo"│43                 │
├──────────┼───────────────────┤
│"Algarve" │1                  │
├──────────┼───────────────────┤
│"Norte"   │3                  │
└──────────┴───────────────────┘
```

6.2.5) Query - SQL

```
select r.designation, count(f.name)
from facilities f
inner join roomtypes r on r.roomtype = f.roomtype
inner join municipalities m on m.cod = f.municipality
inner join regions r on r.cod = m.region
where r.description like '%touros%'
group by r.designation;
```

6.2.6) Result SQL



	DESIGNATION	COUNT(F.NAME)
1	Lisboa	6
2	Norte	3
3	Centro	11
4	Alentejo	43
5	Algarve	1

6.2.7) Query times

The query times are:

- **MongoDB : 0.021 sec**
- **Neo4J: 0.002 sec**
- **SQL: 0.021 sec**

6.3) Query c

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

6.3.1) Query - mongoDB

```
db.cultural_facilities.aggregate(
  {"$unwind": "$municipalities"},
  {"$unwind": "$municipalities.facilities"},
  {$match : { "municipalities.facilities.activities": { "$nin": ["cinema"] } }},
  {$count: "municipalities with facilities with no cinema"})
```

6.3.2) Result - mongoDB

```
{ 'municipalities with facilities with no cinema': 100 }
```

6.3.3) Query - Neo4J

```
match (m:Municipality)
match  (m1:Municipality)<-[:LOCATED_AT]-(f:Facility)-[:HAS]->(:Activity
{activity: "cinema"})
with count(distinct m1) as non_cinema_count, count(distinct m) as total_count
return total_count - non_cinema_count as non_cinema;
```

6.3.4) Result Neo4J

```
╒════════════╕
│"non_cinema"│
╞════════════╡
│100         │
└────────────┘
```

6.3.5) Query - SQL

[QUERY 1]

```
select  c.counter - count(distinct m.cod) as non_cinema
from facilities f
inner join uses u on u.id = f.id
inner join activities a on u.ref = a.ref
inner join municipalities m on f.municipality = m.cod
```

```
cross join (select count(*) as counter from municipalities)  c
where a.activity = 'cinema'
group by counter;
```

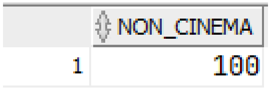
Cost: 227

[QUERY 2]

```
select count(*) - c.has_cinema as hasnt_cinema
from municipalities
cross join (
select count(distinct m.cod) as has_cinema
from facilities f
inner join uses u on u.id = f.id
inner join activities a on u.ref = a.ref
inner join municipalities m on f.municipality = m.cod
where a.activity = 'cinema') c
group by c.has_cinema;
```

Cost: 308

6.3.6) Result SQL



	NON_CINEMA
1	100

6.3.7) Query times

The query times are:

- **MongoDB : 0.024 sec**
- **Neo4J: 0.010 sec**
- **SQL: 0.028 sec**

6.4) Query 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.

6.4.1) Query - mongoDB

The following query was created considering that it might have one municipality that contains the maximum number of a specific activity, however the result contains unnecessary information.

```
db.cultural_facilities.aggregate([
    {$unwind: "$municipalities"},
    {$unwind: "$municipalities.facilities"},
    {$unwind: "$municipalities.facilities.activities"},
    {$group: {_id: {municipality: "$municipalities.designation", activity:
"$municipalities.facilities.activities"}  , numTimes: {$sum: 1}}},
    {$group: {_id: {activity: "$_id.activity"}, maxTimes: {$max: "$numTimes"},
docs: {$push : {municipality: "$_id.municipality", numTimes: "$numTimes"}}}},
    {$project: {
      "_id": 0,
      "municipality": "$_id.municipality",
      "activity": "$_id.activity",
      "maxTimes": "$maxTimes",
      "facilities": {"$filter": {
        input: "$docs",
        as: "element",
        cond: {$eq: ["$$element.numTimes", "$maxTimes"]}
      }
    }}},
  ])
```

6.4.2) Result - mongoDB

[QUERY 1]

```
{ activity: 'circo',
  maxTimes: 2,
  facilities: [ { municipality: 'Lisboa', numTimes: 2 } ] }
{ activity: 'cinema',
  maxTimes: 96,
  facilities: [ { municipality: 'Lisboa', numTimes: 96 } ] }
{ activity: 'dança',
  maxTimes: 47,
  facilities: [ { municipality: 'Lisboa', numTimes: 47 } ] }
{ activity: 'música',
  maxTimes: 77,
  facilities: [ { municipality: 'Lisboa', numTimes: 77 } ] }
{ activity: 'tauromaquia',
  maxTimes: 4,
  facilities: [ { municipality: 'Moura', numTimes: 4 } ] }
{ activity: 'teatro',
  maxTimes: 66,
  facilities: [ { municipality: 'Lisboa', numTimes: 66 } ] }
```

[QUERY 2]

```
{ municipality: 'Lisboa', activity: 'circo', numTimes: 2 }
{ municipality: 'Lisboa', activity: 'cinema', numTimes: 96 }
{ municipality: 'Lisboa', activity: 'música', numTimes: 77 }
{ municipality: 'Moura', activity: 'tauromaquia', numTimes: 4 }
{ municipality: 'Lisboa', activity: 'teatro', numTimes: 66 }
{ municipality: 'Lisboa', activity: 'dança', numTimes: 47 }
```

6.4.3) Query - Neo4J

```
MATCH (a:Activity)<-[:HAS]-(f:Facility)-[:LOCATED_AT]->(m:Municipality) with
a,m,count(f) as num_fac WITH a, collect(m) as mun, collect(num_fac) as facs WITH a, mun,
facs, reduce(x=[0,0], idx in range(0,size(facs)-1) | case when facs[idx] > x[1] then
[idx,facs[idx]] else x end)[0] as index return a.activity AS Activity,
mun[index].designation AS Municipality, facs[index] As num ORDER BY Activity
```

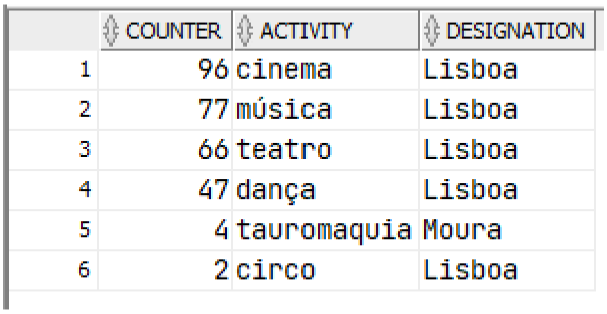
6.4.4) Result Neo4J

```
╒═════════════╤══════════════╤═════╕
│"Activity"   │"Municipality"│"num"│
╞═════════════╪══════════════╪═════╡
│"cinema"     │"Lisboa"      │96   │
├─────────────┼──────────────┼─────┤
│"circo"      │"Lisboa"      │2    │
├─────────────┼──────────────┼─────┤
│"dança"      │"Lisboa"      │47   │
├─────────────┼──────────────┼─────┤
│"música"     │"Lisboa"      │77   │
├─────────────┼──────────────┼─────┤
│"tauromaquia"│"Moura"       │4    │
├─────────────┼──────────────┼─────┤
│"teatro"     │"Lisboa"      │66   │
└─────────────┴──────────────┴─────┘
```

6.4.5) Query - SQL

```
create view max_counter_activity as (
select max(counter) as counter, activity
from (
    select count(a.ref) as counter, a.activity as activity, m.designation as
designation
    from facilities f
    inner join uses u on u.id = f.id
    inner join activities a on a.ref = u.ref
    inner join municipalities m on m.cod = f.municipality
    group by a.activity, m.designation
)
group by activity);

create view counts as (
select count(a.ref) as counter, a.activity as activity, m.designation as
designation
from facilities f
inner join uses u on u.id = f.id
inner join activities a on a.ref = u.ref
inner join municipalities m on m.cod = f.municipality
group by a.activity, m.designation);

select m.counter, m.activity, c.designation
from max_counter_activity m
inner join counts c on m.activity = c.activity and m.counter = c.counter
order by counter desc;
```

6.4.6) Result SQL



	COUNTER	ACTIVITY	DESIGNATION
1	96	cinema	Lisboa
2	77	música	Lisboa
3	66	teatro	Lisboa
4	47	dança	Lisboa
5	4	tauromaquia	Moura
6	2	circo	Lisboa

6.3.7) Query times

The query times are:

- **MongoDB : 0.023 sec**

- **Neo4J: 0.015 sec**
- **SQL: 0.024 sec**

6.5) Query e

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

6.5.1) Query - mongoDB

```
db.cultural_facilities.aggregate([
  {"$match": {"municipalities.facilities": {$ne: null}}},
  {"$project":{
    _id: 0,
    code: "$_id",
    designation: "$designation"
  }}
])
```

6.5.2) Result - mongoDB

```
{ code: 7, designation: 'Évora' }
{ code: 11, designation: 'Lisboa' }
{ code: 12, designation: 'Portalegre' }
{ code: 15, designation: 'Setúbal' }
```

6.5.3) Query - Neo4J

```
MATCH (m:Municipality) WHERE  NOT ()-[:LOCATED_AT]->(m) WITH collect(m) as mun
MATCH (m:Municipality)-[:BELONGS_TO]->(d:District) WHERE ALL(x IN mun WHERE NOT
(x)--(d)) WITH DISTINCT d RETURN d.cod, d.designation
```

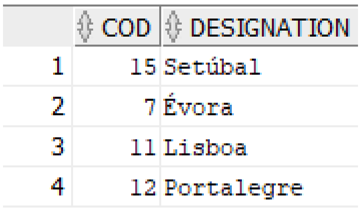
6.5.4) Result Neo4J

```
╒════════╤════════════════╕
│"d.cod" │"d.designation" │
╞════════╪════════════════╡
│12      │"Portalegre"    │
├────────┼────────────────┤
│15      │"Setúbal"       │
├────────┼────────────────┤
│7       │"Évora"         │
├────────┼────────────────┤
│11      │"Lisboa"        │
└────────┴────────────────┘
```

6.5.5) Query - SQL

```
select cod,district from municipalities;
select * from facilities;

create view districtsWithNoFacilities as select cod,district from municipalities
where not exists(select municipality from facilities where
facilities.municipality = municipalities.cod);

select cod,designation from districts where cod in (select distinct district
from municipalities m
where not exists (
    select cod
    from districtsWithNoFacilities df
    where df.district = m.district
));
```

6.5.6) Result SQL



	COD	DESIGNATION
1	15	Setúbal
2	7	Évora
3	11	Lisboa
4	12	Portalegre

6.3.7) Query times

The query times are:

- **MongoDB : 0.039 sec**
- **Neo4J: 0.11 sec**
- **SQL: 0.082 sec**

6.6) Query f

Show, for each district, the average capacity of the facilities.

6.6.1) Query - mongoDB

```
db.getCollection("cultural_facilities").aggregate(
   [
   {$unwind: "$municipalities"},
   {$unwind: "$municipalities.facilities"},
   {$group:{_id: {"cod": "$_id", "designation": "$designation"}, avgCapacity:
{$avg: "$municipalities.facilities.capacity" }}},
   {$project: {
      "_id": "$_id.cod",
      "district": "$_id.designation",
      "avgCapacity": "$avgCapacity"
   }}
   ]
)
```

6.6.2) Result - mongoDB

```
{ _id: 13, district: 'Porto', avgCapacity: 359.38 }
{ _id: 1, district: 'Aveiro', avgCapacity: 332.344262295082 }
{ _id: 18, district: 'Viseu', avgCapacity: 274.7647058823529 }
{ _id: 17, district: 'Vila Real', avgCapacity: 246.85714285714286 }
{ _id: 16, district: 'Viana do Castelo', avgCapacity: 396.1333333333333 }
{ _id: 15, district: 'Setúbal', avgCapacity: 528.6904761904761 }
{ _id: 2, district: 'Beja', avgCapacity: 794 }
{ _id: 7, district: 'Évora', avgCapacity: 887.6486486486486 }
{ _id: 3, district: 'Braga', avgCapacity: 552.0192307692307 }
{ _id: 5, district: 'Castelo Branco', avgCapacity: 316.0952380952381 }
{ _id: 4, district: 'Bragança', avgCapacity: 688.7272727272727 }
{ _id: 9, district: 'Guarda', avgCapacity: 431.4074074074074 }
{ _id: 10, district: 'Leiria', avgCapacity: 417.22641509433964 }
{ _id: 14, district: 'Santarém', avgCapacity: 1080 }
{ _id: 6, district: 'Coimbra', avgCapacity: 383.25531914893617 }
{ _id: 8, district: 'Faro', avgCapacity: 451.38709677419354 }
{ _id: 11, district: 'Lisboa', avgCapacity: 397.72607260726073 }
{ _id: 12, district: 'Portalegre', avgCapacity: 1019.9722222222222 }
```

6.6.3) Query - Neo4J

```
MATCH (f:Facility)-[:LOCATED_AT]->(m:Municipality)-[:BELONGS_TO]->(d:District)
WITH d.designation as Distrito, round(apoc.coll.avg(collect(f.capacity))) as
Capacidade_Media RETURN Distrito, Capacidade_Media
```

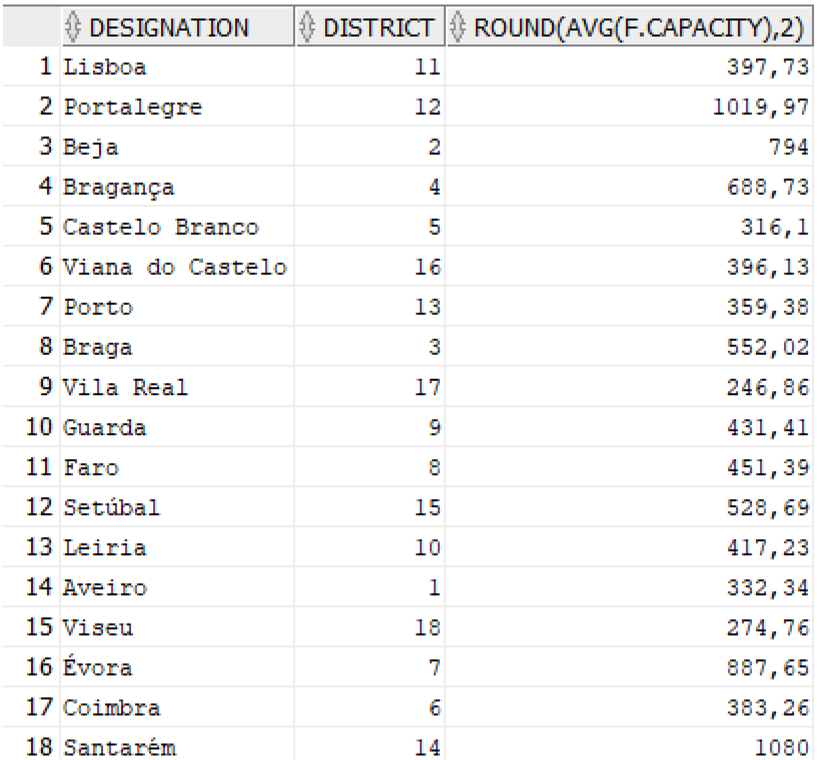
6.6.4) Result Neo4J

```
╒══════════════════╤══════════════════╕
│"Distrito"        │"Capacidade_Media"│
╞══════════════════╪══════════════════╡
│"Portalegre"      │1019.97           │
├──────────────────┼──────────────────┤
│"Porto"           │359.38            │
├──────────────────┼──────────────────┤
│"Santarém"        │1080.0            │
├──────────────────┼──────────────────┤
│"Setúbal"         │528.69            │
├──────────────────┼──────────────────┤
│"Viana do Castelo"│396.13            │
├──────────────────┼──────────────────┤
│"Vila Real"       │246.86            │
├──────────────────┼──────────────────┤
│"Viseu"           │274.76            │
├──────────────────┼──────────────────┤
│"Aveiro"          │332.34            │
├──────────────────┼──────────────────┤
│"Beja"            │794.0             │
├──────────────────┼──────────────────┤
│"Braga"           │552.02            │
├──────────────────┼──────────────────┤
│"Bragança"        │688.73            │
├──────────────────┼──────────────────┤
│"Castelo Branco"  │316.1             │
├──────────────────┼──────────────────┤
│"Coimbra"         │383.26            │
├──────────────────┼──────────────────┤
│"Évora"           │887.65            │
├──────────────────┼──────────────────┤
│"Faro"            │451.39            │
├──────────────────┼──────────────────┤
│"Guarda"          │431.41            │
├──────────────────┼──────────────────┤
│"Leiria"          │417.23            │
├──────────────────┼──────────────────┤
│"Lisboa"          │397.73            │
└──────────────────┴──────────────────┘
```

6.6.5) Query - SQL

```
select * from facilities;
select * from municipalities;

create view facilitiesByDistrict as
select f.name,f.capacity,m.district
from facilities f
inner join municipalities m
on f.municipality = m.cod;

select d.designation,f.district,round(avg(f.capacity),2)
from facilitiesByDistrict f, districts d
where d.cod=f.district
group by f.district,d.designation;
```

6.6.6) Result SQL



	DESIGNATION	DISTRICT	ROUND(AVG(F.CAPACITY),2)
1	Lisboa	11	397,73
2	Portalegre	12	1019,97
3	Beja	2	794
4	Bragança	4	688,73
5	Castelo Branco	5	316,1
6	Viana do Castelo	16	396,13
7	Porto	13	359,38
8	Braga	3	552,02
9	Vila Real	17	246,86
10	Guarda	9	431,41
11	Faro	8	451,39
12	Setúbal	15	528,69
13	Leiria	10	417,23
14	Aveiro	1	332,34
15	Viseu	18	274,76
16	Évora	7	887,65
17	Coimbra	6	383,26
18	Santarém	14	1080

6.3.7) Query times

The query times are:

- **MongoDB : 0.163 sec**
- **Neo4J: 0.006 sec**
- **SQL: 0.279 sec**

7) Analysis

After doing some queries on the NoSQL and Graph databases technologies (i.e. MongoDB and Neo4j) and on the SQL Developer (relational model), we could observe that the performance on standard SQL (e.g. without indexes and prior usage of PL/SQL) is worse than in the other technologies tested. Due to the fact that the join operations in the SQL queries come with a cost.

We can also verify that the query complexity of Neo4j and MongoDB is inferior when compared to the SQL. But, comparing MongoDB and Neo4j, it's evident that Neo4j is a powerful tool when it comes to query answering: it's possible to create elegant, clear queries easily in just a few lines.

8) Resources

[1] - Understanding Neo4j's data on disk