



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Francesco Galbiati - 10728028**

Group Number: **116**

Academic Year: 2024-2025

Contents

Contents	i
----------	---

1	Introduction	1
1.1	Objective	1
1.2	The Dataset	1
1.3	Technological Choices	2
2	Dataset Description	3
2.1	Non-Relational Schema	3
2.2	Fields Description	7
3	Queries	11
3.1	Most frequent origins among placenames	11
3.1.1	Preliminary Information	11
3.1.2	Query Explanation	11
3.1.3	Query Text	12
3.1.4	Results	12
3.2	Names and physical features: Lakes	13
3.2.1	Preliminary Information	13
3.2.2	Query Explanation	13
3.2.3	Query Text	14
3.2.4	Results	14
3.3	Placenames and Folklore: the <i>Púca</i>	15
3.3.1	Preliminary Information	15
3.3.2	Query Explanation	15
3.3.3	Query Text	16
3.3.4	Results	17
3.4	Placenames related to animals	18

3.4.1	Preliminary Information	18
3.4.2	Query Explanation	18
3.4.3	Query Text	19
3.4.4	Results	22
3.5	Most frequent placename of the day	22
3.5.1	Preliminary Information	22
3.5.2	Query Explanation	23
3.5.3	Query Text	23
3.5.4	Results	24
3.6	The Irish Grid Reference System	24
3.6.1	Preliminary Information	24
3.6.2	Query Explanation	25
3.6.3	Query Text	25
3.6.4	Results	28
3.7	City or town with most streets per county	30
3.7.1	Preliminary Information	30
3.7.2	Query Explanation	30
3.7.3	Query Text	31
3.7.4	Results	33
3.8	Longest placenames	34
3.8.1	Preliminary Information	34
3.8.2	Query Explanation	34
3.8.3	Query Text	35
3.8.4	Results	36
3.9	Gaeltacht frequency by county	37
3.9.1	Preliminary Information	37
3.9.2	Query Explanation	37
3.9.3	Query Text	38
3.9.4	Results	39
3.10	Irish surnames by prefix	40
3.10.1	Preliminary Information	40
3.10.2	Query Explanation	40
3.10.3	Query Text	41
3.10.4	Results	44
4	Extra: Analysis using pymongo	45
4.1	Analysis of Query 2	45

4.1.1	Analysis Explanation	45
4.1.2	Results	46
4.2	Analysis of Query 9	48
4.2.1	Analysis Explanation	48
4.2.2	Results	48
4.3	Analysis of Query 3	49
4.3.1	Analysis Explanation	49
4.3.2	Results	49

1 | Introduction

1.1. Objective

Every place on Earth has a name, and each name carries a meaning that is, or has been, relevant to the people who assigned it to that specific place; for this reason, the naming of a place provides interesting insights into the culture of the population that lives in it. The focus of this work is Ireland (including both the Republic of Ireland and Northern Ireland), where the real meaning behind placenames is often hidden by their anglicisation, but can still be understood by looking at the original versions of the names in the Irish language.

The objective of this project is to use one of the main NoSQL DBMS technologies presented during the "System and Methods for Big and Unstructured Data" course at Politecnico di Milano to analyse the *logainm.ie* dataset, in order to extract insightful information about the Irish language and geography. The dataset contains huge amounts of information about the different places of Ireland and their names, with a focus on the Republic of Ireland, while Northern Ireland has fewer data available. The dataset is property of the Republic of Ireland government and is made publicly available by the [Gaois research group](#) through their APIs.

1.2. The Dataset

The *logainm.ie* dataset is available for consultation under an open data license. To access the data through the APIs it's first necessary to create a Gaois API key by following the instructions at this [link](#). The dataset can then be queried using the proper [resource paths](#), and results will be returned in JSON formatted pages. The dataset itself, which will be thoroughly described in a dedicated section later, includes 126752 locations, which may vary in category and extension; each location is identified by an id which can also be used in the *logainm.ie* search bar to access graphical information about the place. No data wrangling was performed on the dataset, which has been imported and used as-is.

1.3. Technological Choices

The technology chosen to analyse the chosen dataset is a documental approach, in particular MongoDB. The main reason for this choice resides in the structure of the dataset: each location is conceptually organized as a document that lists many relevant characteristics of the place, like the different names associated with it, the category of the place, and the different features or other places that may be included in it; given this initial configuration, a documental approach is a natural and intuitive option. Furthermore, the dataset itself is already provided in a JSON format, which is well-suited to be imported into MongoDB.

2 | Dataset Description

2.1. Non-Relational Schema

For each location present in the *logainm* dataset, a document lists all the useful information about it using this non-relational data schema:

```
{
  "_id": objectid,
  "id": int32,
  "dateCreated": string,
  "dateModified": string,
  "permalink": string,
  "featured": Array[string],
  "cluster": Object {
    "focusID": int32,
    "members": Array[
      Object {
        "placeID": int32,
        "category": {
          "id": string,
          "nameEN": string,
          "nameGA": string
        }
      }
    ]
  },
  "placenames": Array[
    Object {
      "id": int32,
      "language": string,
      "wording": string,
```

```

    "genitive": string,
    "main": boolean,
    "acceptability": {
      "id": int32,
      "textEN": string,
      "textGA": string
    },
    "audio": {
      "filename": string,
      "uri": string
    },
    "subNames": {
      "disambiguates": boolean,
      "text": string
    }
  }
],
"glossary": Array[
  Object {
    "id": int32,
    "headword": string,
    "translation": string
  }
],
"categories": Array[
  Object {
    "id": string,
    "nameEN": string,
    "nameGA": string,
    "namePluralEN": string,
    "namePluralGA": string
  }
],
"includedIn": Array[
  Object {
    "id": int32,
    "nameEN": string,

```

```

        "nameGA": string,
        "category": {
            "id": string,
            "nameEN": string,
            "nameGA": string
        }
    },
    ],
    "includes": Array[
        "id": string,
        "nameEN": string,
        "nameGA": string,
        "namePluralEN": string,
        "namePluralGA": string,
        "count": int32
    ],
    "geography": {
        "accurate": boolean,
        "source": string,
        "coordinates": {
            "latitude": double,
            "longitude": double
        }
    },
    },
    "gridReferences": Array[
        Object {
            "square": string,
            "easting": int32,
            "northing": int32
        }
    ],
    "gaeltacht": {
        "extent": string
    },
    "postOffice": {
        "extent": string
    },
    },

```

```
"northernIreland": {
  "extent": string
},
"images": Array[
  Object {
    "fileName": string,
    "labelEN": string,
    "labelGA": string,
    "uri": string
  }
],
"links": Array[
  Object {
    "target": string,
    "type": string
  }
],
"bornHere": Array[
  Object {
    "id": int32,
    "title": string,
    "uri": string
  }
],
"folklore": Array[
  Object {
    "type": string,
    "uriEN": string,
    "uriGA": string
  }
],
"sameAs": Array[
  Object {
    "uri": string
  }
]
}
```

2.2. Fields Description

Follows the description of all the attributes listed in the previous pages, a more in-depth version of it can be found in the [Gaois documentation](#):

- **_id**: the id of the document assigned by MongoDB
- **id**: the id of the place in the *logainm.ie* dataset, it can also be used in the search bar on the dedicated [website](#)
- **dateCreated**: the date and time the location was inserted in the dataset.
- **dateModified**: the last date and time the entry of the location was modified.
- **permalink**: a permanent link to the *logainm.ie* page dedicated to the location.
- **featured**: an array of dates when the place was chosen as "placename of the day" on the *logainm.ie* website.
- **cluster**: a cluster is a group of places that share the same name and are near each other. A cluster is represented by the following attributes:
 - **focusID**: the id of the place most readily associated with the name, such a place is called the "focus" of the cluster.
 - **members**: the individual places that make up the cluster, represented by the id and category (see the later definition of category).
- **placenames**: each place includes one or more names associated with it, the general case is a name in English and a name in Irish, but this may change in some cases. Each placename has the following attributes:
 - **id**: the identifier of the placename.
 - **language**: the language in which the placename is expressed ("en" for English or "ga" for Irish).
 - **wording**: the placename in written form.
 - **genitive**: in the case of an Irish language name, the dataset also reports its genitive case.
 - **main**: true if the name can be considered the main one associated with the place in the specified language.

- **acceptability**: states if a particular name is to be considered "official", the description of this status is provided in both languages.
- **audio**: a link to a recording of the pronunciation of the placename.
- **subNames**: in the case of a placename composed of different conjoined names, each object in this array provides one of those names. If one part of the name has a disambiguation function the **disambiguates** boolean is set to true, otherwise to false.
- **glossary**: the glossary contains the main words from which the Irish language placenames take their origin, for each placename the glossary provides a set of headwords in Irish and their translations in English; each item in the glossary also has a unique id. Another important aspect of the glossary is that it groups all the different possible grammatical forms of a name under its nominative case, in this way, we don't need to take into account different variations of the same word when matching on the glossary's headwords.
- **category**: this field is dedicated to the category to which the place belongs, a category may be administrative units (counties, baronies, civil parishes, townlands, ...) or geographical features (lakes, hills, islands, ...), and each of them is reported in English and Irish, both in the singular and plural forms.
- **includedIn**: this array lists all the other places in the dataset that include the location; for example, a barony is included in a county. For each location, the dataset lists the id, the names in both languages and the category.
- **includes**: this field provides a list of all the places categories included in the location. Each category is characterized by an id, a name in English and one in Irish, both in singular and plural forms, and a count field, corresponding to how many places of a certain category are included in the location.
- **geography**: the geographical information of the location, expressed in terms of latitude and longitude, other attributes are the accuracy of the coordinates (true or false) and the source from which they were taken (e.g. Ordnance Survey Ireland (osi), extrapolation, grid or manual).
- **gridReferences**: the geographical information of a place, expressed in terms of the [Irish Grid Reference System](#), in which each location is characterized by the square in which it is located, the easting and the northing values.
- **gaeltacht**: the gaeltacht field is present only if the location is, at least partially, part

of a [Gaeltacht](#), specific area of territory where Irish is recognized as the predominant language. The "extent" value is "all" if the place is entirely included in such a district, or "part" if the inclusion is only partial.

- **postOffice**: indicates the presence, now or in the past, of a post office in the location, the "extent" value is "all" or the attribute is not present at all in the document.
- **northernIreland**: if the place is in Northern Ireland, the "extent" field has "all" as a value, otherwise the attribute is not present.
- **images**: a list of image files relative to the location, each file is described by the name, the label in Irish and English and the link to the image.
- **links**: provides a list of links to external resources relative to various aspects of the place described by the document
- **bornHere**: lists the names of the people born in the location who are present in the [ainm.ie](#) database. Each person is identified by an id and characterized by the "title" attribute, which includes the name and the years of birth and death. A direct link to the *ainm.ie* website is also available in the "uri" attribute.
- **folklore**: provides folkloric data about the location by referencing the [dúchas.ie](#) National Folklore Collection. Different types of collections are linked in this manner, such as manuscripts, texts written by school students and images.
- **sameAs**: specifies other external references to the place in datasets other than the Placenames Database of Ireland.

3 | Queries

This section provides the queries requested by the project assignment; each of them will be preceded by some preliminary contextual information (i.e. the question answered by the query and how the analysed real-world phenomena are mapped onto the dataset), followed by an explanation of the query, comprising a natural language description of the code of the query. Lastly, the results of the query are shown, using a textual representation, aided by a graphical one when needed. Suggestions for the topics of some of the queries have been taken from the [themes](#) and [educational resources](#) sections of [logainm.ie](#).

3.1. Most frequent origins among placenames

3.1.1. Preliminary Information

As already mentioned, the primary focus of the dataset chosen for this project is the placenames of Ireland and their origins. The name of the dataset itself, the word *logainm*, means "placename" and is composed of two parts: *log*, meaning "place" and *ainm*, which translates to "name". A first simple query that can be performed on the available data is to extract the ten most frequent words found in the placenames included in the dataset. To this avail, we can employ the **glossary** attribute, which, for each place, contains an entry for every relevant word found in its name.

3.1.2. Query Explanation

The query is a simple aggregation composed by four stages:

- A first **\$unwind** stage, used to separate the elements in the glossary array of each document.
- A **\$group** stage, in which the documents are grouped by their headwords and translations. The group stage also counts the number of documents in each group.
- The groups are then sorted, using a **\$sort**, by their count in descending order.

- Lastly, we use a **\$limit** to keep only the documents with the ten most frequent headwords.

3.1.3. Query Text

```
db.logainm.aggregate([
  {
    $unwind: "$glossary"
  },
  {
    $group: {
      _id: {
        headword: "$glossary.headword",
        translation: "$glossary.translation"
      },
      count: {
        $sum: 1
      }
    }
  },
  {
    $sort: {
      count: -1
    }
  },
  {
    $limit: 10
  }
])
```

3.1.4. Results

When run using MongoDB, the query produces the following result:

```
{
  "_id": {
    "headword": "baile",
    "translation": "townland, town, homestead"
  },
  "count": 9538
},
{
  "_id": {
    "headword": "cill",
    "translation": "church"
  },
  "count": 3273
},
{
  "_id": {
    "headword": "bóthar",
    "translation": "road"
  },
  "count": 2760
},
{
  "_id": {
    "headword": "mór",
    "translation": "great, big"
  },
  "count": 2654
},
{
  "_id": {
    "headword": "Ó",
    "translation": ""
  },
  "count": 2393
},
}
```

```
{
  "_id": {
    "headword": "cnoc",
    "translation": "hill"
  },
  "count": 2363
},
{
  "_id": {
    "headword": "loch",
    "translation": "lake; inlet"
  },
  "count": 2094
},
{
  "_id": {
    "headword": "beag",
    "translation": "small"
  },
  "count": 1651
},
{
  "_id": {
    "headword": "cluain",
    "translation": "meadow, pasture"
  },
  "count": 1614
},
{
  "_id": {
    "headword": "droim",
    "translation": "ridge"
  },
  "count": 1537
},
}
```

Figure 3.1: Results of Query 1

As can be seen from the result shown above, different categories of words can be found in this list: there are names like *baile* or *bóthar* often used in names of towns and roads, names referring to natural features (*cnoc* and *loch*) and adjectives (*mór* and *beag*) that generally describe some kind of characteristic of a place. Other particular examples are *cill* ("church"), which is one of many examples of the influence of Christianity in Irish placenames, and *Ó*, a particle found in surnames, which is another kind of placename origin commonly found in Ireland.

3.2. Names and physical features: Lakes

3.2.1. Preliminary Information

As shown by the previous query, many placenames take their origin from some kind of natural features like hills, lakes or ridges. By querying the dataset we can easily confirm the correspondence between names and physical features using, for example, lakes.

3.2.2. Query Explanation

The query is a simple **find** operation that filters the document with two predicates:

- The **includes** attribute must contain an entry for the lake category (the id must be equal to "L").
- The **glossary** must contain an entry for the headword "loch".

After this step the query projects a small set of essential attributes about the place: the id in the dataset, the names associated with the place and the category in which the place belongs.

3.2.3. Query Text

```
db.logainm.find(  
  {  
    "includes.id": "L",  
    "glossary.headword": "loch"  
  },  
  {  
    "_id": 0,  
    "id": 1,  
    "placenames.language": 1,  
    "placenames wording": 1,  
    "categories.nameEN": 1  
  }  
)
```

3.2.4. Results

The query retrieves 53 different locations that respect the filtering constraints, belonging to different kinds of categories (mostly townlands, the smallest of the administrative division categories); while a couple of examples are shown below, the complete list is provided in the project folder.

```
{
  "id": 100019,
  "placenames": [
    {
      "language": "ga",
      "wording": "Loch Garman"
    },
    {
      "language": "ga",
      "wording": "An Contae Riabhach"
    },
    {
      "language": "en",
      "wording": "Wexford"
    }
  ],
  "categories": [
    {
      "nameEN": "county"
    }
  ]
},
```

```
{
  "id": 14779,
  "placenames": [
    {
      "language": "ga",
      "wording": "Taobh an Locha"
    },
    {
      "language": "en",
      "wording": "Tievelough"
    }
  ],
  "categories": [
    {
      "nameEN": "townland"
    }
  ]
},
```

Figure 3.2: Results from Query 2

The two examples illustrated above have very different origins for their name: Co. Wexford is called *Loch Garman* in Irish, which translates to "Lake of Garma" or "Lake of Garman", and has its roots in the Irish mythology, where it is said that Garman, a mythological character, drowned in the mouth of the Slaney, the main river in Co. Wexford. On the other hand, *Taobh an Locha* (meaning "lakeside") is a townland situated, as the name suggests, on the side of a small lake called "Loch Fhia" in Co. Donegal.

3.3. Placenames and Folklore: the *Púca*

3.3.1. Preliminary Information

Mythology and folklore are among the fundamental components of a culture, for this reason, it's not rare to encounter them when looking at placenames. An example of this phenomenon is the *Púca* (a word gaelicized from the English "pooka" or "puck"), a little mischievous spirit that, among other things, is said to be able to make blackberries inedible after *Oíche Shamhna*/Halloween. From the dataset point of view, there is no entry in the glossary for the word *Púca*, so, to retrieve the places with a name referencing it, the query will need to match the placenames against the words *Púca* and *Phúca* (the genitive case).

3.3.2. Query Explanation

The query is a 5-stage aggregation that retrieves the places whose names contain a variation of the word *Púca*, listing their names and county. The execution pipeline is as follows:

- a first **\$match** that filters the documents by comparing the placenames to the words *púca* and *phúca*. The match is performed using regular expression, to identify the searched text in any part of the placename.
- an **\$unwind** stage, which separates the various elements of the **includedIn** arrays.
- a second **\$match** that keeps only the documents with a county (id = "CON") in their **includedIn** field.
- a **\$project** selecting only the useful fields to show in the output.
- a **\$sort**, which sorts the resulting document by the county name.

3.3.3. Query Text

```
db.logainm.aggregate([
  {
    $match: {
      $or: [
        {
          "placenames.wording": {
            $regex: "Phúca"
          }
        },
        {
          "placenames.wording": {
            $regex: "Púca"
          }
        }
      ]
    }
  },
  {
    $unwind: "$includedIn"
  },
  {
    $match: {
      "includedIn.category.id": "CON"
    }
  }
])
```

```

    },
    {
      $project: {
        _id: 0,
        id: 1,
        "placenames.language": 1,
        "placenames.wording": 1,
        county: {
          nameGA: "$includedIn.nameGA",
          nameEN: "$includedIn.nameEN"
        }
      }
    },
    {
      $sort: {
        "county.nameGA": 1
      }
    }
  ]
})

```

3.3.4. Results

The resulting documents appear as follows:

```

{
  "id": 27009,
  "placenames": [
    {
      "language": "ga",
      "wording": "Baile an Phúcaigh"
    },
    {
      "language": "en",
      "wording": "Foulkstown"
    }
  ],
  "county": {
    "nameGA": "Cill Chainnigh",
    "nameEN": "Kilkenny"
  }
},

```

```

{
  "id": 113024,
  "placenames": [
    {
      "language": "ga",
      "wording": "Poll an Phúca"
    },
    {
      "language": "en",
      "wording": "Pollaphuca"
    }
  ],
  "county": {
    "nameGA": "Cill Mhantáin",
    "nameEN": "Wicklow"
  }
},

```

Figure 3.3: Results from Query 3

The first example, *Baile an Phúcaigh* (lit. "Town of *an Phúcaigh*"), is one of some interesting cases in which, while it may seem that the word *Phúcaigh* refers to the *Púca*, historical sources mostly suggest that this name has its origin in the English given name

Foulke, which was later gaelicized as *Fúca*, *Phúca*, or, in some cases, *Phúcaigh*. The second example (lit. "Pool/Hole/Cave of the Puck") is instead a real example of the presence of the *Púca* in Irish placenames that can be found multiple times in different regions of the island. Other interesting readings about this topic can be found at [this](#) and [this](#) links.

3.4. Placenames related to animals

3.4.1. Preliminary Information

Many placenames in Ireland also have their origins in the local fauna. We find many examples of places named after animals that are part of rural life, but there are also many instances of wild animals used in such names. The amount of animals that can be found in the dataset's names is huge, but, for the sake of simplicity, we will inspect only those that have an entry in the glossary: the fox (*sionnach*), the pig (*muc*), the dog (*madra*), the eagle (*iolar*), the wolf (*mac tíre*) and the calf (*gamhain* or *lao*). The objective of the query is to find the most common categories of places with the previously listed animals in their name.

3.4.2. Query Explanation

To account for the double entry of the calf, we need to split the execution pipeline in two, and then make a union of the two results. The aggregation is as follows:

- An **\$unwind** of the glossary, to separate the different entries of the array.
- A **\$match** on the animals' names, without the ones of the calf.
- A second **\$unwind** on the categories, to account for the rare cases in which the field has more than one entry.
- A first **\$group** that groups the documents by both category and glossary, counting the elements belonging to each group.
- A second **\$group** stage, only by glossary, that for each different category pushes an element in a new array, to create a list of categories as a sub-document.
- The same pipeline is performed for the calf using both of its names, the result is then reunited with the rest of the documents using a **\$unionWith** command.
- A final **\$project** is used to present the output in a more readable way.

3.4.3. Query Text

```
db.logainm.aggregate([
  {
    $unwind: "$glossary"
  },
  {
    $match: {
      "glossary.headword": {
        $in: ["muc", "iolar", "madra", "sionnach", "mac tíre"]
      }
    }
  },
  {
    $unwind: "$categories"
  },
  {
    $group: {
      _id: {
        category_id: "$categories.id",
        category_name: "$categories.nameEN",
        glossaryGA: "$glossary.headword",
        glossaryEN: "$glossary.translation"
      },
      count: {
        $sum: 1
      }
    }
  },
  {
    $group: {
      _id: {
        animalGA: "$_id.glossaryGA",
        animalEN: "$_id.glossaryEN"
      },
      categories: {
        $push: {
```

```

        category: "$_id.category_name",
        count: "$count"
    }
}
}
},
{
    $unionWith: {
        coll: "logainm",
        pipeline: [
            {
                $unwind: "$glossary"
            },
            {
                $match: {
                    $or: [
                        {
                            "glossary.headword": "gamhain"
                        },
                        {
                            "glossary.headword": "lao"
                        }
                    ]
                }
            },
            {
                $unwind: "$categories"
            },
            {
                $group: {
                    _id: {
                        category_id: "$categories.id",
                        category_name: "$categories.nameEN",
                        glossaryGA: "gamhain/lao",
                        glossaryEN: "calf"
                    },
                    count: {

```

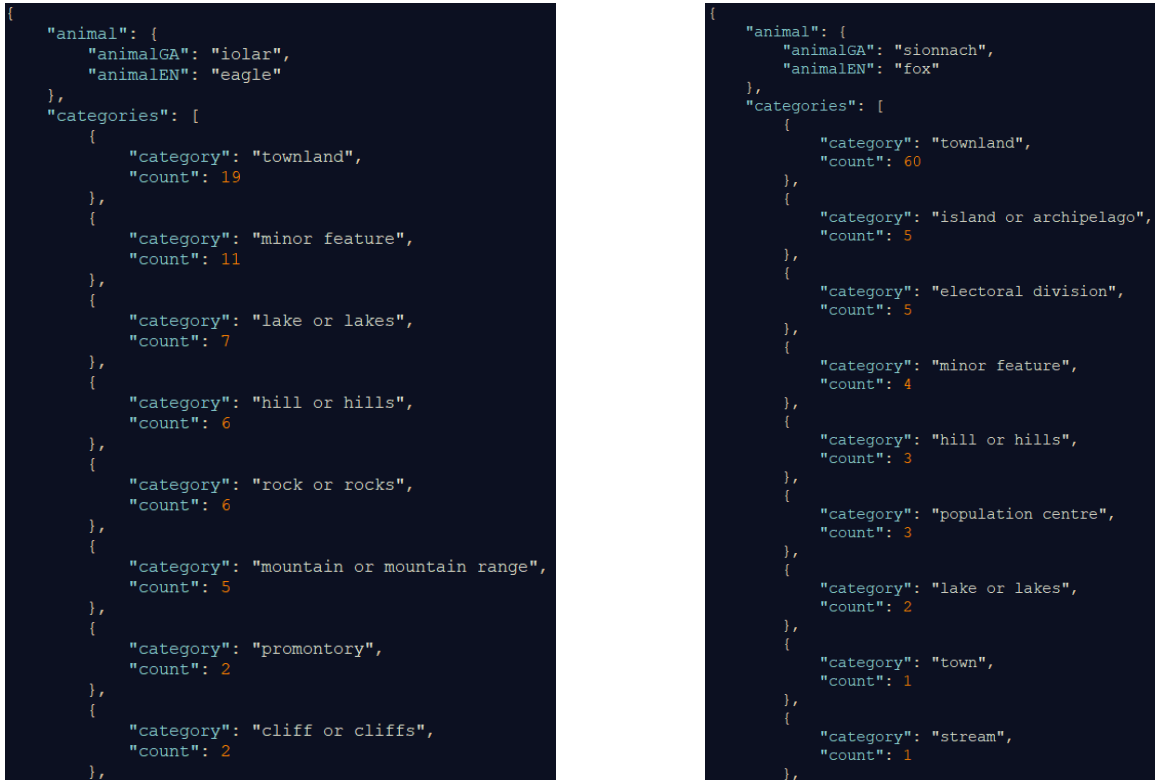
```

        $sum: 1
      }
    }
  },
  {
    $group: {
      _id: {
        animalGA: "$_id.glossaryGA",
        animalEN: "$_id.glossaryEN"
      },
      categories: {
        $push: {
          category: "$_id.category_name",
          count: "$count"
        }
      }
    }
  }
]
}
},
{
  $project: {
    _id: 0,
    animal: "$_id",
    categories: {
      $sortBy: {
        input: "$categories",
        sortBy: {
          count: -1
        }
      }
    }
  }
}
}
})

```

3.4.4. Results

The resulting documents are presented in this form (since some documents can be very long, only a partial visualization of them is provided):



```
{
  "animal": {
    "animalGA": "iolar",
    "animalEN": "eagle"
  },
  "categories": [
    {
      "category": "townland",
      "count": 19
    },
    {
      "category": "minor feature",
      "count": 11
    },
    {
      "category": "lake or lakes",
      "count": 7
    },
    {
      "category": "hill or hills",
      "count": 6
    },
    {
      "category": "rock or rocks",
      "count": 6
    },
    {
      "category": "mountain or mountain range",
      "count": 5
    },
    {
      "category": "promontory",
      "count": 2
    },
    {
      "category": "cliff or cliffs",
      "count": 2
    }
  ]
}
```

```
{
  "animal": {
    "animalGA": "sionnach",
    "animalEN": "fox"
  },
  "categories": [
    {
      "category": "townland",
      "count": 60
    },
    {
      "category": "island or archipelago",
      "count": 5
    },
    {
      "category": "electoral division",
      "count": 5
    },
    {
      "category": "minor feature",
      "count": 4
    },
    {
      "category": "hill or hills",
      "count": 3
    },
    {
      "category": "population centre",
      "count": 3
    },
    {
      "category": "lake or lakes",
      "count": 2
    },
    {
      "category": "town",
      "count": 1
    },
    {
      "category": "stream",
      "count": 1
    }
  ]
}
```

Figure 3.4: Results from Query 4

From these results it can be interesting to notice the different types of places that are usually named after different species, for example, it can be seen how the eagle is more often associated with hills or mountains, while the fox is more present in the names of population centres, administrative divisions and environments more suited to its habitat. Another interesting result that can be found in the complete results file is the wolf, an extinct species in Ireland, but that still lives in some placenames.

3.5. Most frequent placename of the day

3.5.1. Preliminary Information

"Placename of the day" is a section of *logainm.ie* in which every day a place is chosen to be featured on the front page of the website. A query that may produce interesting results is to extract the most frequently appearing of these placenames. The dataset has

a dedicated field for each document, called **featured**, that lists all the dates in which the placename has been chosen. An important consideration is that the query has been performed on a dataset downloaded at the start of December 2024, so it's not updated with the last entries.

3.5.2. Query Explanation

The query is a very simple 3-stage aggregation, composed as follows:

- A **\$project** that for each document only keeps the useful attributes and creates a new field called **numFeatured**, which holds the number of times each place has been featured, computed using **\$size** on the **featured** array
- A **\$sort** of the documents by the number of featurings in decreasing order.
- A **\$limit** that keeps only the ten most featured placenames.

3.5.3. Query Text

```
db.logainm.aggregate([
  {
    $project: {
      _id: 0,
      id: 1,
      "placenames.language": 1,
      "placenames wording": 1,
      numFeatured: {
        $size: "$featured"
      }
    }
  },
  {
    $sort: {
      "numFeatured": -1
    }
  },
  {
    $limit: 10
  }
])
```

1)

3.5.4. Results

Some of the resulting documents are reported below:

```
{
  "id": 21027,
  "placenames": [
    {
      "language": "ga",
      "wording": "Buaille na Nollag"
    },
    {
      "language": "en",
      "wording": "Boleynanollag"
    }
  ],
  "numFeatured": 31
},
{
  "id": 17542,
  "placenames": [
    {
      "language": "ga",
      "wording": "Inis Mac Neasáin"
    },
    {
      "language": "en",
      "wording": "Ireland's Eye"
    }
  ],
  "numFeatured": 14
},
{
  "id": 16148,
  "placenames": [
    {
      "language": "ga",
      "wording": "Ailt an tSneachta"
    },
    {
      "language": "en",
      "wording": "Altatraght"
    }
  ],
  "numFeatured": 9
},
{
  "id": 40582,
  "placenames": [
    {
      "language": "ga",
      "wording": "Mullach na Cásca"
    },
    {
      "language": "en",
      "wording": "Mullanacask"
    }
  ],
  "numFeatured": 9
}
```

Figure 3.5: Results from Query 5

From the results of the query, it can be seen that *Buaille na Nollag* is the most featured placename of the entire dataset, this fact is easily explainable: it is the only example of a placename with a direct reference to Christmas in its name; *Buaille na Nollag* means, in fact, "Cattle-fold of Christmas" and is a townland located in the south-east of Co. Galway. This is not, however, the only instance of festivities mentioned in Irish placenames, another example is at the tenth position: *Mullach na Cásca*, meaning "Hilltop of Easter".

3.6. The Irish Grid Reference System

3.6.1. Preliminary Information

Aside from data about the placenames of Ireland, the *logainm.ie* dataset also contains plenty of information about Irish geography. The position of a place can be expressed in

two ways: using coordinates or using the grid reference system. The Irish Grid Reference System divides the map of Ireland into 25 squares, each represented by a letter of the alphabet, while inside the single squares, the position of a place is represented by two values called "northing" (latitude) and "easting" (longitude) that have their origin point in the south-west corner of the square. An interesting query to perform with MongoDB can be the use of the grid reference systems to get the northmost, eastmost, southmost and westmost places in the dataset. Since the northing and easting values are not absolute, the query will first need to select the correct square on which to check the coordinate values, and only then select the largest or smallest ones.

3.6.2. Query Explanation

The query consists of an aggregation that splits the execution pipeline into four different parts, one for each carinal point, using a **\$facet** command, each pipeline then proceeds to execute the following operations:

- a **\$match** on the desired squares
- a **\$sort** by the coordinate relative to the cardinal point considered by the pipeline.
- a **\$limit** that keeps only the document with the highest or lowest values.
- a **\$project** that keeps only the relevant fields of the documents

3.6.3. Query Text

```
db.logainm.aggregate([
  {
    $facet: {
      "northmostPoint": [
        {
          $match: {
            "gridReferences.square": {
              $in: ["A", "B", "C", "D", "E"]
            }
          }
        },
        {
          $sort: {
```

```

        "gridReferences.northing": -1
    }
},
{
    $limit: 1
},
{
    $project: {
        _id: 0,
        id: 1,
        name: "$placenames.wording",
        coordinates: "$gridReferences"
    }
}
],
"westmostPoint": [
    {
        $match: {
            "gridReferences.square": {
                $in: ["A", "F", "L", "Q", "V"]
            }
        }
    },
    {
        $sort: {
            "gridReferences.easting": 1
        }
    },
    {
        $limit: 1
    },
    {
        $project: {
            _id: 0,
            id: 1,
            name: "$placenames.wording",
            coordinates: "$gridReferences"
        }
    }
]

```



```

    }
  }
],
"southmostPoint": [
  {
    $match: {
      "gridReferences.square": {
        $in: ["V", "W", "X", "Y", "Z"]
      }
    }
  },
  {
    $sort: {
      "gridReferences.northing": 1
    }
  },
  {
    $limit: 1
  },
  {
    $project: {
      _id: 0,
      id: 1,
      name: "$placenames wording",
      coordinates: "$gridReferences"
    }
  }
],
"eastmostPoint": [
  {
    $match: {
      "gridReferences.square": {
        $in: ["D", "J", "O", "T", "Y"]
      }
    }
  },
  {

```

```

    $sort: {
      "gridReferences.easting": -1
    },
    {
      $limit: 1
    },
    {
      $project: {
        _id: 0,
        id: 1,
        name: "$placenames.wording",
        coordinates: "$gridReferences"
      }
    }
  ]
}
})

```

3.6.4. Results

The resulting places are listed below, their id can also be used in the *logainm.ie* website to check their position:

```

"northmostPoint": [
  {
    "id": 14546,
    "name": [
      "Inis Trá Tholl",
      "Inishtrahull"
    ],
    "coordinates": [
      {
        "square": "C",
        "easting": 48466,
        "northing": 65341
      }
    ]
  }
],

```



Figure 3.6: The northmost point (highlighted in blue)

```

"westmostPoint": [
  {
    "id": 1394425,
    "name": [
      "An Feo Láir",
      "Foze Rock Middle"
    ],
    "coordinates": [
      {
        "square": "V",
        "easting": 15332,
        "northing": 89121
      }
    ]
  }
],

```

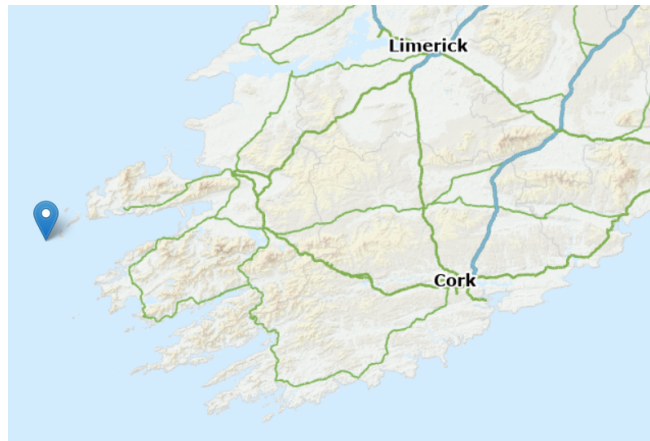


Figure 3.7: The westmost point (highlighted by the marker)

```

"southmostPoint": [
  {
    "id": 1165639,
    "name": [
      "Carraig Aonair",
      "Fastnet Rock"
    ],
    "coordinates": [
      {
        "square": "V",
        "easting": 88035,
        "northing": 15946
      }
    ]
  }
],

```

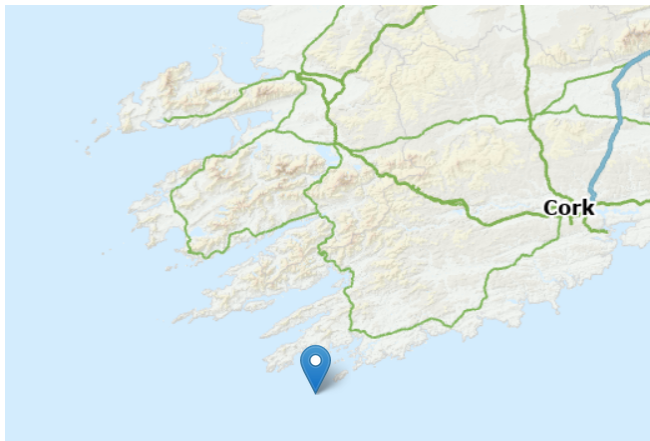


Figure 3.8: The southmost point (highlighted by the marker)

```

"eastmostPoint": [
  {
    "id": 1411807,
    "name": [
      "Port an Bhogaigh",
      "Portavogie"
    ],
    "coordinates": [
      {
        "square": "J",
        "easting": 66070,
        "northing": 58990
      }
    ]
  }
],

```

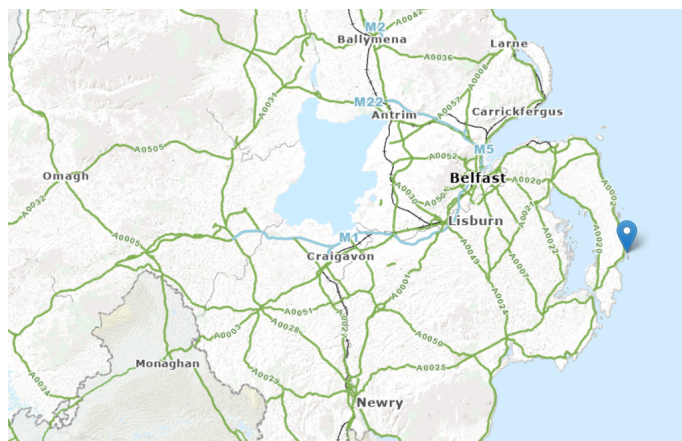


Figure 3.9: The westmost point (highlighted by the marker)

3.7. City or town with most streets per county

3.7.1. Preliminary Information

The *logainm.ie* dataset also documents the main streets and roads that traverse the main cities of Ireland. Streets are treated as other places and belong to the "street" category. An interesting analysis that can be done with a query is to group the streets by the cities they are included in and see which ones are the most represented for each county.

3.7.2. Query Explanation

The query is an aggregation composed of 10 stages:

- an **\$unwind** operation on the categories field, to take into account the rare cases in which the field has more than one entry.
- a **\$match** that filters the document on two different conditions:
 - if the category of the document is of type "street" (id = "SR", from *sráid*).
 - if the includedIn field has an entry for town (id = "B", from *baile*) or city (id = "CTH", from *cathair*)
- a **\$project** that creates two new fields, one for the county and one for the city/town in which the street is included. These fields are created by filtering the **includedIn** arrays of each document with a **\$filter**, keeping only the data about the names of the county or city/town.
- two **\$unwind** stages on the newly created fields, this step is necessary because a street can be included in more than one county or city. By unwinding those fields we separate the different county/city names.
- a **\$group** by county and city/town that counts the documents belonging to each group.
- a **\$sort** by the count field (in descending order), this is necessary to prepare the documents for the next stage.
- another **\$group** stage, this time only by county, that for each group takes the first element in the array of grouped documents, the order of the documents has been defined in the previous step, so the first sub-document will be the city/town with the most streets.

- a **\$project** that organizes the relevant data in the resulting documents.
- a **\$sort** by the name of the counties in English.

3.7.3. Query Text

```
db.logainm.aggregate([
  {
    $unwind: "$categories"
  },
  {
    $match: {
      "categories.id": "SR",
      "includedIn.category.id": {
        $in: ["B", "CTH"]
      }
    }
  },
  {
    $project: {
      id: "$id",
      county: {
        $filter: {
          input: "$includedIn",
          as: "place",
          cond: {
            $eq: ["$$place.category.id", "CON"]
          }
        }
      }
    },
    city_town: {
      $filter: {
        input: "$includedIn",
        as: "place",
        cond: {
          $or: [
            {

```

```

        $eq: ["$$place.category.id", "B"]
      },
      {
        $eq: ["$$place.category.id", "CTH"]
      }
    ]
  }
}
}
},
{
  $unwind: "$county"
},
{
  $unwind: "$city_town"
},
{
  $group: {
    _id: {
      county: {
        nameGA: "$county.nameGA",
        nameEN: "$county.nameEN"
      },
      city_town: {
        nameGA: "$city_town.nameGA",
        nameEN: "$city_town.nameEN"
      }
    },
    count: {
      $sum: 1
    }
  }
},
{
  $sort: {
    "count": -1
  }
}

```

```

    }
  },
  {
    $group: {
      _id: "$_id.county",
      city_town: {
        $first: "$$ROOT"
      }
    }
  },
  {
    {
      $project: {
        _id: 0,
        county: "$_id",
        city_town: "$city_town._id.city_town",
        count: "$city_town.count"
      }
    },
    {
      $sort: {
        "county.nameEN": 1
      }
    }
  }
)]

```

3.7.4. Results

A sample of the resulting document is provided below:

```

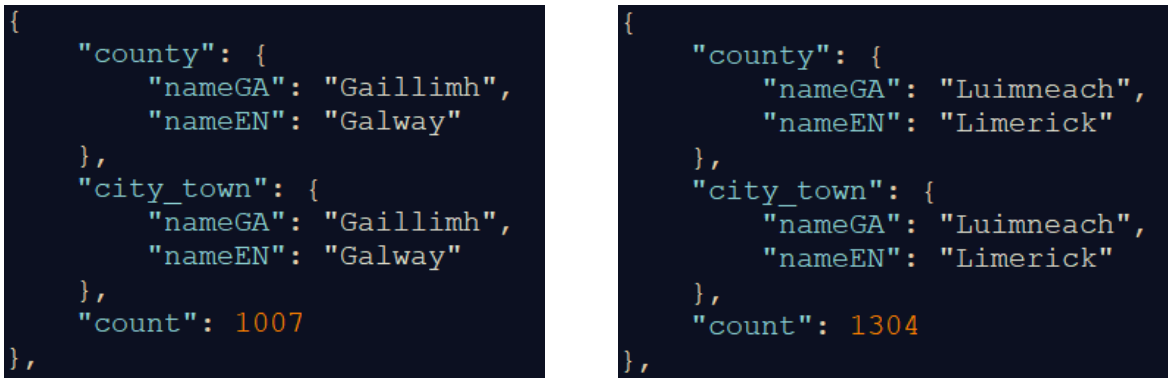
{
  "county": {
    "nameGA": "Corcaigh",
    "nameEN": "Cork"
  },
  "city_town": {
    "nameGA": "Corcaigh",
    "nameEN": "Cork"
  },
  "count": 3170
},

```

```

{
  "county": {
    "nameGA": "Baile Átha Cliath",
    "nameEN": "Dublin"
  },
  "city_town": {
    "nameGA": "Baile Átha Cliath",
    "nameEN": "Dublin"
  },
  "count": 4965
},

```



```
{
  "county": {
    "nameGA": "Gaillimh",
    "nameEN": "Galway"
  },
  "city_town": {
    "nameGA": "Gaillimh",
    "nameEN": "Galway"
  },
  "count": 1007
},
```

```
{
  "county": {
    "nameGA": "Luimneach",
    "nameEN": "Limerick"
  },
  "city_town": {
    "nameGA": "Luimneach",
    "nameEN": "Limerick"
  },
  "count": 1304
},
```

Figure 3.10: Results of query 7

As expected, the cities with the most documented streets are the most important ones. From this query we can also see how Northern Ireland is under-represented in the dataset since the city of Belfast has only 4 streets registered; this is due to the fact that the dataset is property of the Republic of Ireland's government.

3.8. Longest placenames

3.8.1. Preliminary Information

Returning to the matter of placenames, another query that may produce insightful results is to get the longest Irish placenames of towns and cities present in the dataset and analyse their meaning.

3.8.2. Query Explanation

The query is fairly simple and is made up of 5 stages:

- a **\$match** that filters the documents to keep only those that are cities or towns.
- a **\$project** that extracts the English and Irish placenames from the document and puts them in a dedicated attribute to them.
- a second **\$project** that takes the Irish name of each document and computes its length with the **\$strLenCP** function.
- a **\$sort** by the length attribute in descending order.
- a **\$limit** that keeps only the ten longest name.

3.8.3. Query Text

```

db.logainm.aggregate([
  {
    $match: {
      "placenames.language": "ga",
      "categories.id": {
        $in: ["CTH", "B"]
      }
    }
  },
  {
    $project: {
      _id: 0,
      id: 1,
      nameGA: {
        $first: {
          $filter: {
            input: "$placenames",
            as: "name",
            cond: {
              $eq: ["$$name.language", "ga"]
            }
          }
        }
      },
      nameEN: {
        $first: {
          $filter: {
            input: "$placenames",
            as: "name",
            cond: {
              $eq: ["$$name.language", "en"]
            }
          }
        }
      }
    }
  }
])

```

```

    }
  },
  {
    $project: {
      _id: 0,
      id: 1,
      nameGA: "$nameGA.wording",
      nameEN: "$nameEN.wording",
      length: {
        $strLenCP: "$nameGA.wording"
      }
    }
  },
  {
    $sort: {
      length: -1
    }
  },
  {
    $limit: 10
  }
])

```

3.8.4. Results

The ten longest names on the dataset are displayed on the next page. From the results, we can see that the longest town name, with 29 total characters, is *Droichead Abhann Ó gCearnaigh*, which means "The bridge of the river of *Uí Cearnaigh*", where *Uí Cearnaigh* is a clan name meaning "the descendants of Cearnach" and *Ó gCearnaigh* its genitive case; this townland is also traversed by *Abhainn Ó gCearnaigh* ("the river of *Uí Cearnaigh*") and has a bridge that unites the two sides of the stream. Another interesting name is *Baile Chaisleáin Bhéarra*, which means "The castle-town of Béarra", in this case, the word "Béarra" refers to the peninsula of Béarra, where this townland is located.

```

{
  "id": 1416537,
  "nameGA": "Droichead Abhann Ó gCearnaigh",
  "nameEN": "Six-mile-bridge",
  "length": 29
},
{
  "id": 1416575,
  "nameGA": "Baile Chaisleáin an Róistigh",
  "nameEN": "Castletownroche",
  "length": 28
},
{
  "id": 1403359,
  "nameGA": "Modh-Shráidbhaile na Druipsí",
  "nameEN": "Dripsey Model Village",
  "length": 28
},
{
  "id": 4507,
  "nameGA": "Droichead an Bhuitléaraigh",
  "nameEN": "Butler's Bridge",
  "length": 26
},
{
  "id": 1413969,
  "nameGA": "Droichead Chaisleán Loiste",
  "nameEN": "Rochfortbridge",
  "length": 26
},
},

```

```

{
  "id": 1416577,
  "nameGA": "Baile an Teampaill Theas",
  "nameEN": "Churchtown",
  "length": 24
},
{
  "id": 1416574,
  "nameGA": "Baile Chaisleáin Bhéarra",
  "nameEN": "Castletownbere",
  "length": 24
},
{
  "id": 1421794,
  "nameGA": "Buirg Bhaile Átha Cliath",
  "nameEN": "Dublin Borough",
  "length": 24
},
{
  "id": 11245,
  "nameGA": "Béal Átha an Ghaorthaidh",
  "nameEN": "Ballingeary",
  "length": 24
},
{
  "id": 1416619,
  "nameGA": "Mainéar Uí Chuinneagáin",
  "nameEN": "Manorcunningham",
  "length": 23
},
}

```

Figure 3.11: Results of query 8

3.9. Gaeltacht frequency by county

3.9.1. Preliminary Information

Although from the 18th century onwards the Irish language has progressively lost its status as the primary language used in everyday life in Ireland, it is recognized as the first official language of the Republic of Ireland, which recognizes certain areas of territory as *Gaeltacht*, regions in which Irish is still used on a daily basis. Assessing the distribution of these areas by using the *logainm.ie* dataset is an easy task: in the dataset, each document representing a place inside a *Gaeltacht* is marked with a dedicated attribute that states its total or partial belonging to such an area.

3.9.2. Query Explanation

The aggregation execution pipeline of the query is:

- a **\$match** stage that keeps only the documents referring to townlands (the smallest possible administrative division, `id = "BF"`, from *baile fearainn*) and that are completely part of a *Gaeltacht* region (the **extent** attribute of the **gaeltacht** field

must be equal to "all").

- an **\$unwind** of the **includedIn** field, to separate the different elements of the array.
- once **includedIn** has been unwinded, the documents are filtered with another **\$match**, to keep only the ones that reference counties in the **includedIn** field.
- a **\$group** by the county attribute that counts the documents in each group.
- a last **\$project** to adjust the names of the fields in the output documents.

3.9.3. Query Text

```
db.logainm.aggregate([
  {
    $match: {
      "categories.id": "BF",
      "gaeltacht.extent": "all"
    }
  },
  {
    $unwind: "$includedIn"
  },
  {
    $match: {
      "includedIn.category.id": "CON"
    }
  },
  {
    $group: {
      _id: {
        nameEN: "$includedIn.nameEN",
        nameGA: "$includedIn.nameGA"
      },
      count: {
        $sum: 1
      }
    }
  },
],
```

```

    {
      $project: {
        _id: 0,
        county: "$_id",
        count: "$count"
      }
    }
  ]
})

```

3.9.4. Results

The resulting documents are the following:

```

{
  "county": {
    "nameEN": "Waterford",
    "nameGA": "Port Láirge"
  },
  "count": 67
},
{
  "county": {
    "nameEN": "Galway",
    "nameGA": "Gaillimh"
  },
  "count": 556
},
{
  "county": {
    "nameEN": "Cork",
    "nameGA": "Corcaigh"
  },
  "count": 159
},
{
  "county": {
    "nameEN": "Mayo",
    "nameGA": "Maigh Eo"
  },
  "count": 266
},

```

```

{
  "county": {
    "nameEN": "Donegal",
    "nameGA": "Dún na nGall"
  },
  "count": 589
},
{
  "county": {
    "nameEN": "Kerry",
    "nameGA": "Ciarraí"
  },
  "count": 394
},
{
  "county": {
    "nameEN": "Meath",
    "nameGA": "An Mhí"
  },
  "count": 14
}

```

Figure 3.12: Results of query 9

This data can also be shown using a map:

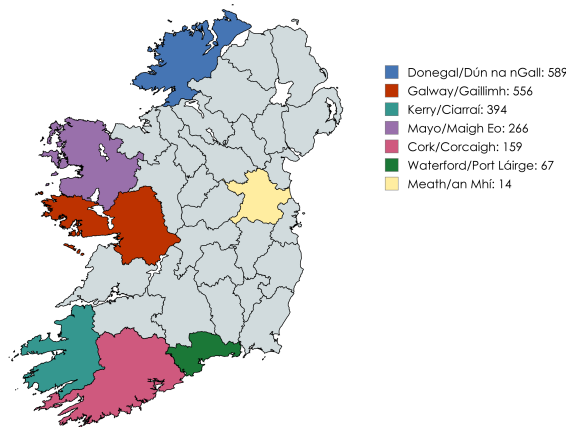


Figure 3.13: Graphical visualization of Query 9 results

As it can be seen from this last representation of the data, the *Gaeltacht* regions are mostly located on the western coast of Ireland, with the exception of Co. Meath. This is due to the proximity of the eastern coast of Ireland to England, which heavily influenced the use of the Irish language during the last few centuries.

3.10. Irish surnames by prefix

3.10.1. Preliminary Information

Each place in the *logainm.ie* dataset also includes a field dedicated to people born in that place that are part of the *ainm.ie* database, with 1286 total names included. A common occurrence in Irish surnames is the use of prefixes, which generally indicate some kind of relation to another component of the family; the most common ones are *Mac* (son of), *Ó* (grandson/descendant of), *Ní/Nic* (daughter of), *Uí/Mhic* (wife of) and *de* (of). A possible query on this kind of data would be to group all the names by their surname prefix (if they have one) and see which are the most used among the ones present in the dataset.

3.10.2. Query Explanation

The query is articulated in an 8-stage pipeline:

- an **\$unwind** of the **bornHere** field, to dedicate exactly one document to each name.
- a **\$project** that takes each name and computes both the complete surname and the

prefix using the **\$split** function. Since the names are provided in the form "*prefix rest of the surname, name*", the complete surname is extracted by splitting on the comma, while the prefix can be found by splitting on a white space.

- a **\$match** on the most common prefixes, to exclude the ones which are not adequately represented.
- a first **\$group** by the surname, in order to count the occurrences of each surname.
- a **\$sort** of the surnames by count, this ordering is needed to prepare the documents for the next stage.
- a second **\$group**, this time by prefix, that for each surname pushes an element in a new array field. Thanks to the previous stage, the documents will be inserted in the array in descending order of count.
- a **\$project** that selects only the useful field of the documents and that, for each prefix, keeps only the five most common surnames having at least two entries; this result is achieved by using a combination of **\$firstN** and **\$filter**.
- a **\$sort** of the prefixes by their total number of occurrences.

3.10.3. Query Text

```
db.logainm.aggregate([
  {
    $unwind: "$bornHere"
  },
  {
    $project: {
      person: "$bornHere.title",
      first_word: {
        $first: {
          $split: [
            "$bornHere.title",
            " "
          ]
        }
      },
      surname: {
```

```

        $first: {
            $split: [
                "$bornHere.title",
                ","
            ]
        }
    }
},
{
    $match: {
        "first_word": {
            $in: ["MAC", "NÍ", "Ó", "UÍ", "DE", "NIC", "MHIC"]
        }
    }
},
{
    $group: {
        _id: {
            prefix: "$first_word",
            surname: "$surname"
        },
        count: {
            $sum: 1
        }
    }
},
{
    $sort: {
        "count": -1
    }
},
{
    $group: {
        _id: "$_id.prefix",
        count: {
            $sum: "$count"
        }
    }
}

```



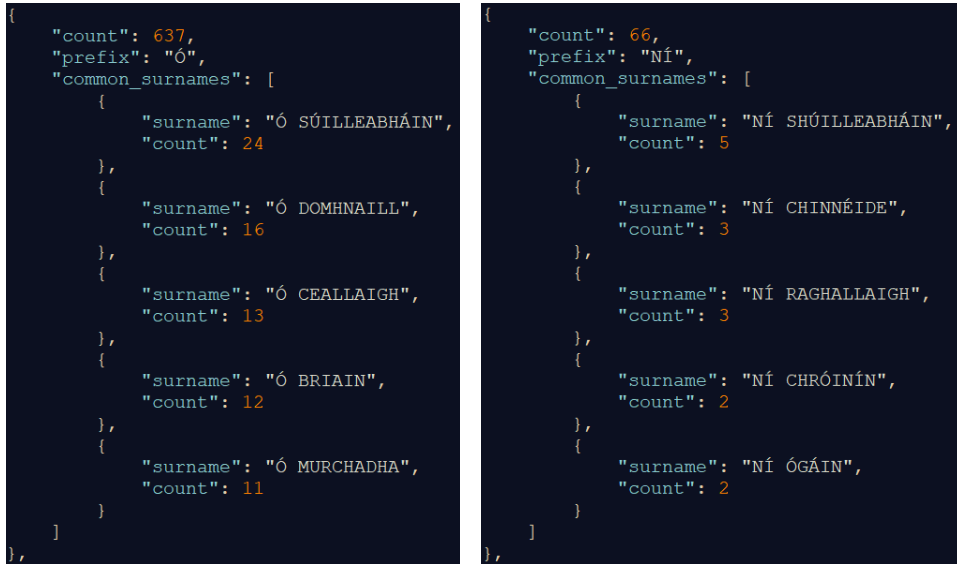
```

    },
    common_surnames: {
      $push: {
        surname: "$_id.surname",
        count: "$count"
      }
    }
  }
},
{
  $project: {
    _id: 0,
    prefix: "$_id",
    count: 1,
    common_surnames: {
      $firstN: {
        input: {
          $filter: {
            input: "$common_surnames",
            as: "surname",
            cond: {
              $gt: ["$$surname.count", 1]
            }
          }
        },
        n: 5
      }
    }
  }
},
{
  $sort: {
    "count": -1
  }
}
])

```

3.10.4. Results

Some of the returned documents are shown below:



```
{
  "count": 637,
  "prefix": "Ó",
  "common_surnames": [
    {
      "surname": "Ó SÚILLEABHÁIN",
      "count": 24
    },
    {
      "surname": "Ó DOMHNAILL",
      "count": 16
    },
    {
      "surname": "Ó CEALLAIGH",
      "count": 13
    },
    {
      "surname": "Ó BRIAIN",
      "count": 12
    },
    {
      "surname": "Ó MURCHADHA",
      "count": 11
    }
  ]
},
```

```
{
  "count": 66,
  "prefix": "Ní",
  "common_surnames": [
    {
      "surname": "Ní SHÚILLEABHÁIN",
      "count": 5
    },
    {
      "surname": "Ní CHINNÉIDE",
      "count": 3
    },
    {
      "surname": "Ní RAGHALLAIGH",
      "count": 3
    },
    {
      "surname": "Ní CHRÓINÍN",
      "count": 2
    },
    {
      "surname": "Ní ÓGÁIN",
      "count": 2
    }
  ]
},
```

Figure 3.14: Results of query 10

From these images, we can see that the most common masculine and feminine prefixes are, respectively, *Ó* and *Ní*; furthermore, for both of them, the most common surname is made by concatenating the prefix with the word *Súilleabháin* (it becomes *Shúilleabháin* in the feminine version). The etymology of the *Ó Súilleabháin* (anglicized as O' Sullivan) surname is uncertain, it may derive from *súildubhán*, meaning "descendant of the little dark-eyed one" (*súil* -> "eye", *dubh* -> "black", *-án* -> diminutive suffix), or "descendant of the one-eyed" (*súil* -> "eye", *abháin* (nowadays spelt as *amháin*) -> "one").

4 | Extra: Analysis using pymongo

This chapter provides some performance analysis of queries described in the previous chapter. The profiling is performed using the pymongo package for Python, by launching the queries with and without indexes and using the **explain** command with the **executionStats** verbosity; the result of the analysis are then analyzed, plotted with the **matplotlib** Python package and shown on this document, accompanied by some comment about the results.

4.1. Analysis of Query 2

4.1.1. Analysis Explanation

In order to assess the effectiveness of an index, a simple analysis that can be performed is the profiling of Query 2 ("Names and physical features: Lakes"). After connecting to MongoDB with pymongo, we can create the explain command using a dictionary representation:

```
query_stats = db.logainm.find(
    {
        "includes.id": "L",
        "glossary.headword": "loch"
    },
    {
        "_id": 0,
        "id": 1,
        "placenames.language": 1,
        "placenames.wording": 1,
        "categories.nameEN": 1
    }
).explain()["executionStats"]
```

This command automatically runs the query and produces the explanation in output. After the first run, we can create the indexes on the attributes that the query uses to match the documents.

```
db.logainm.drop_index("includes.id_1")
db.logainm.drop_index("glossary.headword_1")
```

Then we can re-run the previous command, drop the indexes, save the output of both explain queries to different files and proceed with the analysis. In particular, we need the total execution times of the two different queries, this information can be found under the **"executionTimeMillis"** attribute in the JSON files resulting from the explanations.

```
print("Execution time without indexes: " +
      str(query_stats["executionTimeMillis"]) + " ms")
print("Execution time with indexes: " +
      str(query_stats_idx["executionTimeMillis"]) + " ms")
```

4.1.2. Results

The output of the script is:

```
Execution time without indexes: 855 ms
Execution time with indexes: 31 ms
```

In fact, if we look closely at the output files, we can see that the attribute **"executionStages.inputStage.inputStage.stage"** is **"COLLSCAN"** when the query does not use an index and **"IXSCAN"** when the query is using the indexes defined in the program. The output files can also be examined to check how MongoDB executes the query in both cases: on the next page, the image on the left (4.1) reports the execution plan when no index is used: in this case, the operation is a **"COLLSCAN"** that filters the documents using the conditions specified in the query. On the other hand, the right image (4.1) represents the execution plan while using the indexes: the first stage is an **"IXSCAN"** that uses only the index on the **includes.id** (4.2) attribute, the second operation is a **"FETCH"** that filters on the other attribute. In the output file for the query with the indexes are also present the other execution plans that were discarded by MongoDB (i.e. one that uses both indexes and one that uses only the **categories.headword** index) because considered not as efficient as the one used.

```

"inputStage": {
  "stage": "COLLSCAN",
  "filter": {
    "$and": [
      {
        "glossary.headword": {
          "$eq": "loch"
        }
      },
      {
        "includes.id": {
          "$eq": "L"
        }
      }
    ]
  },
  "nReturned": 53,
  "executionTimeMillisEstimate": 835,
  "works": 126753,
  "advanced": 53,
  "needTime": 126699,
  "needYield": 0,
  "saveState": 52,
  "restoreState": 52,
  "isEOF": 1,
  "direction": "forward",
  "docsExamined": 126752
},
"inputStage": {
  "stage": "FETCH",
  "filter": {
    "glossary.headword": {
      "$eq": "loch"
    }
  },
  "nReturned": 53,
  "executionTimeMillisEstimate": 10,
  "works": 1864,
  "advanced": 53,
  "needTime": 1809,
  "needYield": 0,
  "saveState": 2,
  "restoreState": 2,
  "isEOF": 1,
  "docsExamined": 1862,
  "alreadyHasObj": 0,
  "inputStage": {
    "stage": "IXSCAN",
    "nReturned": 1862,
    "executionTimeMillisEstimate": 0,
    "works": 1863,
    "advanced": 1862,
    "needTime": 0,

```

Figure 4.1: Execution stages of Query 2 in both cases

```

"inputStage": {
  "stage": "IXSCAN",
  "nReturned": 1862,
  "executionTimeMillisEstimate": 0,
  "works": 1863,
  "advanced": 1862,
  "needTime": 0,
  "needYield": 0,
  "saveState": 2,
  "restoreState": 2,
  "isEOF": 1,
  "keyPattern": {
    "includes.id": 1
  },
  "indexName": "includes.id_1",
  "isMultiKey": true,
  "multiKeyPaths": {
    "includes.id": [
      "includes"
    ]
  },
  "isUnique": false,
  "isSparse": false,
  "isPartial": false,
  "indexVersion": 2,
  "direction": "forward",

```

Figure 4.2: IXSCAN parameters

4.2. Analysis of Query 9

4.2.1. Analysis Explanation

To complicate things we can do a similar analysis on an aggregation query, in this way we can also display the partial execution times, to see which ones are the most demanding and how indexes behave when the query is made up of more than a single stage. A suitable query for this job is Query 9 (Gaeltacht frequency by county), since it provides a **\$match** at the start of the pipeline, which is an ideal condition when dealing with indexes on aggregation queries.

After connecting to the database and providing the execution pipeline in dictionary notation, we can launch the explanation without indexes with the following commands:

```
gaeltacht_query_pipeline = ...
query_stats_noIdx = db.command("explain", {
    "aggregate": "logainm",
    "pipeline": gaeltacht_query_pipeline,
    "cursor": {}
}, verbosity="executionStats")
```

We then repeat the process three more times: one with only the **categories.id** index, one with only the **gaeltacht.extent** index and one with both indexes, save the JSON output on file and plot the partial execution times contained in it.

4.2.2. Results

The images below show the execution time of the query in all four cases and the number of documents returned by each stage of the pipeline.

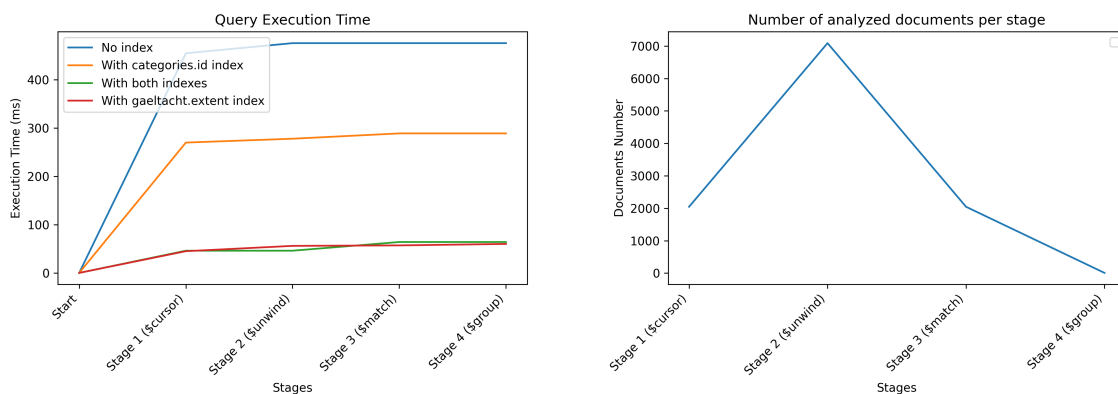


Figure 4.3: Execution times and returned documents of Query 9

From the left image, it can be seen that the query without indexes takes far longer to execute with respect to the other cases. We can also appreciate how the query with both indexes and the query with only the **gaeltacht.extent** index takes approximately the same time, this happens because, like in the previous analysis, only one index is used by the query, even if two are available, with the most efficient one being chosen by the planner. The use of the other index is also advantageous and is effective in reducing the execution time of the query, but it's not the most efficient. Another interesting consideration is the fact that, as one could expect, the first stage of the query (the **\$match/\$cursor**) is the most expensive one, given the huge amount of documents present in the dataset; we can also see, from the right picture, how the first stage alone is responsible for the elimination of the majority of the documents, leaving only 2045 left from the starting 126752. The other stages of the pipeline also behave as expected, with the **\$unwind** slightly raising the number of documents and the other two stages decreasing it.

4.3. Analysis of Query 3

4.3.1. Analysis Explanation

Even if indexes may seem to be able to reduce the execution time of a query regardless of the way they are used, if not used properly, they may not get considered at all by the planner or even worsen the performance of a query. A suitable example to show the problems in the misuse of an index is Query 3 (Placenames and Folklore: the *Púca*), which has two **\$regex** operations in the first **\$match** stage. **\$regex** operations are problematic with indexes because they are not exact matches, but can still be efficient if the regex match is anchored at the start of the string, which, however, is not the case for this query. To perform this analysis we proceed in the same way as Analysis 2: we create the pipeline, run it without an index, create an index on **placenames.wording**, and perform another run of the query; the results are then saved on JSON files and plotted using the matplotlib package.

4.3.2. Results

The result obtained by running the program is:

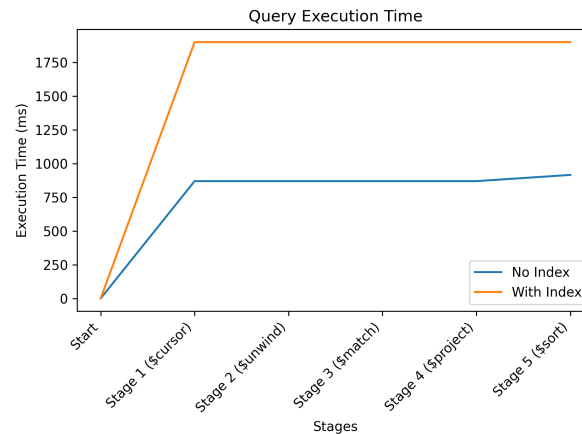


Figure 4.4: Caption

As can be seen from the image, the execution time when using an index is far worse than when performing the query without any indexing. By looking at the JSON result of the analysis we can see that the query uses the index that was created, but the execution of the first stage takes just more time, since, as previously said, **\$regex** operations are not suitable to be used with indexes, as they do not exactly match the value of a field.

```
"winningPlan": {
  "isCached": false,
  "stage": "PROJECTION_DEFAULT",
  "transformBy": {
    "county": 1,
    "id": 1,
    "includedIn": 1,
    "placenames.language": 1,
    "placenames.wording": 1,
    "_id": 0
  },
  "inputStage": {
    "stage": "COLLSCAN",
    "filter": {
      "placenames.wording": {
        "$in": [
          {
            "$regexExpression": {
              "pattern": "Phúca",
              "options": ""
            }
          },
          {
            "$regexExpression": {
              "pattern": "Púca",
              "options": ""
            }
          }
        ]
      }
    }
  },
  "direction": "forward"
},
```

```
"winningPlan": {
  "isCached": false,
  "stage": "PROJECTION_DEFAULT",
  "transformBy": {
    "county": 1,
    "id": 1,
    "includedIn": 1,
    "placenames.language": 1,
    "placenames.wording": 1,
    "_id": 0
  },
  "inputStage": {
    "stage": "FETCH",
    "filter": {
      "placenames.wording": {
        "$in": [
          {
            "$regexExpression": {
              "pattern": "Phúca",
              "options": ""
            }
          },
          {
            "$regexExpression": {
              "pattern": "Púca",
              "options": ""
            }
          }
        ]
      }
    }
  },
  "inputStage": {
    "stage": "IXSCAN",
    "keyPattern": {
      "placenames.wording": 1
    },
    "indexName": "placenames.wording_1",
```

Figure 4.5: Execution stages of Query 3 in both cases