Large-Scale and Multi-Structured Databases

Project Design

GastronoMate

Gemelli Mattia, Nocella Francesco, Petruzzella Christian

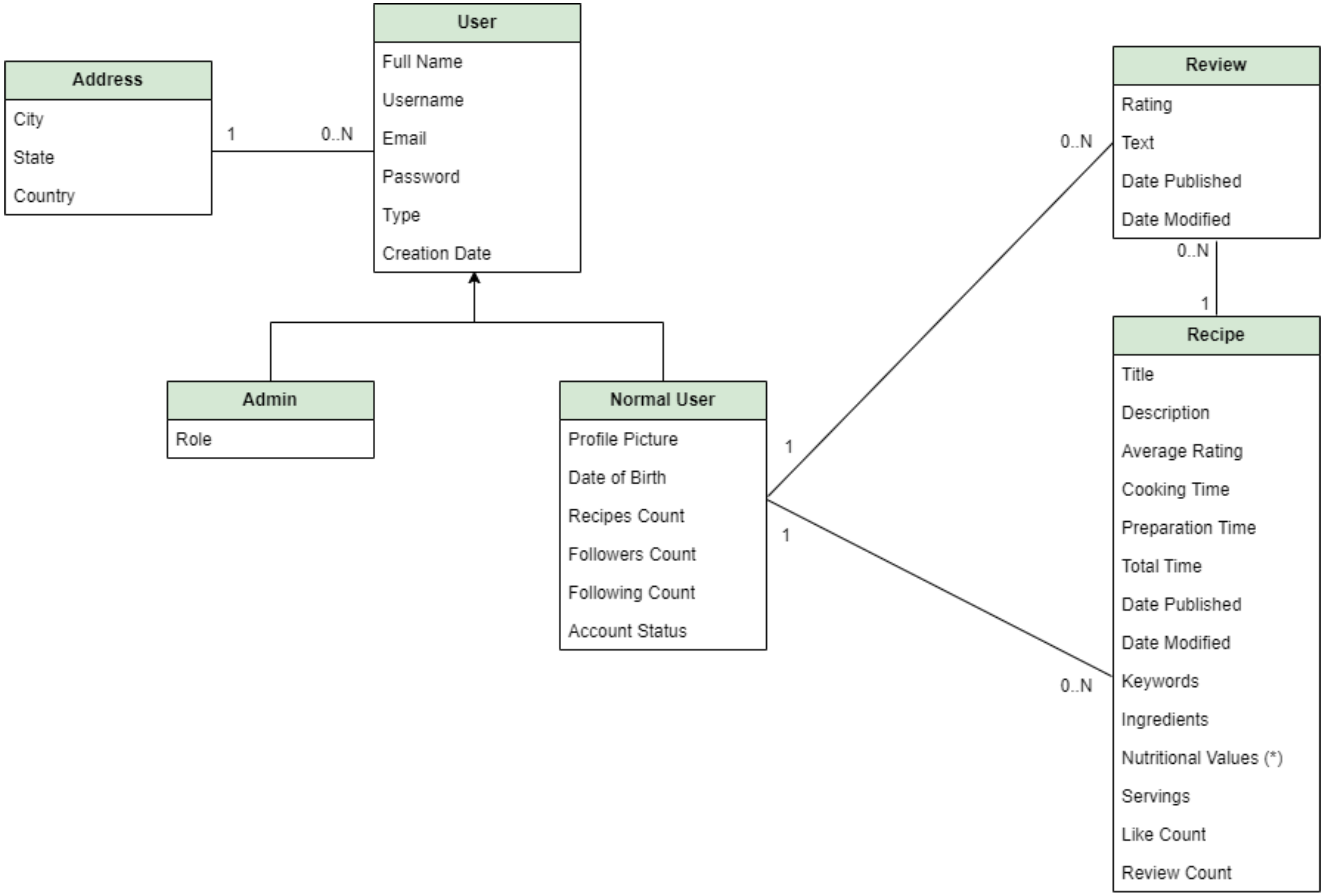
Application Highlights

GastronoMate is a social network with the aim of creating an interactive and involving community. The name of the application represents our vision for the project: connect friends using their own common passion for the gastronomy.

Main features:

- Search users by username.
- Follow and unfollow other users.
- View users' profile and recipes.
- Publish, update, and delete recipes.
- Like and review recipes.
- Receive suggestions based on user's and followers' preferences.
- Search recipes based on various criteria.
- Admin panel to explore analytics regarding users and recipes.

Analysis Class Diagram



DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

IN SUPREMÆ DIGNITATIS 1343 UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Non-Functional Requirements

1. The system must allow a stateless authentication system.
2. The system must be responsive.
3. The system must be capable of handling many concurrent users.
4. The system must be always available.
5. The system must securely store users' data.
6. The system must have an authentication and authorization mechanism to prevent unauthorized access.
7. The system must have a user-friendly interface.
8. The system must maintain data integrity to prevent data corruption or loss.
9. The database must be deployed on both local and virtual clusters with at least three replica sets, ensuring high availability through eventual consistency.
10. The system must manage consistency between different database architectures.

Dataset Description

Source: Cookbooks.com & Food.com

The recipes dataset shows about 500k recipes taken from two distinct sources, providing information on each recipe, including cooking times, servings, ingredients, nutritional details, instructions, and more. Moreover, some reviews about them are provided by users.

Volume: around 3,08 GB

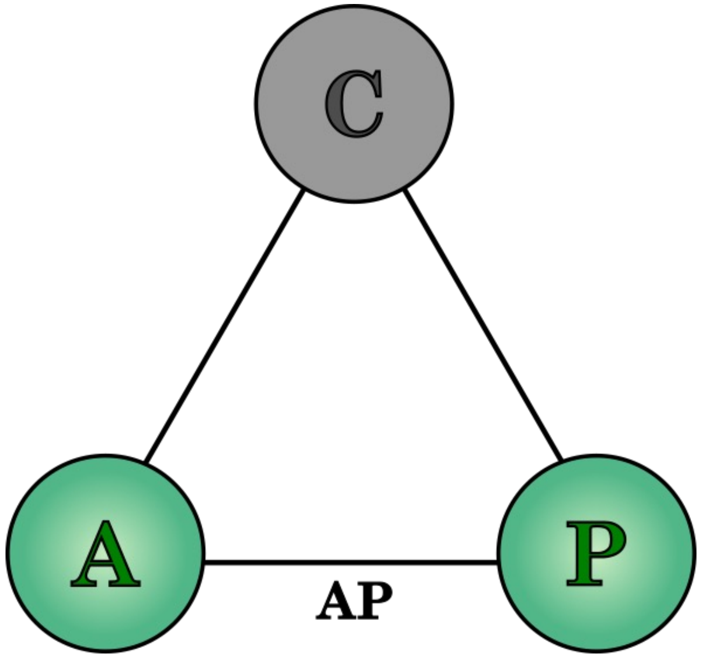
Variety:

- Multi-sources: Food.com & Cookbooks.com
- Multi-format: different attributes

Cleaning and preprocessing

Python with pandas has been used to perform several operations on the raw data, deleting the columns of the csv file we thought were not necessary. The missing information about users, email, address, full name has been added, and then the two data sources have been mixed to create the definitive database.

Availability and Partition Tolerance



GastronoMate, as a social network, prioritizes the AP configuration of the CAP theorem, ensuring Availability and Partition Tolerance. This is crucial as it allows users to access the application, even if they see different views of data.

This approach enhances user experience by prioritizing continuous service over data consistency.

Document DB

The system must handle a large amount of data with varying attributes in terms of number and type. This led to choose a Document Database, due to its flexibility, along with ease to use and high-performance capabilities, which allow to make complex queries, including advanced filtering on different types of attributes.

The following analytic queries have been designed:

- **Monthly subscriptions percentage** (provides the percentage of monthly subscriptions starting from a specified month to the end of the year).
- **Yearly Subscriptions** (provides the number of subscriptions per year within a specified year interval).
- **Users per State** (provides the number of users per location State).
- **Trending Keywords** (provides the most used keywords within a specified time interval).
- **Highest Scored Recipes** (provides a list of the recipes with the highest scores).

Document DB

User

```
{
 "_id": {
  "$oid": "65787f95d3100bd34861bcae"
 },
 "username": "Suzy Q 2",
 "Email": "jon.hendricks@protonmail.com",
 "Password": "L89C1a$b+a",
 "DateOfBirth": {
  "$date": "1974-05-22T00:00:00.000Z"
 },
 "Address": {
  "City": "Millerland",
  "State": "AK",
  "Country": "US"
 },
 "Recipes": [
  {
   "RecipeId": {
    "$oid": "6579b97f72d58fd14fd36e70"
   },
   "Title": "Chocolate Bonbons",
   "DatePublished": {
    "$date": "2018-01-17T01:31:20.000Z"
   }
  },
],
 "FullName": "Jon Hendricks"
}
```

Recipe

```
{
 "_id": {
  "$oid": "6579b97d72d58fd14fcf9bca"
 },
 "title": "Balsamic Maple Vinaigrette",
 "PrepTime": "5M",
 "TotalTime": "5M",
 "DatePublished": {
  "$date": "2007-01-01T14:41:00.000Z"
 },
 "Description": "Make and share this
Balsamic Maple Vinaigrette recipe from
Food.com.",
 "Keywords": [
  "< 15 Mins",
  "For Large Groups",
  "No Cook",
  "Beginner Cook",
  "Easy",
  "Salad Dressings"
 ],
 "RecipeServings": 16,
 "Likes": 87,
 "AuthorUsername": "Tarynne"
},
```

Review

```
[{
 "_id": {
  "$oid": "6571834e2244c888f74ae7d3"
 },
 "Rating": 4,
 "DateSubmitted": {
  "$date": "2011-06-01T17:48:25.000Z"
 },
 "DateModified": {
  "$date": "2011-06-01T17:48:25.000Z"
 },
 "Recipe": {
  "RecipeId": {
   "$oid": "6579b97e72d58fd14fd0a8bc"
  },
  "Title": "Italian Bean and Tuna Salad",
  "AuthorUsername": "JackieOhNo"
 },
 "Text": "Quick, light, refreshing salad. A
switch from the old standby tuna salad. And
it's pretty!<br/>made for ZWT7",
 "AuthorUsername": "Linky"
},
```

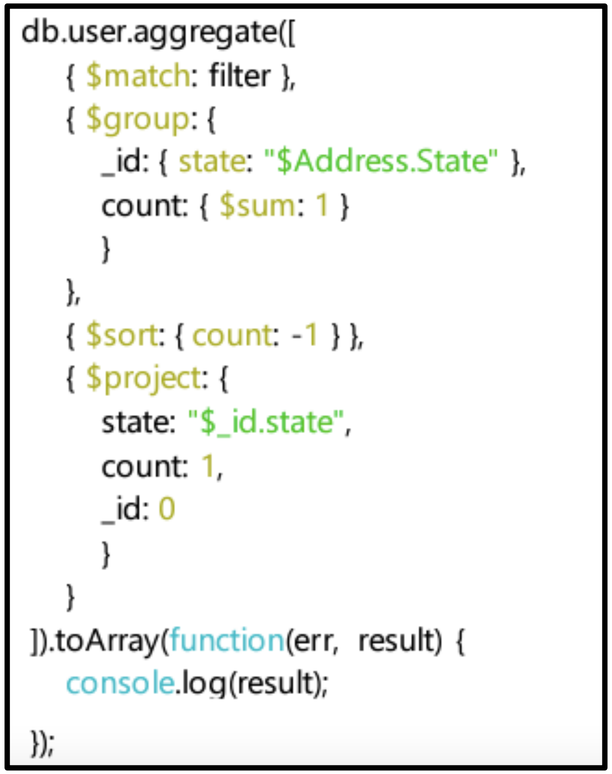
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Examples of Aggregations

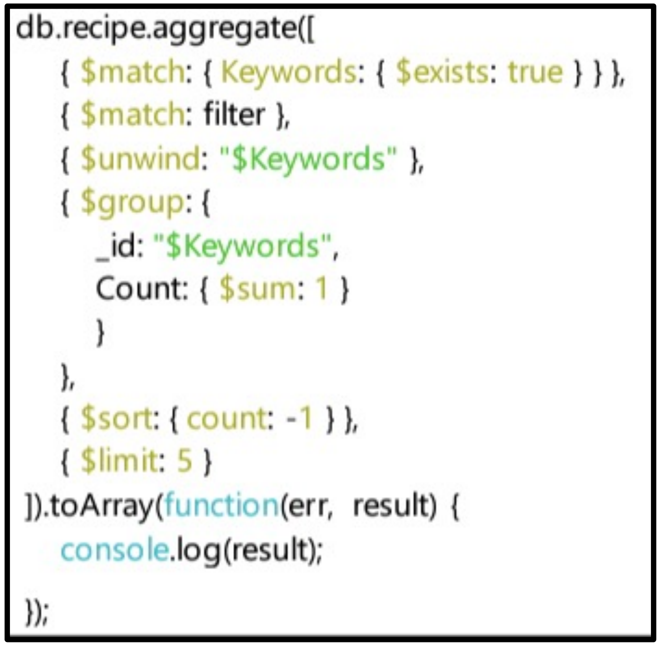
Users per State



```
db.user.aggregate([
    { $match: filter },
    { $group: {
        _id: { state: "$Address.State" },
        count: { $sum: 1 }
        }
    },
    { $sort: { count: -1 } },
    { $project: {
        state: "$_id.state",
        count: 1,
        _id: 0
        }
    }
]).toArray(function(err,  result) {
    console.log(result);

});
```

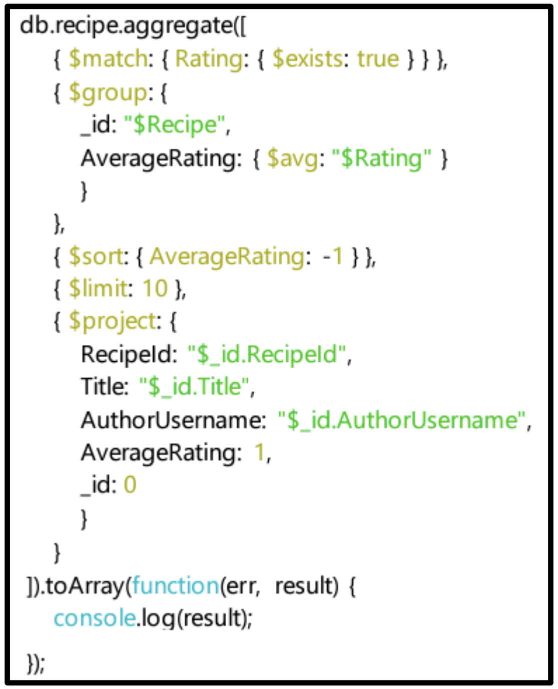
Trending Keywords



```
db.recipe.aggregate([
    { $match: { Keywords: { $exists: true } } },
    { $match: filter },
    { $unwind: "$Keywords" },
    { $group: {
        _id: "$Keywords",
        Count: { $sum: 1 }
        }
    },
    { $sort: { count: -1 } },
    { $limit: 5 }
]).toArray(function(err,  result) {
    console.log(result);

});
```

Highest Scored Recipes



```
db.recipe.aggregate([
    { $match: { Rating: { $exists: true } } },
    { $group: {
        _id: "$Recipe",
        AverageRating: { $avg: "$Rating" }
        }
    },
    { $sort: { AverageRating: -1 } },
    { $limit: 10 },
    { $project: {
        RecipeId: "$_id.RecipeId",
        Title: "$_id.Title",
        AuthorUsername: "$_id.AuthorUsername",
        AverageRating: 1,
        _id: 0
        }
    }
]).toArray(function(err,  result) {
    console.log(result);

});
```

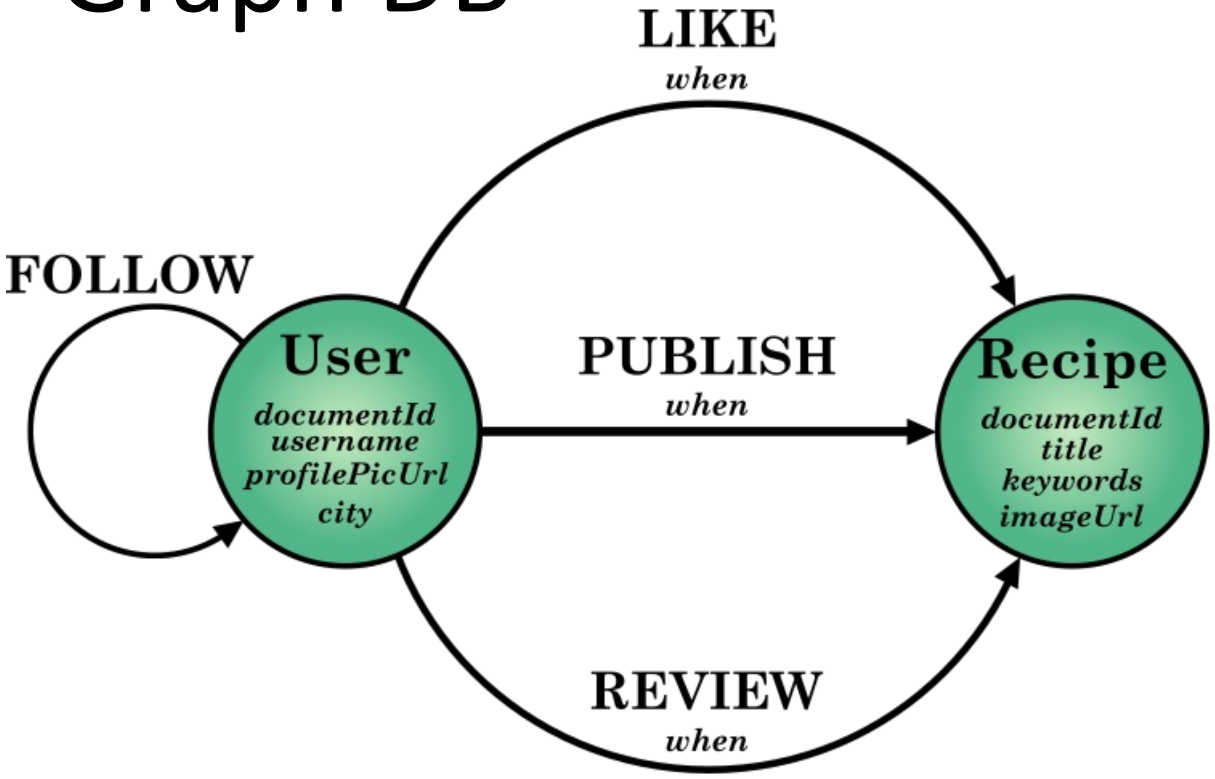
Graph DB

Social functionalities have been designed, which required the use of a Graph Database, to rapidly traverse the paths between entities and handle relations between different instances of entities (users, recipes, and reviews).

The following suggestions and analytic queries have been designed:

- **Suggest Users** (retrieves information about users who might be of interest to a specific user based on various criteria such as similar liking patterns, common followers, and mutual liking relationships).
- **Suggest Recipes** (retrieves information about recipes that have been liked by users whom a specific user is following, but he hasn't liked yet, showing only recipes created by authors whom the user is not following).
- **Most Influential Users** (provides a list of the most influential users, based on likes, number of followers, and posted reviews).
- **Most Liked Recipes** (provides a list of the most liked recipes of all time).

Graph DB



Nodes

- **User**: storing users with relative id, username, profile picture, and city.
- **Recipe**: storing recipes with relative id, title, images, keywords.

Relationships

- **LIKE**: edge between a user vertex and a recipe vertex.
- **FOLLOW**: edge between different user vertices.
- **PUBLISH**: edge between a recipe vertex and its author (user) vertex.
- **REVIEW**: edge between a recipe vertex and review's author (user) vertex.

Consistency

Real-time updates and data synchronization requires consistency handling. To achieve consistency between the databases, asynchronous tasks mechanisms have been implemented. By decoupling operations such as user interactions (*likes*, *review posting*) and content modification (*user information editing*, *recipe details updates*), at the same time data integrity have been ensured and the application responsiveness maintained. This approach allows to handle concurrent user actions without compromising consistency, as tasks are executed asynchronously, minimizing conflicts and latency.

The use of two databases in a concurrent manner entails careful management of the data in both. To maintain data consistency, initial operations involve MongoDB. Subsequently, asynchronous tasks are initialized to address redundancies in Neo4j. If errors occur during the asynchronous task execution, the system serialize it into a log for further analysis.

Sharding

Even if not implemented, a suitable Sharding strategy for the application could be achieved by:

- All the recipes associated to a user should be stored in the same shard.
- All the reviews associated to a recipe should be stored in the same shard.
- Data processed by complex aggregation queries, such as the application analytics, are spread among the shards to balance the computational load.

Collection	Field	Description
User	username	Considering that no particular queries are performed on the User collection, the Sharding is achieved due to the unicity property of the field, that allows to balance the distribution of data.
Recipe	user._id	Ensures that all documents about the recipes published by the same user will be stored in the same shard. The Sharding is achieved due to the randomicity of the field.
Review	recipe._id	Ensures that all documents about the reviews posted on the same recipe will be stored in the same shard. The Sharding is achieved due to the randomicity of the field.

Software Architecture

FRONT-END:
- Web app (JSP, CSS, JavaScript)

BACK-END:
- Spring Boot (Java)

DATABASES:
- MongoDB (document database)
- Neo4J (graph database)

Conclusion

Guided by users' feedback, GastronoMate remains committed to continuous improvement and innovation.

To explore the project development details, here is the URL of the GitHub repository: https://github.com/franocella/GastronoMate.

Gemelli Mattia

Nocella Francesco

Petruzzella Christian