

DIGIMON DATABASE

MongoDB



Lovely Fernandez C20305696
3rd Year Computer Science International TU858-3

Contents

Digimon Database	2
Design.....	2
Structure	3
Validation Rules.....	3
Attribute	3
Profile	4
Select Queries	4
Select All Documents	4
Select Array Data	4
Show Projection of Profile or Attribute.....	4
Sort by Attribute.....	4
Aggregation by Type.....	5
Data Manipulation	5
Insert	5
Update.....	5
Delete.....	5
Summary	5

Digimon Database

For this assignment, I will work on manipulating the dataset of Digimon characters, sourced from Kaggle's DigiDB (<https://www.kaggle.com/datasets/rtatman/digidb>).

The main collection will represent the variety of Digimon including their name, stage, type, attributes, memory, equip slots and stat abilities such as health, SP, attack, defence, INT and speed.

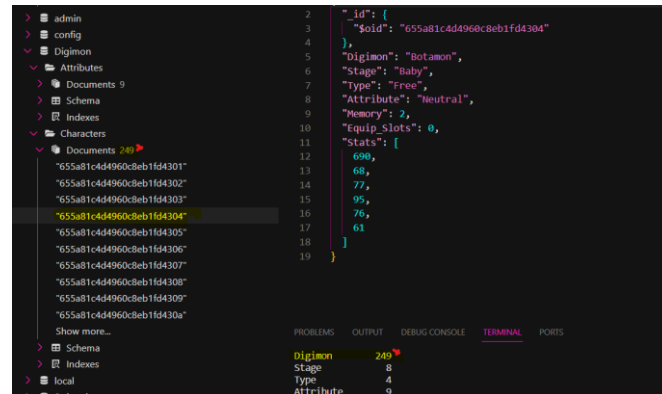


Figure 1: Character Collection of Digimon DB

My collection design will centre on exploring the different attributes associated with these characters. I will categorize the Digimon dataset, examining which characters fall into in to which attribute and will integrate a profile which includes the name, stage, and type. This will provide information of each Digimon and allows us to understand the variety of characters and their unique traits within this game.

Design

To represent my collection, I will filter all the Digimon characters under the attributes they fall into. There are 9 types of attributes; *dark, earth, electric, fire, light, neutral, plant, water, and wind*. These are fundamental characteristics that define the nature and abilities acquired of each Digimon. In addition, I will also include a profile information with the name, stage, and type of each Digimon under that attribute.

Categorising by attribute allows us to base our Digimon by their elemental trait. This is beneficial as pre-planning match ups may help players counter pick their opponent during a game. For example, wind is stronger against plant, while plant is stronger against earth. As a strategy, if your opponent tends to use Earth type characters, the optimum choice is to counter by Plan type characters.

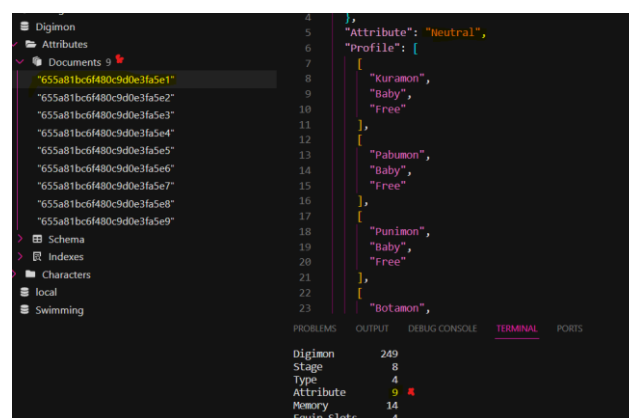


Figure 2: Attribute Collection of Characters from Digimon DB

Structure

The "profile" field is an embedded array containing character information for the Digimon, which includes their name, stage, and type. Using an array in this context allows the grouping of Digimon characters under the same attributes, which results to a cleaner and more organised database structure and preventing data duplications.

For each unique attribute, the data frame is filtered to include only the Digimon characters with that attribute. In Figure 5, there are two collections of the following Dark and Neutral attributes, displaying each Digimon with that trait.



```
{
  "_id": {
    "$oid": "655aa60c6823245d0d5511b7"
  },
  "Attribute": "Dark",
  "Profile": [
    {
      "Digimon": "Tsumemon",
      "Stage": "In-Training",
      "Type": "Free"
    },
    {
      "Digimon": "Pagumon",
      "Stage": "In-Training",
      "Type": "Free"
    },
    {
      "Digimon": "Impmon",
      "Stage": "Rookie",
      "Type": "Virus"
    }
  ]
},
{
  "_id": {
    "$oid": "655aa60c6823245d0d5511b3"
  },
  "Attribute": "Neutral",
  "Profile": [
    {
      "Digimon": "Kuramon",
      "Stage": "Baby",
      "Type": "Free"
    },
    {
      "Digimon": "Pabumon",
      "Stage": "Baby",
      "Type": "Free"
    },
    {
      "Digimon": "Punimon",
      "Stage": "Baby",
      "Type": "Free"
    }
  ]
}
```

Figure 5: Each Digimon listed under the same attribute

Validation Rules

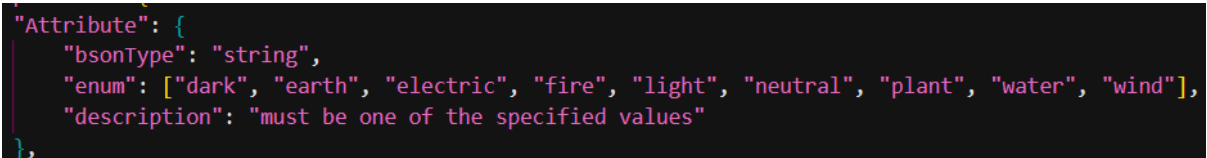
A validation rule is applied when updating or inserting documents into the "Attribute" collection using a JSON schema. This schema outlines the expected structure and constraints that the data must adhere to. The purpose of this rule is to uphold data consistency, simplify the process of querying and analysis, and ensure the data integrity to enable precise retrieval and manipulation.

By enforcing a standard structure through this rule, we can effectively prevent data entry errors that may compromise data consistency.

Attribute

For the "attribute" field, I have constrained the values to the nine existing attributes: dark, earth, electric, fire, light, neutral, plant, water, and wind. This means that if a document is inserted or updated in the collection where the "attribute" field contains a value not included in this list, MongoDB will reject the document because it does not comply to the defined schema structure.

In addition, I have defined the data type for this field as a string, ensuring that it can only contain text-based values. and a description is also included which documents the purpose of this validation.



```
"Attribute": {
  "bsonType": "string",
  "enum": ["dark", "earth", "electric", "fire", "light", "neutral", "plant", "water", "wind"],
  "description": "must be one of the specified values"
},
```

Figure 3: Validation Rule for Attribute Field

Profile

For the "profile" field, I have defined the data type as an array because it will contain three elements: the name, stage, and type of the Digimon. Additionally, I have included a description to explain the rationale behind this rule.

```
"Profile": {  
  "bsonType": "array",  
  "description": "must be an array"  
}
```

Figure 4: Validation Rule for Profile Field

Select Queries

For the efficiency of my system, I have integrated all select queries into functions that can be executed from a single Python file. The following queries are: *selecting all documents in a collection (attribute) and an array, showing a projection of a field, sorting output of attributes, and aggregation by type.*

```
Choose a function:  
1. Select All Documents  
2. Select Array Data  
3. Show Projection of Profile or Attribute  
4. Sort by attribute  
5. Aggregation by Type  
6. Exit  
  
Enter your choice (1-6): 
```

Figure 5: Display Menu of Select Queries

Select All Documents

This function is designed to print all the existing documents from the Attribute collection. As there are 9 types of attributes, it will print 9 documents along with their corresponding data.

Select Array Data

The purpose of this function is to print the embedded array associated with the provided attribute. This function will display the Profile array for each Digimon document that matches the specified attribute.

Show Projection of Profile or Attribute

This function is designed to print information for the specified field provided as input. When "attribute" is chosen, it will print the nine available traits, and when "profile" is chosen, it will print all the information from the "name," "stage," and "type" of each Digimon.

Sort by Attribute

This function is intended to sort the attributes in alphabetical order from "a" to "z." For a cleaner display, only the attributes are printed and sorted. However, the code to include profile information is available in the comments.

Aggregation by Type

Within this function, the ability to group values from other documents in the collection is achieved through aggregation. In this example, I have provided a demonstration of how to group all documents that share a specific given profile type.

Data Manipulation

Similarly, to *Select Queries*, I have integrated all my *insert*, *update*, and *delete* statements into one Python file, with each operation being defined as a separate function.

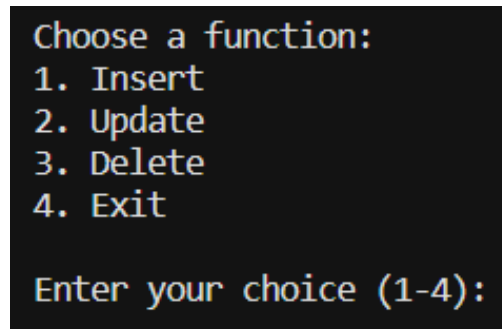


Figure 6: Display Menu of Insert, Update and Delete Statements

Insert

Using the insert statement, I can create a new document to be added to the Attributes collection. This document includes a new attribute, along with a set of profiles containing the name, stage, and type of Digimon, all of which inherit the specified attribute.

Update

With the update statement, we have the capability to modify the attributes of any Digimon, including their name, stage, and type, under the specified attribute. If the Digimon does not exist, an error message will be displayed to indicate that the Digimon does not exist, and no updates will be performed.

Delete

Documents of a specific attribute can be deleted using the delete statement. This action will remove the specified document from the collection – Attribute collection.

Summary

Within this document, I have provided an overview of my thought process and the design structure of my database and collection. I have carefully organised the dataset into a suitable category, which is the "attributes" collection. This report also includes all the validation rules to detect errors during the creation of documents.

Lastly, I have outlined a menu of select queries that enable users to view various documents and perform data manipulation operations such as inserting, updating, and deleting documents.