DATA 514- System Summary & Design

Danish Nadeem, Harshit Rai, Ishank Vasania

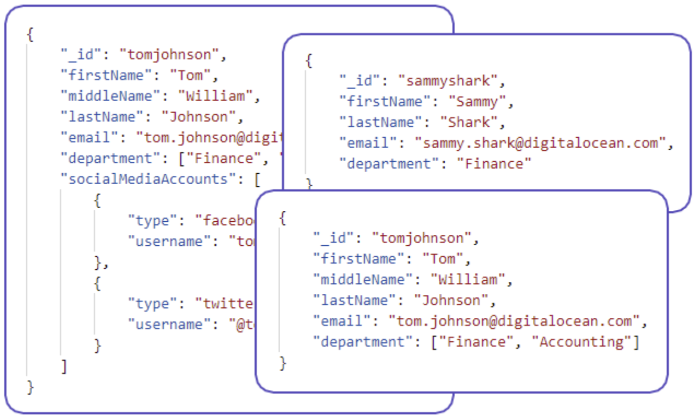
1. System Summary

1.1 Introduction

The system of our choice, MongoDB, is a NoSQL document-based database management system. MongoDB is an open-source, highly scalable, and flexible database that is designed to handle unstructured data. It is widely used in web applications and big data environments due to its ease of use and high performance. Following are the key features of MongoDB.
Overall, MongoDB is a powerful and flexible database management system that can handle a wide range of use cases. Its document-based data model, strong consistency guarantees, and built-in support for replication and sharding make it an excellent choice for big data environments and web applications.

1.2 Data Model

MongoDB uses a document-based data model. Documents are stored as JSON-like objects, and each document can have a different structure. Each document contains key-value pairs, where the keys represent field names and the values represent data. MongoDB supports dynamic schema, which means that we can add new fields to a document at any time. Below is an example of how data can be stored in a MongDB document-based data *collection*. The Data Model used for the Amazon dataset is explained in more detail in the later part of this document.



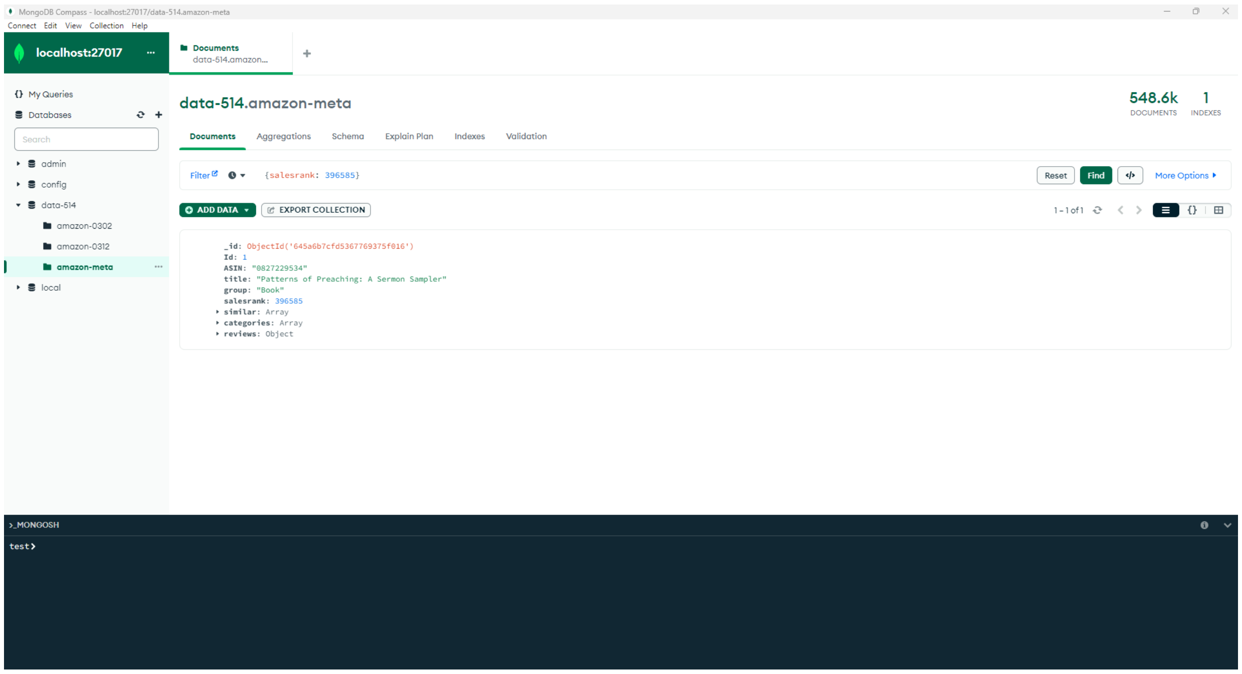
```
{
    "_id": "tomjohnson",
    "firstName": "Tom",
    "middleName": "William",
    "lastName": "Johnson",
    "email": "tom.johnson@digi
    "department": ["Finance", "
    "socialMediaAccounts": [
        {
            "type": "facebo
            "username": "to
        },
        {
            "type": "twitte
            "username": "@t
        }
    ]
}
```

```
{
    "_id": "sammyshark",
    "firstName": "Sammy",
    "lastName": "Shark",
    "email": "sammy.shark@digitalocean.com",
    "department": "Finance"
}
```

```
{
    "_id": "tomjohnson",
    "firstName": "Tom",
    "middleName": "William",
    "lastName": "Johnson",
    "email": "tom.johnson@digitalocean.com",
    "department": ["Finance", "Accounting"]
}
```

1.3 User Interface

MongoDB provides a command-line interface and a graphical user interface called MongoDB Compass. Using these interfaces, we can interact with the database by running queries, creating collections, and manipulating documents. Below is an illustration of running a search operation using the Compass GUI.



data-514.amazon-meta

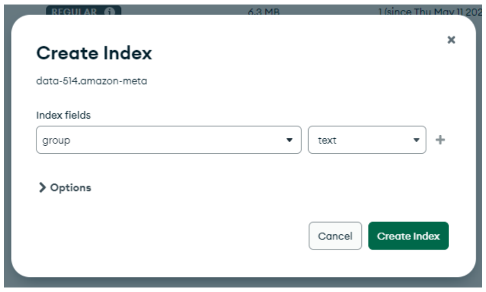
Filter: `{salesrank: 396585}`

```
_id: ObjectId('645a6b7cfd5367769375f016')
Id: 1
ASIN: "0827229534"
title: "Patterns of Preaching: A Sermon Sampler"
group: "Book"
salesrank: 396585
similar: Array
categories: Array
reviews: Object
```

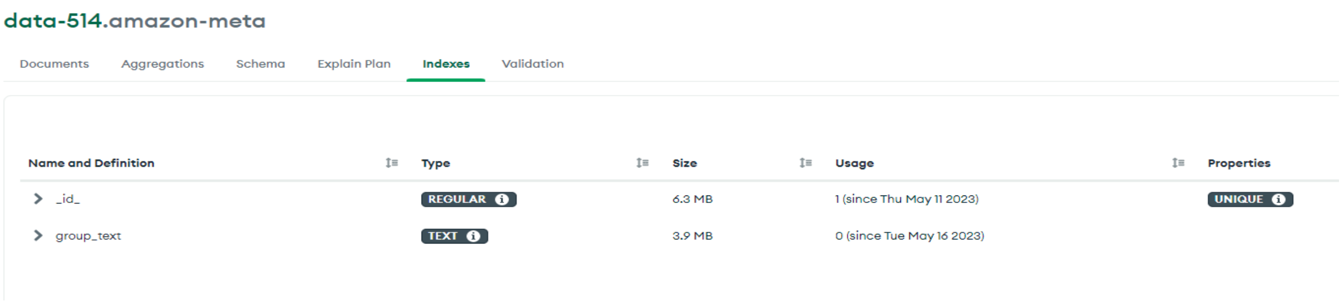
```
test>
```

1.4 Indexes

MongoDB supports various types of indexes, including single field indexes, compound indexes, multikey indexes, geospatial indexes, and text indexes. Indexes can significantly improve query performance by reducing the number of documents that need to be scanned. Below is an example of an index created on the *group* field of the dataset.



Index fields: group | text



Name and Definition	Type	Size	Usage	Properties
_id_	REGULAR	6.3 MB	1 (since Thu May 11 2023)	UNIQUE
group_text	TEXT	3.9 MB	0 (since Tue May 16 2023)	

1.5 Consistency

MongoDB provides strong consistency guarantees, which means that all reads and writes to a document are consistent and ordered. MongoDB also supports transactions, which enable multiple operations to be executed as an atomic unit.

1.6 Scalability and Replication

MongoDB provides built-in support for replication and sharding, enabling high availability and horizontal scaling. Replication involves copying data across multiple servers, while sharding involves partitioning data across multiple servers. Sharding can be implemented in MongoDB by using the query mentioned below.

```
sh.enableSharding(database, primaryShard)
```

2. System Installation

We've decided to follow a two-step approach-

1. Initially install the desktop version of MongoDB on the local machine for testing and exploration.
2. Then create a fully-managed MongoDB service via Azure CosmosDB.

This two-step approach will allow us to quickly set up the dataset for initial testing and exploration on the local machine. Once we have a clear understanding of MongoDB and the data, then we move it to the fully-managed service on Microsoft Azure and work on query optimization in the context of a cloud service.

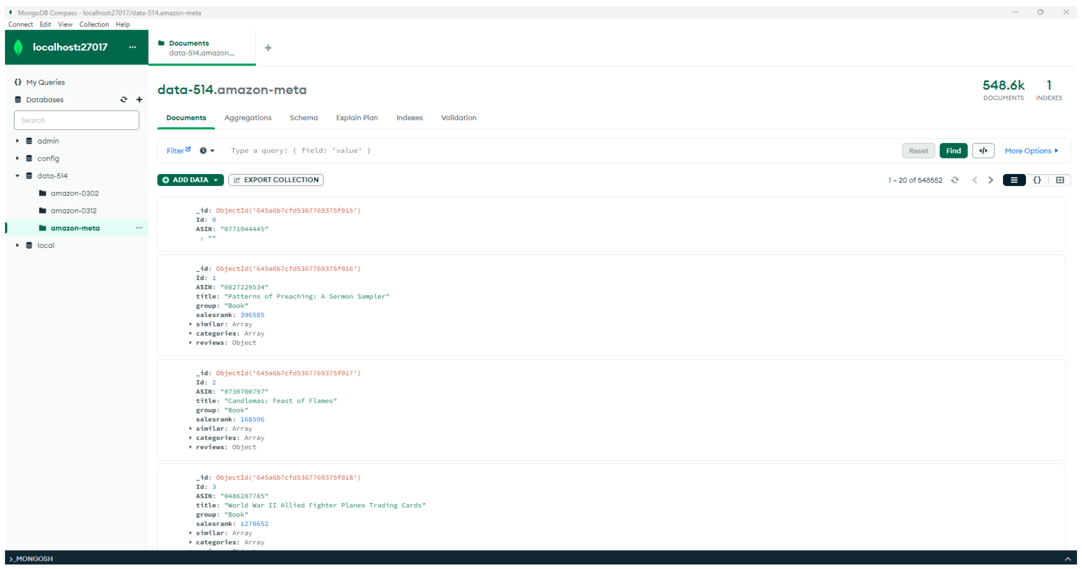
Due to the above approach, we first verify that MongoDB and its client facing user-interface called Compass are installed and functional on the local machine. After that, we will verify the set up of a fully-managed MongoDB service on Microsoft Azure via CosmosDB.

2.1 MongoDB on local machine

The screenshot below shows that MongoDB and its companion GUI, Compass, was set up and the Amazon dataset was imported successfully.

2.2 Fully-managed MongoDB service on Microsoft Azure

The screenshot below shows that MongoDB was set up successfully on Microsoft Azure via the CosmosDB API.



MongoDB Compass - localhost:27017/data-514.amazon-meta

data-514.amazon-meta — 548.6k DOCUMENTS, 1 INDEXES — 1 – 20 of 548552

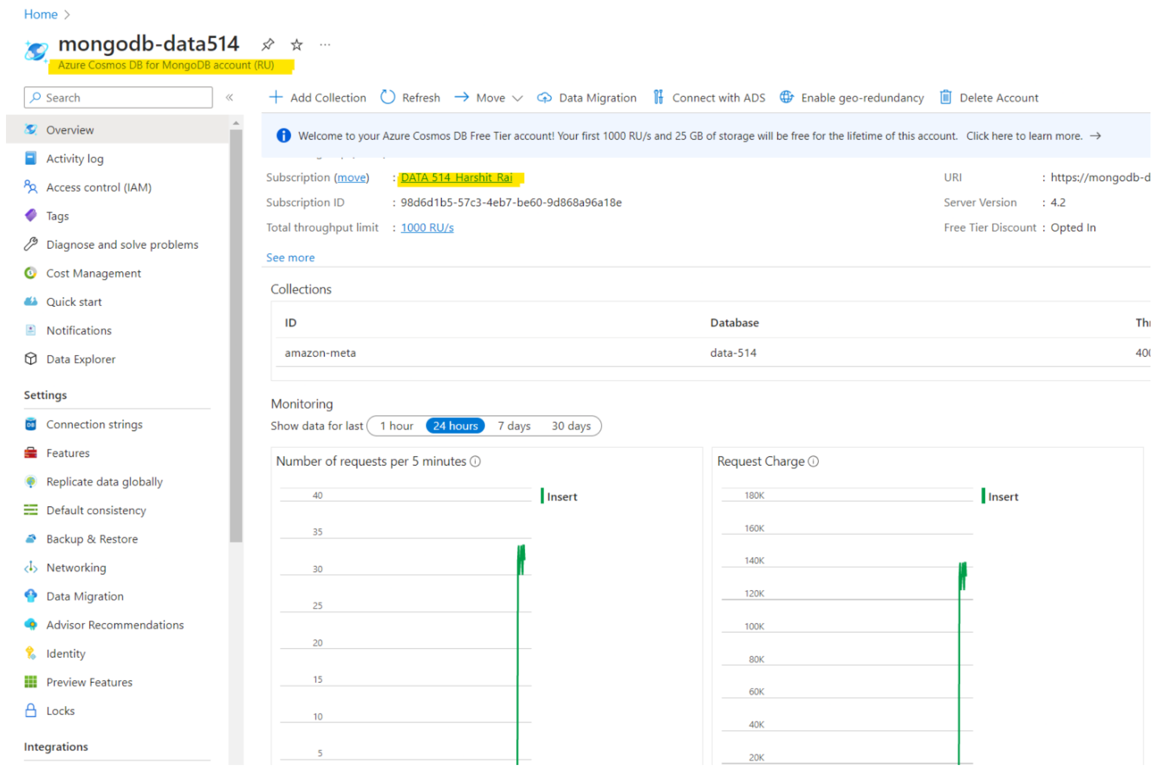
```
_id: ObjectId('645a6b7cfd5367769375f015')
Id: 0
ASIN: "0771044445"
 : ""

_id: ObjectId('645a6b7cfd5367769375f016')
Id: 1
ASIN: "0827229534"
title: "Patterns of Preaching: A Sermon Sampler"
group: "Book"
salesrank: 396585
similar: Array
categories: Array
reviews: Object

_id: ObjectId('645a6b7cfd5367769375f017')
Id: 2
ASIN: "0738700797"
title: "Candlemas: Feast of Flames"
group: "Book"
salesrank: 168596
similar: Array
categories: Array
reviews: Object

_id: ObjectId('645a6b7cfd5367769375f018')
Id: 3
ASIN: "0486287785"
title: "World War II Allied Fighter Planes Trading Cards"
group: "Book"
salesrank: 1270652
similar: Array
categories: Array
```

2.1 MongoDB on local machine



mongodb-data514

Azure Cosmos DB for MongoDB account (RU)

Welcome to your Azure Cosmos DB Free Tier account! Your first 1000 RU/s and 25 GB of storage will be free for the lifetime of this account. Click here to learn more.

Subscription (move)	: DATA 514 Harshit Rai
Subscription ID	: 98d6d1b5-57c3-4eb7-be60-9d868a96a18e
Total throughput limit	: 1000 RU/s
URI	: https://mongodb-d
Server Version	: 4.2
Free Tier Discount	: Opted In

Collections

ID	Database	Th
amazon-meta	data-514	40

Monitoring

Show data for last 1 hour [✓ 24 hours] 7 days 30 days

2.2 Fully-managed MongoDB service on Microsoft Azure

3. System Design

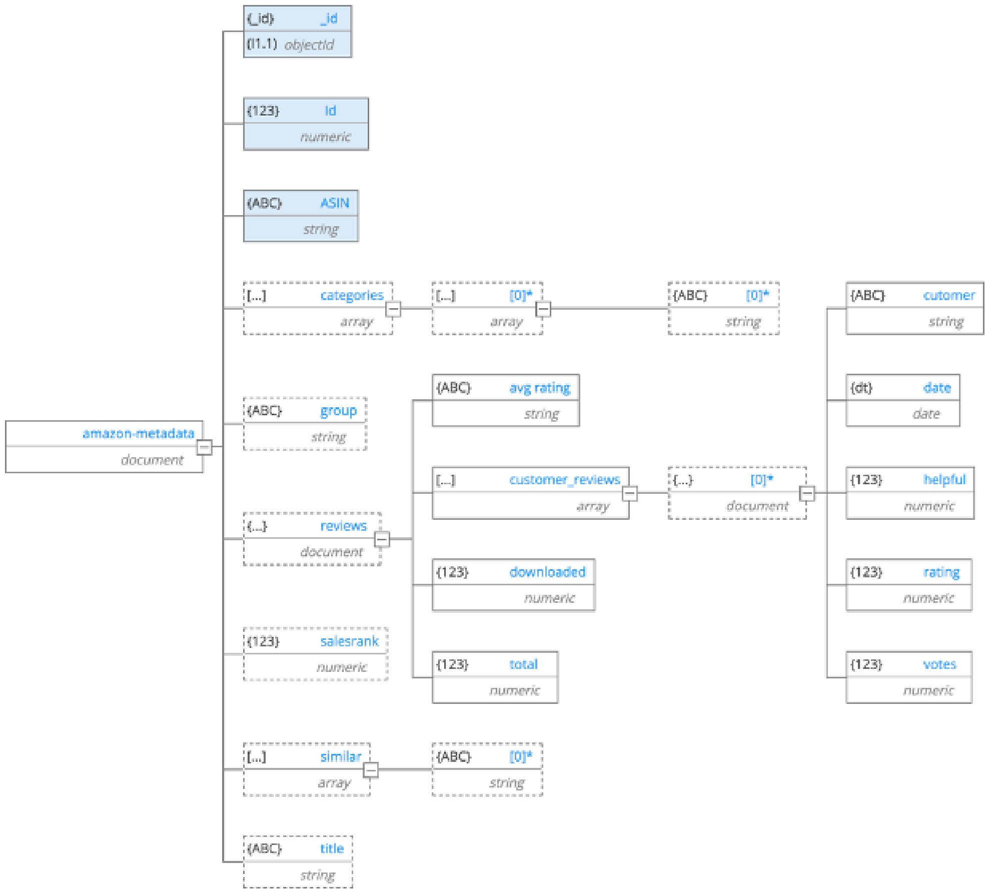
3.1 Storing the data in MongoDB

The following steps will be followed to store the dataset in MongoDB.

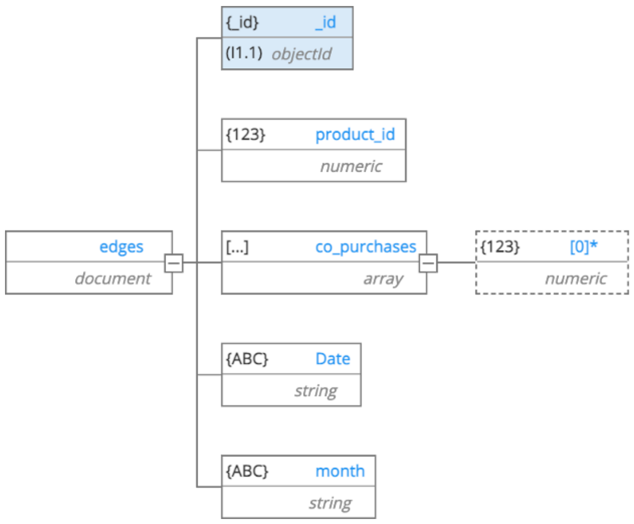
1. Setup the MongoDB system on a local machine and create a corresponding MongoDB instance on Microsoft Azure.
2. Design the data model necessary to store the data into the MongoDB database. The complete schema and with correct data types are also decided in this step.
3. Perform preprocessing to convert the raw data in text format to JSON format using Python. This will help loading the data into a format supported by MongoDB and the data model designed in the previous step.
2. Create a MongoDB database and collection to store the dataset. We use the MongoDB (Compass) graphical user interface to create the database and collection.
3. Once the data is stored in MongoDB, perform various queries and analysis on the data using the supported query language and aggregation framework.

3.2 Data Model & Schema

The two diagrams below illustrate the schema of the *amazon-metadata* and *edges* collections that are part of the Amazon dataset under consideration.



3.2.1: Schema of the *amazon-metadata* collection.



3.2.2: Schema of the *edges* collection.

Resultant schema generated for *amazon-metadata* collection in MongoDB:

```
{
        _id: ObjectId,
        Id: NumberInt,
        ASIN: String,
        title: String,
        group: String,
        salesrank: NumberInt,
        similar: [String],
        categories: [[String]],
        reviews: {
        total: NumberInt,
        downloaded: NumberInt,
        avg_rating: String,
        customer_reviews: [
        {
                date: ISODate,
                customer: String,
                rating: NumberInt,
                votes: NumberInt,
                helpful: NumberInt
        }
        ]
        }
}
```

Detail description of each field in the schema:

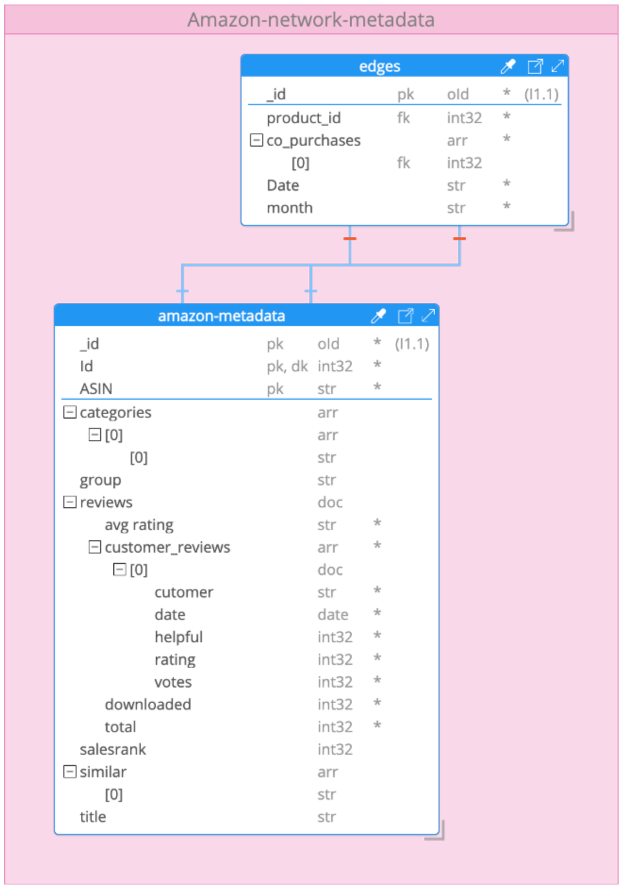
1. **_id**: A unique identifier for the document (MongoDB's ObjectId type)
2. **Id**: An integer identifier for the document
3. **ASIN**: The Amazon Standard Identification Number for the book (a string)
4. **title**: The title of the book (a string)
5. **group**: The type of item (a string)
6. **salesrank**: The sales rank of the book on Amazon (an integer)
7. **similar**: An array of ASINs for books that are similar to this one (an array of strings)
8. **categories**: An array of arrays representing the categories and subcategories that the book belongs to (an array of arrays of strings)
9. **reviews**: An object containing information about the book's reviews, including the total number of reviews, the number that have been downloaded, and the average rating (an object with three fields: **total, downloaded, and avg_rating**)
10. **customer_reviews**: An array of objects representing individual customer reviews, including the date the review was posted, the customer's identifier, the rating, the number of votes the review received, and the number of helpful votes the review received (an array of objects with five fields: **date, customer, rating, votes, and helpful**).

Resultant schema generated for *amazon-metadata* collection in MongoDB:

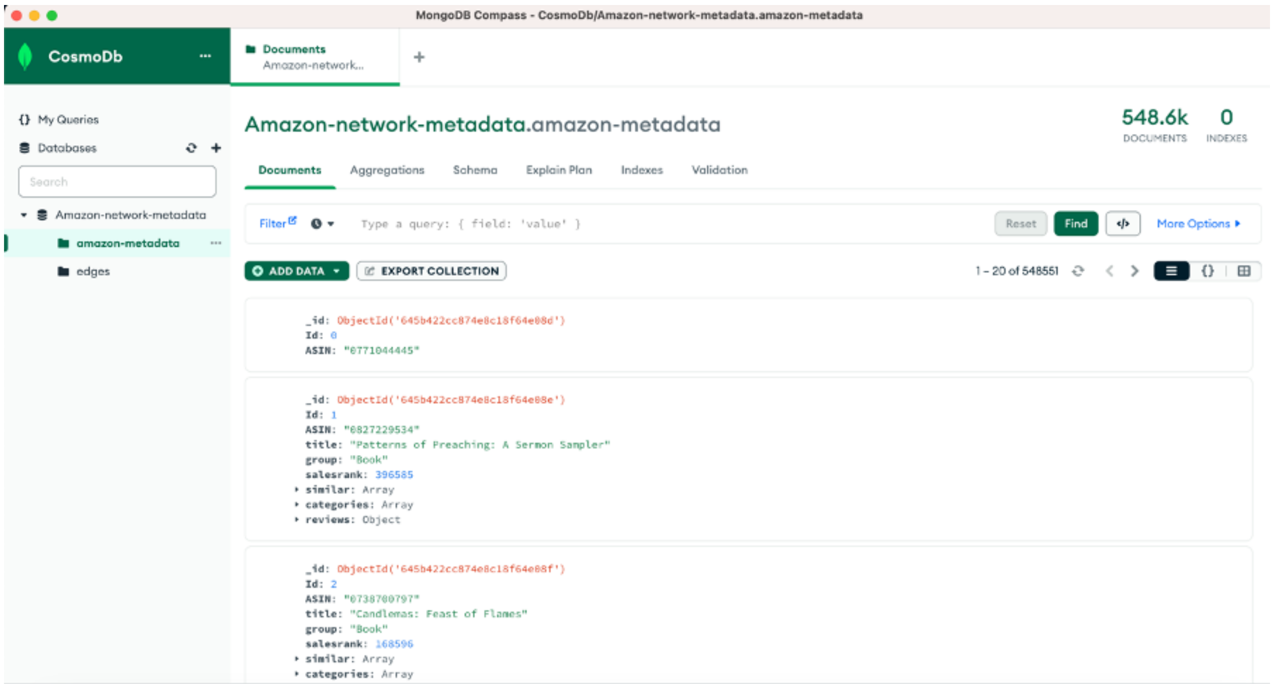
```
co-purchases:
{
        _id: ObjectId,
        product_id: NumberInt,
        co_purchases: [NumberInt],
        date: ISODate,
        month: String
}
```

Detail description of each field in the schema:

1. **_id**: A unique identifier for the document (MongoDB's ObjectId type)
2. **product_id**: An integer identifier for the product that was purchased
3. **co_purchases**: An array of integers representing the product IDs of items that were purchased together with the product identified by product_id
4. **date**: The date the purchase was made (a MongoDB-specific ISODate type)
5. **month**: The month in which the purchase was made (a string)



3.2.3: ER Diagram for the relationship between edges and the amazon-metadata collections

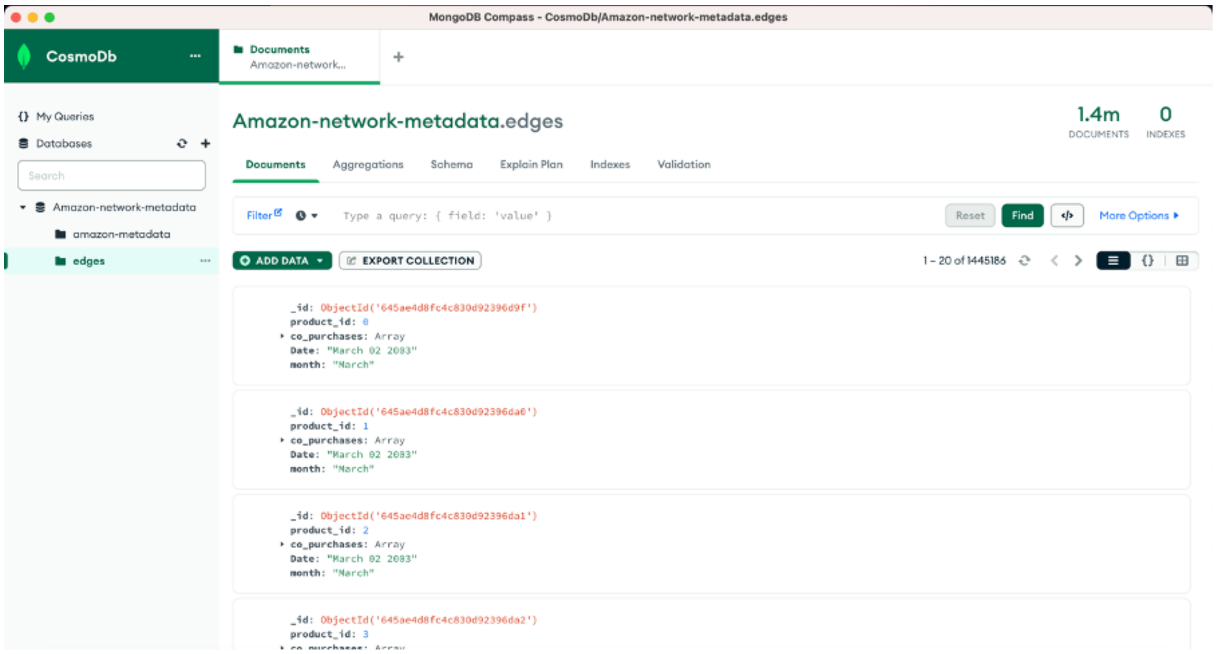


```
_id: ObjectId('645b422cc874e8c18f64e08d')
Id: 0
ASIN: "0771044445"

_id: ObjectId('645b422cc874e8c18f64e08e')
Id: 1
ASIN: "0827229534"
title: "Patterns of Preaching: A Sermon Sampler"
group: "Book"
salesrank: 396585
▸ similar: Array
▸ categories: Array
▸ reviews: Object

_id: ObjectId('645b422cc874e8c18f64e08f')
Id: 2
ASIN: "0738700797"
title: "Candlemas: Feast of Flames"
group: "Book"
salesrank: 168596
▸ similar: Array
▸ categories: Array
```

3.2.4: Example of how the amazon-metadata collection looks like in MongoDb



```
_id: ObjectId('645ae4d8fc4c830d92396d9f')
product_id: 0
▸ co_purchases: Array
Date: "March 02 2003"
month: "March"

_id: ObjectId('645ae4d8fc4c830d92396da0')
product_id: 1
▸ co_purchases: Array
Date: "March 02 2003"
month: "March"

_id: ObjectId('645ae4d8fc4c830d92396da1')
product_id: 2
▸ co_purchases: Array
Date: "March 02 2003"
month: "March"

_id: ObjectId('645ae4d8fc4c830d92396da2')
product_id: 3
▸ co_purchases: Array
```

3.2.5: Example of how the edges collection looks like in the MongoDb.

Note: The above screenshots represent that state of the dataset before creating indexes and with only 90% data loaded.
Updated images will be attached in the final report.

3.3 Indexes for the Dataset

1. Unique Index on ASIN: The **ASIN** field is unique for each document in the **products** collection, hence we created a unique index on this field to prevent duplicate products from being inserted.
2. Unique Index on Product ID: The **product_id** field is unique for each document in the **products** collection, hence we created a unique index on this field to prevent duplicate products from being inserted.

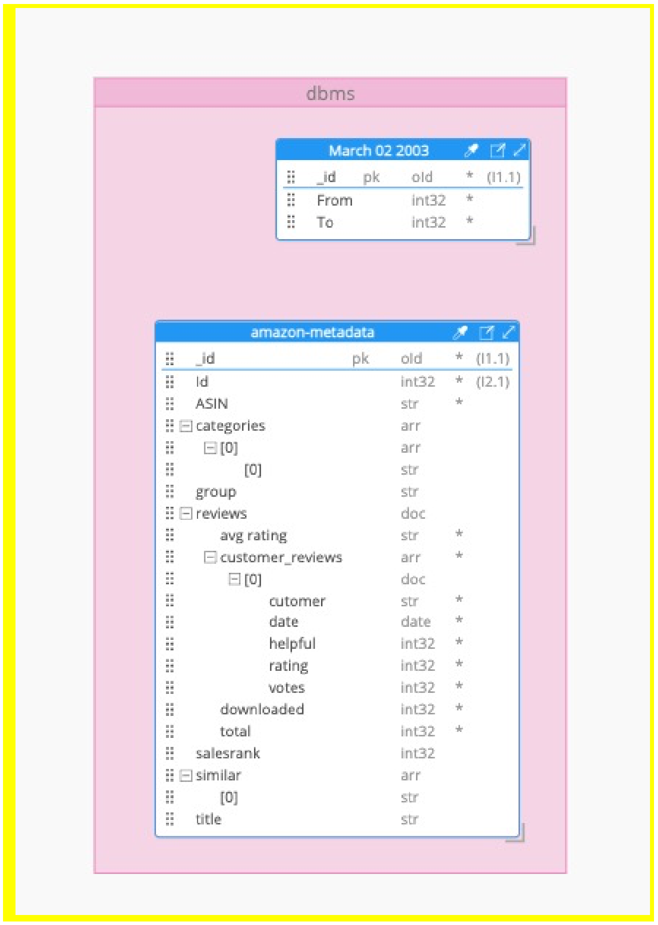
3.4. Reason for selecting the data model and it’s potential variant

While designing the data model, the key considerations include the performance requirements of the application, the complexity of the data relationships, and the need for data consistency and integrity.

The original dataset had 4 different files to represent *edges*. If we included these 4 files directly into our mongodb database the size limit was exceeded as a total of 5 collections were found to be too intensive for MongoDB to handle given the resource constraints. In the variant, as can be seen in the image below, we had the edges from 02-March-2003.

To tackle this issue, all of the 4 files containing *edges* collection were merged into one file. The new resultant file created is shown in the ER diagrams in the previous parts and now also has date as well as co-purchases. We basically converted the data from long format to wide format hence saving memory consumption.

In conclusion, for the Amazon product review dataset, we chose a document-based data model in MongoDB due to its flexibility, scalability, and ability to handle unstructured data efficiently.



March 02 2003

_id	pk	oId	*	(I1.1)
From		int32	*	
To		int32	*	

amazon-metadata

_id	pk	oId	*	(I1.1)
Id		int32	*	(I2.1)
ASIN		str	*	
categories		arr		
[0]		arr		
[0]		str		
group		str		
reviews		doc		
avg rating		str	*	
customer_reviews		arr	*	
[0]		doc		
cutomer		str	*	
date		date	*	
helpful		int32	*	
rating		int32	*	
votes		int32	*	
downloaded		int32	*	
total		int32	*	
salesrank		int32		
similar		arr		
[0]		str		
title		str		

3.5 Preprocessing to restructure the data

The "amazon-meta.txt" input file contains data that our custom Python script analyses and converts to JSON format. The input file includes details on different items sold on Amazon’s website. To extract particular patterns, including customer reviews, overall reviews, and information about global blocks, our script makes use of regular expressions. The aforementioned Python script handles a range of keys and values, transforms some values into integers or floats, and arranges the information into dictionaries. The parsed data is then written in JSON format to a new file called "sample.json" and kept in a list. Overall, the script organizes and extracts the pertinent data from the input file into a more understandable and consistent format for additional processing or analysis.

3.6 Questions answered

The following questions will be answered using various query operations as part of the final implementation.

1. What are the percentages of each rating digit for the product with id: 21?
2. Which pairs of products have consistently stayed within each other's "Customers who bought this item also bought" lists through March, May, and June of 2003?
3. For the product with ASIN: 0385492081, display the most helpful review(s) of the highest rating and the most helpful review(s) of the lowest rating.

3.7 Decision Regarding Implementation

It was mutually decided that we will be demonstrating various features of MongoDB by hosting the Amazon Co-purchasing Networking dataset on MongoDB service provided by Microsoft Azure.

3.8 Project Plan Revision

There are currently no revisions to the proposed project plan.