Big Data Assignment 1

KFWJOR001 MRCGAB004 WHLJOS001 CRGMAT002

March 1, 2024

Contents

1 Find or Create a Suitable Data Set

1.1 Data Set Explanation

Link to the dataset: https://github.com/zygmuntz/goodbooks-10k

The dataset includes data from an online book review platform (https://goodreads.com/). It includes information about the books, users, book tags, and book rating scores. The dataset initially contained multiple csv files. This dataset was chosen as its ideal for a MongoDB database due to its semi-structured nature and nested data, which is particularly useful for storing ratings and book tags.

Dataset Content:

- **books.csv**: Each entry represents a book with a unique `book_id`. There are multiple data fields for a book:
  - **`book_id`, `goodreads_book_id`, `best_book_id`, `work_id`**: Unique id's representing a book, each with a different purpose. We only used `book_id` and `goodreads_book_id` as they're used to link books to user `ratings` and user `to_read` lists.
  - **`ratings_1`, `ratings_2`, `...`**: Number of user ratings by rating value. eg. `ratings_1` represents the number of 1 star ratings given to that book.
  - The rest of the fields are self explanatory but include info relating to authors, title, release date, and isbn number.
- **ratings.csv**: Each entry is a `user_id` to `book_id` mapping with a rating.
  - **`book_tags.csv`**: Each entry is a `book_id` to `tag_id` mapping.
  - **`tags.csv`**: Each entry is a tag_{id} to `tag_name` mapping.
  - **`to_read.csv`** : Each entry is a `user_id` to `goodreads_book_id` mapping which represents a user adding a book to their `to_read` list.

1.2 Data Pre-Processing

The data was processed such that the data was represented in JSON format with evidence of nested objects so that we could demonstrate the capabilities of MongoDB.

Here is a quick outline on how we processed the data to create JSON files:
Libraries used: `Pandas`, `PyArrow`, `Faker`

`Pandas` was used to load the csv files into dataframes where we merged data and applied `group by` aggregate functions to obtain lists of data objects per a unique entry id. This was useful, for example, when we obtained a list of tags per `book_id`.

`Faker` was used to generate random usernames for each id that were then written to `user_data.csv`. The `dataframes` were then converted into JSON files.

All data pre-processing code is in the data-processing directory but the output JSON files are included in the final submission.

2 Design a MongoDB Database

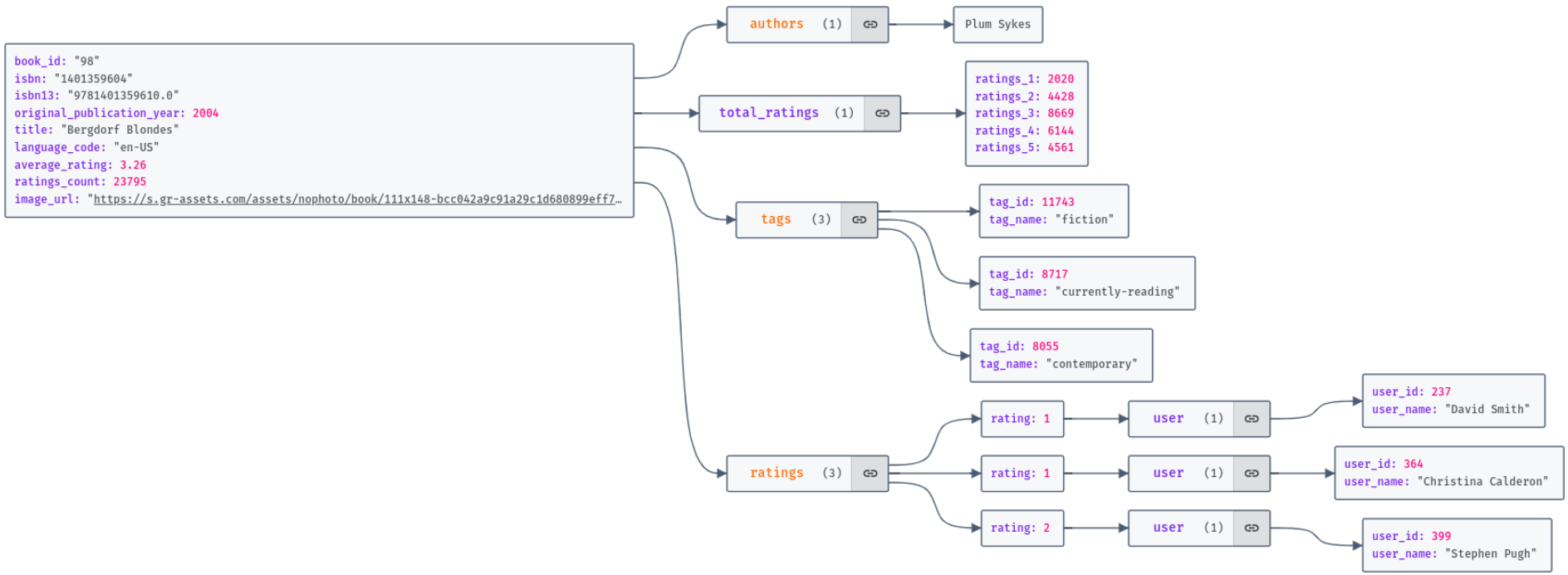
Both Collection Schemas were designed by creating hand-made JSON example objects. Each of these objects shows what a document in the DB would look like. Underneath each JSON example, we have included a diagram which represents the example's nesting visually.

2.1 Collection 1 - books

JSON example


```
{
    "book_id": "98",
    "isbn": "1401359604",
    "isbn13": "9781401359610.0",
    "authors": [
        "Plum Sykes"
    ],
    "original_publication_year": 2004,
    "title": "Bergdorf Blondes",
    "language_code": "en-US",
    "average_rating": 3.26,
    "ratings_count": 23795,
    "total_ratings": {
        "ratings_1": 2020,
        "ratings_2": 4428,
        "ratings_3": 8669,
        "ratings_4": 6144,
        "ratings_5": 4561
    },
    "image_url": "https://s.gr-assets.com/assets/nophoto/book/111x148-bcc042a9c91a29c1d680899eff700a03.png
    "tags": [
        {
            "tag_id": 11743,
            "tag_name": "fiction"
        },
        {
            "tag_id": 8717,
            "tag_name": "currently-reading"
        },
        {
            "tag_id": 8055,
            "tag_name": "contemporary"
        }
    ],
    "ratings": [
        {
            "user": {
                "user_id": 237,
                "user_name": "David Smith"
            },
            "rating": 1
        },
```

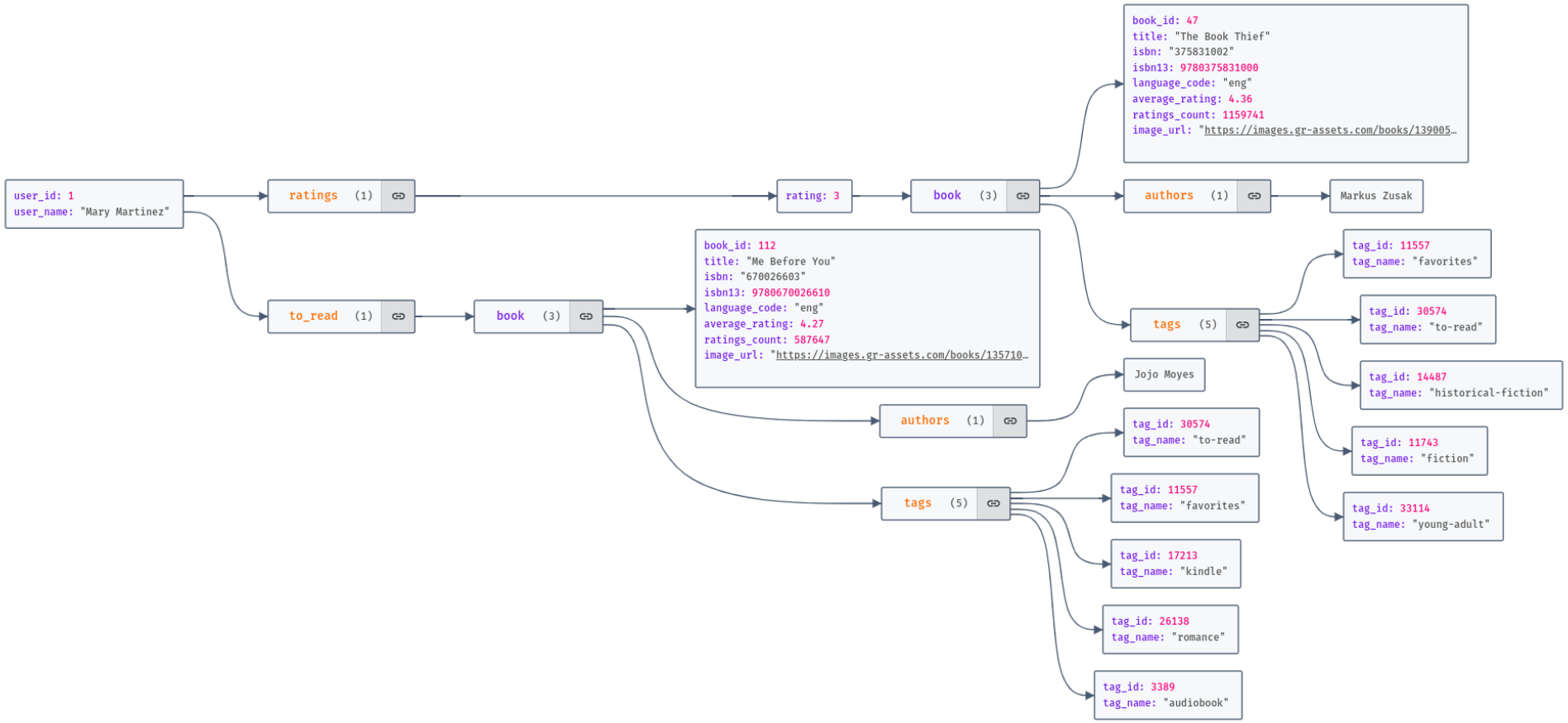
```
        {
            "user": {
                "user_id": 364,
                "user_name": "Christina Calderon"
            },
            "rating": 1
        },
        {
            "user": {
                "user_id": 399,
                "user_name": "Stephen Pugh"
            },
            "rating": 2
        }
    ]
}
```


book_id: "98"
isbn: "1401359604"
isbn13: "9781401359610.0"
original_publication_year: 2004
title: "Bergdorf Blondes"
language_code: "en-US"
average_rating: 3.26
ratings_count: 23795
image_url: "https://s.gr-assets.com/assets/nophoto/book/111x148-bcc042a9c91a29c1d680899eff7…
authors (1)
Plum Sykes
total_ratings (1)
ratings_1: 2020
ratings_2: 4428
ratings_3: 8669
ratings_4: 6144
ratings_5: 4561
tags (3)
tag_id: 11743
tag_name: "fiction"
tag_id: 8717
tag_name: "currently-reading"
tag_id: 8055
tag_name: "contemporary"
ratings (3)
rating: 1
user (1)
user_id: 237
user_name: "David Smith"
rating: 1
user (1)
user_id: 364
user_name: "Christina Calderon"
rating: 2
user (1)
user_id: 399
user_name: "Stephen Pugh"

2.2 Collection 2 - users


```
{
    "user_id": 1,
    "user_name": "Mary Martinez",
    "ratings": [
        {
            "book": {
                "book_id": 47,
                "authors": [
                    "Markus Zusak"
                ],
                "title": "The Book Thief",
                "isbn": "375831002",
                "isbn13": 9780375831000.0,
                "language_code": "eng",
                "average_rating": 4.36,
                "ratings_count": 1159741,
                "image_url": "https://images.gr-assets.com/books/1390053681m/19063.jpg",
                "tags": [
                    {
                        "tag_id": 11557,
                        "tag_name": "favorites"
                    },
                    {
                        "tag_id": 30574,
                        "tag_name": "to-read"
                    },
                    {
                        "tag_id": 14487,
                        "tag_name": "historical-fiction"
                    },
                    {
                        "tag_id": 11743,
                        "tag_name": "fiction"
                    },
                    {
                        "tag_id": 33114,
                        "tag_name": "young-adult"
                    }
                ]
            },
            "rating": 3
        }
    ],
    "to_read": [
```

```
        {
            "book": {
                "book_id": 112,
                "authors": [
                    "Jojo Moyes"
                ],
                "title": "Me Before You",
                "isbn": "670026603",
                "isbn13": 9780670026610.0,
                "language_code": "eng",
                "average_rating": 4.27,
                "ratings_count": 587647,
                "image_url": "https://images.gr-assets.com/books/1357108762m/15507958.jpg",
                "tags": [
                    {
                        "tag_id": 30574,
                        "tag_name": "to-read"
                    },
                    {
                        "tag_id": 11557,
                        "tag_name": "favorites"
                    },
                    {
                        "tag_id": 17213,
                        "tag_name": "kindle"
                    },
                    {
                        "tag_id": 26138,
                        "tag_name": "romance"
                    },
                    {
                        "tag_id": 3389,
                        "tag_name": "audiobook"
                    }
                ]
            }
        }
    ]
}
```


user_id: 1
user_name: "Mary Martinez"
ratings (1)
rating: 3
book (3)
book_id: 47
title: "The Book Thief"
isbn: "375831002"
isbn13: 9780375831000
language_code: "eng"
average_rating: 4.36
ratings_count: 1159741
image_url: "https://images.gr-assets.com/books/139005…
authors (1)
Markus Zusak
tags (5)
tag_id: 11557
tag_name: "favorites"
tag_id: 30574
tag_name: "to-read"
tag_id: 14487
tag_name: "historical-fiction"
tag_id: 11743
tag_name: "fiction"
tag_id: 33114
tag_name: "young-adult"
to_read (1)
book (3)
book_id: 112
title: "Me Before You"
isbn: "670026603"
isbn13: 9780670026610
language_code: "eng"
average_rating: 4.27
ratings_count: 587647
image_url: "https://images.gr-assets.com/books/135710…
authors (1)
Jojo Moyes
tags (5)
tag_id: 30574
tag_name: "to-read"
tag_id: 11557
tag_name: "favorites"
tag_id: 17213
tag_name: "kindle"
tag_id: 26138
tag_name: "romance"
tag_id: 3389
tag_name: "audiobook"

2.3 Explanation and Justification

The data in CSV format emulates the functionality of a relational database. Many of the fields have foreign keys that point to elements in the other CSV files. Document store database favour efficiency over consistency, thus, we have nested a copy of the relevant object where the value would have otherwise been a foreign key. The process by how this was achieved was highlighted in "Data Pre-Processing".

The data was seeded into the following 2 collections:

- `books`
- `users`

2.3.1 Books

The `books` collection roughly followed the format of `books.csv` (outlined in "Data Set Explanation") with some modifications. We omitted unnecessary information and altered the names of some of the properties to make their semantic meaning clearer. As a result each book document includes:

- General information about the book (`title`, `author`, `original_publication_year`, etc.).
- Aggregated rating values (`average_rating`, `total_ratings`, `ratings_counts`).
- A `ratings` list
  - Each element represents a `user`'s review of that book. It includes the `user`'s basic information and the assigned `rating` score.
- A `tags` list. Each tag represents a genre or category the book belongs to.

The most significant element of our design was the aforementioned nesting. Rather than store a separate `tags` collection, all the tags associated with the book are stored as a list of objects. The same is true for `ratings`, which is a list of rating objects.

Use case examples:

- Collection of books: Querying to see the average rating of a book and the distribution of ratings of a particular book

2.3.2 Users

Unlike `books`, the `users` collection does not directly correspond to a csv file. Instead, `users` was created by combining data from `ratings.csv`, `to_read.csv`, and `books.csv`. We chose to create the user collection in order to demonstrate the importance of collection design with regards to query efficiency. While the `users` contains a lot of duplicate data from `books`, it does so in a way that places information about the users at the top of the nesting hierarchy. This means that data about individual users can be obtained without performing expensive joins. The chosen collection design allows one to access, store and perform analytics from the perspective of the user. Each document in the `users` collection includes the following:

- A `user_id` and `user_name` (randomly generated, as explained in "Data Pre-Processing").
- A `ratings` list:
  - Each element represents a score that the user has given to a book.
  - The element includes a `book` object and a given `rating` score.
  - The nested `book` object includes all high-level data about that book, as would be found in the `books` collection.
- A `to_read` list.
  - Each element represents a book that the user has added to their `to_read` list (ie. plans to read that book).
  - The element is represented as a `book` object that is identical in structure to those represented in the `ratings` list.

Use case examples:
Collection of users: what books user X wants to read.

3 Create and Load This MongoDB Database

3.1 Load the Database

The process of creating and loading the database was significantly simplified on account of the fact that we had already Pre-Processed the data into JSON format. For each collection, a shell command is called to seed the JSON data into their associated collections. The creation of the database is implicit ie. inserting data into the non-existent database leads to its creation. The commands use `mongoimport`, a CLI tool designed for extracting data from plain-text formats (eg. JSON) and inserting them into MongoDB databases.

3.1.1 Books Collection

```
mongoimport --db bookstore --collection books --file mongo-seed/books.json --jsonArray
```

3.1.2 Users Collection

```
mongoimport --db bookstore --collection users --file mongo-seed/users.json --jsonArray
```

3.2 Testing

The shell commands were tested and the output has been annotated on the following page:

```
BDAssignment1 on ᚠ main [$!?] via ⬢ v18.17.1 via 🐍 v3.11.5
❯ mongosh
Current Mongosh Log ID: 65e02d3d4ff0cb22198e19a8
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.5
Using MongoDB:          7.0.5
Using Mongosh:          2.1.5

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-02-28T15:34:02.442+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-02-28T15:34:02.442+00:00: vm.max_map_count is too low
------

test> show dbs
admin    40.00 KiB
config  108.00 KiB
local    72.00 KiB
test>
```

Show that db "bookstore" does not exist.

```
BDAssignment1 on ᚠ main [$!?] via ⬢ v18.17.1 via 🐍 v3.11.5 took 28s
❯ mongoimport --db bookstore --collection users --file mongo-seed/users.json --jsonArray
  mongoimport --db bookstore --collection books --file mongo-seed/books.json --jsonArray
2024-02-29T09:08:31.249+0200    connected to: mongodb://localhost/
2024-02-29T09:08:31.589+0200    255 document(s) imported successfully. 0 document(s) failed to import.
2024-02-29T09:08:31.602+0200    connected to: mongodb://localhost/
2024-02-29T09:08:31.824+0200    2184 document(s) imported successfully. 0 document(s) failed to import.
```

Import JSON

```
BDAssignment1 on ᚠ main [$!?] via ⬢ v18.17.1 via 🐍 v3.11.5
❯ mongosh
Current Mongosh Log ID: 65e02d938dc73afd1fd4d2df
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.5
Using MongoDB:          7.0.5
Using Mongosh:          2.1.5

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-02-28T15:34:02.442+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-02-28T15:34:02.442+00:00: vm.max_map_count is too low
------

test> use bookstore
switched to db bookstore
bookstore>
```

use bookstore

```
BDAssignment1 on ᚠ main [$!?] via ⬢ v18.17.1 via 🐍 v3.11.5 took 1m
❯ mongosh
Current Mongosh Log ID: 65e02dd18ea8219511814a69
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.5
Using MongoDB:          7.0.5
Using Mongosh:          2.1.5

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-02-28T15:34:02.442+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-02-28T15:34:02.442+00:00: vm.max_map_count is too low
------

test> show dbs
admin       40.00 KiB
bookstore    2.28 MiB
config     108.00 KiB
local       72.00 KiB
test> use bookstore
switched to db bookstore
bookstore> db.books.find({ tags: { $elemMatch: { tag_name: "mystery" } } }, { _id: 0, title: 1, authors: { $slice: 1 }, average_rating: 1 }).sort({ average_rating: -1 }).limit(5)
[
  {
    authors: [ 'Arthur Conan Doyle' ],
    title: 'Sherlock Holmes: The Complete Novels and Stories, Volume I',
    average_rating: 4.45
```

Bookstore is now 2.28MiB

Confirm with query.
Couldn't fit it in fully.

4 Discuss the Relative Benefits and Disadvantages of MongoDB

4.1 MongoDB (Document Store)

MongoDB is a document store database. Document store databases favour low retrieval latency and efficiency over consistency and functionality (expressiveness) when compared with relational databases. MongoDB is flexible: it fascilitates incremental schema modification as the schema is not rigidly defined. This can improve development velocity and account for changing stakeholder requirements.

MongoDB is highly scalable and will be efficient with the ever increasing addition of books being published. MongoDB meets a good middle-ground when compared with other NoSQL databases. It is fascilitates nested data and objects and allows querying on those objects without being overly strict or complex. Each Book and User is able to store all necessary information while allow for performant data retrieval without the overhead of complex joins.

The disadvantages of using a MongoDB is that there is a high degree of duplication and redundant data. This means that storing and transferring all of this data is computationally expensive. Secondly, inserting and updating data can be much more expensive than a relational database on account of the necessity to make updates in multiple places (due to the duplication). However, data about books is unlikely to change frequently (as the book has already been published) and thus this downside is heavily mitigated in the chosen example.

- **Benefits** (When compared with relational): Scalable, schema-free design, data-access performance
- **Disadvantages** (When compared with relational): Duplication, high storage use, slow modification, poor query expressiveness (functionality)
- **Role in polyglot persistence**:
  - A future polyglot design could store data that is high in volume but does not change frequently inside a document store db.
  - Books do not change frequently so the information about the books could be stored in the document store while users and ratings could be stored in other databases.
  - User telemetry could also be stored as in a document store database as it is high volume with very infrequent modification

4.2 Graph

Graph based databases allow for expressive and performant relationship representation and would be ideal for relationship related queries. Each relationship can be given properties that qualitatively describe that relationship. This is ideal for data where the relationships to other data is the primary provider of utility (eg. a knowledge graph), as opposed to the hierarchical categorisation (as is the case in MongoDB and relational databases). The chosen dataset is intrinsically hierarchical and it thus be a poor choice for graph DBs. The dataset would not leverage the benefits of a graph DB and would miss out on the well-fitted the benefits of MongoDB, namely schema flexibility, performance, and scalability, and simplicity.

- **Benefits**: Efficient modelling of highly inter-linked data, Query expressiveness (functionality), Great relationship representation (for which it is scalable and performant)
- **Disavantages**: Unnecessary complexity, poor modelling of hierarchical/structured data
- **Role in polyglot persistence**:
  - A future polyglot design could include a graph database that models the relationship books have to each other.
  - Each book could have an outgoing edge to related books with a descriptive tag that describes their relationship strength and type.
  - This would prove highly useful for the sites recommendation engine.

4.3 Key-value

Benefits:High performance, Simplicity, Flexible Disadvantages: Limited query capabilities and Does not support complex data structures. Role in polyglot persistence: It can store the current session data of users. Easy to keep track of users' sessions such as how long they were on the app.

Flexibility would be very important as it would allow us to store more complex session information about the user which can be used for processing later on to improve the users experience on the application. Column-family Column family databases could work well with the 'good reads' dataset in calculating metrics and performing analytics, such as how many books are users reading and the total number of stars people rated individual books, how long people stay on the website, etc. The disadvantages are the queries one can perform are more basic and one cannot perform more complex queries beyond this.

Benefits: Compression of data, Scalable, fast to load queries, good for simple analytics Disadvantages:limited querying capabilities, limited data modelling capabilities Role in polyglot persistence: cannot think of any reason for its use on goodreads data storage. We would always have the same values for each object and so a columns family compression/efficiency would not be needed

Relational

Relational databases are great for keeping data consistent and for effectively maintaining the relationships of the data. The issue comes when we have too much data. Data analysis and operations are impacted substantially. it becomes inefficient with a very large (big) amount of data which is the issue one would run into with good-reads.

Benefits: Easy to use, accuracy, data integrity Disadvantages:cost , physical storage, decrease in performance over time Role in polyglot persistence:no function needed in this data but a good use case would be for keeping track of transactional data. Database is so large that one needs to use Nosql techniques to not be impacted by the size the relational database will create. We do not keep track of transactions as well so would not need the data integrity of a relational database Hierarchical Hierarchical databases are great for data with inherent hierarchies. Hierarchical databases would be great for fast lookup and to effectively store data. The issue is the data isn't inherently hierarchical and there are much more effective structures one could use to store the good reads data set.

Benefits:Fast data retrieval, efficient storage of data, predictable data structure, Disadvantages:Limited flexibility, hard to maintain and update, limited interoperability. Role in polyglot persistence: cannot think of any reason for its use on goodreads data storage.Our data is not hierarchical in structure and is flat and so there would be no need for a hierarchical data structure.