ADM Project

Maurizio Meschi & Nicolò Trebino

January 30, 2025

Email: maurizio.meschi@gmail.com, nicolo.trebino@gmail.com
GitHub: @Maurizio-Meschi, @nicolotrebino
Repo: GitHub Repository

Contents

1 Introduction

As passionate people who read books, we have chosen to take inspiration from this dataset from kaggle.com.
However, since it does not fit our application domain quite well, we have used Hardcover API to retrieve all the needed data and create the data set.

Our domain application consists of an on-line catalog where USERS can REVIEW and rate all the books and share their thoughts with a huge community of readers around the world.

Our application aims to help people find the perfect book that matches their preferences, and we chose a workload that includes the main operations that a normal user usually does in our catalog.

A *possible future implementation* can be a recommendation system that is very important in all modern social network applications that, based on what a user reads or likes, can recommend books or users to our customers, strengthening the interconnections between users, improving the overall impact and effectiveness of our application.

2 System requirements

Our application is suitable for a transactional, read/write intensive scenario since users can search for specific books and information and write reviews and scores.

Batch processing is not the focus of our application, even if our catalog automatically performs the average of the ratings of a particular book when a user lends on the book page: an action that requires batch access in the database.

We want our system to be scalable to provide for an ever-increasing number of requests in the future, expecting an ever-growing number of users. We also need high availability and high throughput to provide all requests, both read and write or update, that our users may make.

Partition tolerance is also a key point in our system since we need the system to continue to work even after a network partition or failure, so we have to rely, following the CAP theorem, on an eventual consistency level, to guarantee that, given enough time and no new updates, all replicas of data will eventually converge to the same state.

3 ER diagram

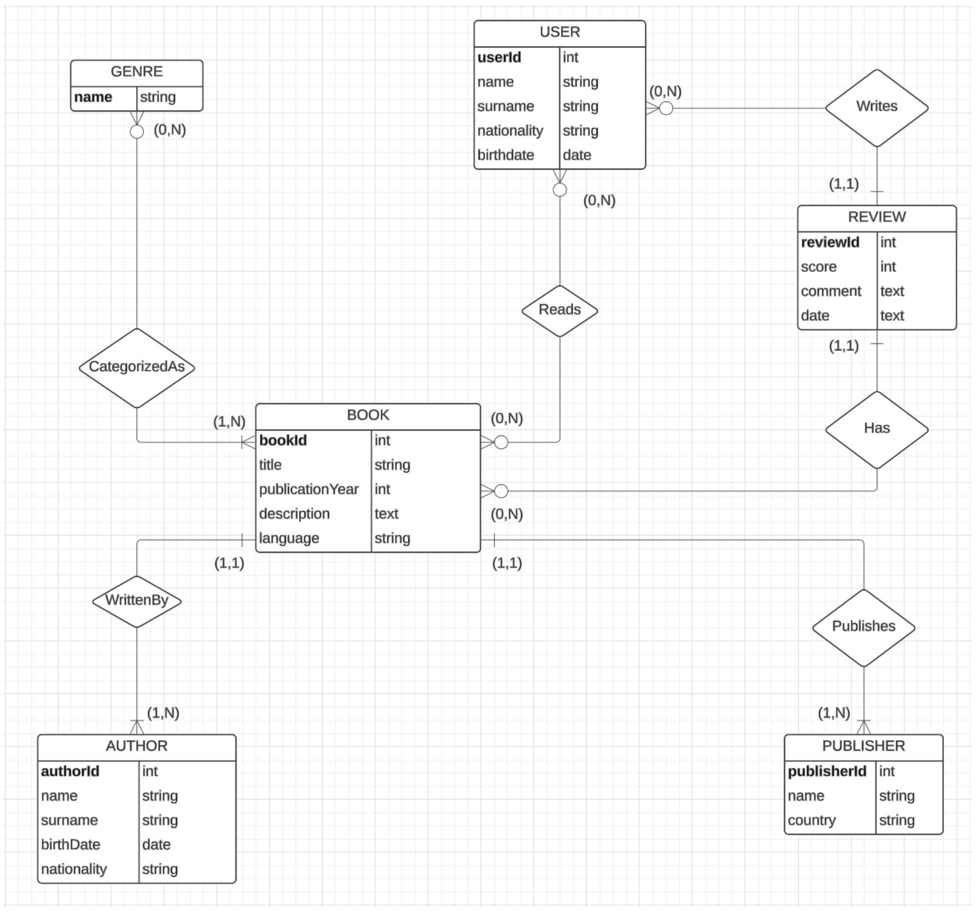


Figure 1: Library relational model

4 Workload

Q1: Given the book with ID 218619, determine the average score and all the book information.

Q2: Determine all books written by 737939 of the genre "Science fiction".

Q3: Find all books written by the author with ID 206880 along with their publication year.

Q4: Find all books published in 2017 in English.

Q5: Provide all the review information for the book with ID 350392, including the users who wrote them.

Q6: Given the user with ID 54, determine all the information about the books read by them and their genres.

Q7: Provide all the books published in the last year along with their publishers.

Q8: Show 10 books for the home page.

5 Aggregates methodology

5.1 Write the query in a formal and non-ambiguous way

Q1. (BOOK, [BOOK(bookId)_!], [BOOK_!, REVIEW(score)_H])

E = `BOOK`
LS = `[BOOK(bookId)_!]`
LP = `[BOOK_!, REVIEW(score)_H]`

Q2. (AUTHOR, [GENRE_WbCa, AUTHOR(authorId)_!], [BOOK(bookId)_Wb])

E = `AUTHOR`
LS = `[GENRE_WbCa, AUTHOR(authorId)_!]`
LP = `[BOOK(bookId)_Wb]`

Q3. (AUTHOR, [AUTHOR(authorId)_!], [BOOK(bookId, publicationYear)_Wb])

E = `AUTHOR`
LS = `[AUTHOR(authorId)_!]`
LP = `[BOOK(bookId, publicationYear)_Wb]`

Q4. (BOOK, [BOOK(publicationYear, language)_!], [BOOK(bookId)_!])

E = `BOOK`
LS = `[BOOK(publicationYear, language)_!]`
LP = `[BOOK(bookId)_!]`

Q5. (BOOK, [BOOK(bookId)_!], [REVIEW_H, USER(userId)_HW])

E = `BOOK`
LS = `[BOOK(bookId)_!]`
LP = `[REVIEW_H, USER(userId)_HW]`

Q6. (USER, [USER(userId)_!], [BOOK_R, GENRE_RCa])

E = `USER`
LS = `[USER(userId)_!]`
LP = `[BOOK_R, GENRE_RCa]`

Q7. (BOOK, [BOOK(publicationYear)_!], [BOOK(bookId)_!, PUBLISHER(publisherId)_P])

E = `BOOK`
LS = `[BOOK(publicationYear)_!]`
LP = `[BOOK(bookId)_!, PUBLISHER(publisherId)_P]`

Q8. (BOOK, [], [BOOK(bookId)_!])

E = `BOOK`
LS = `[]`
LP = `[BOOK(bookId)_!]`

5.2 Annotated the ER schema

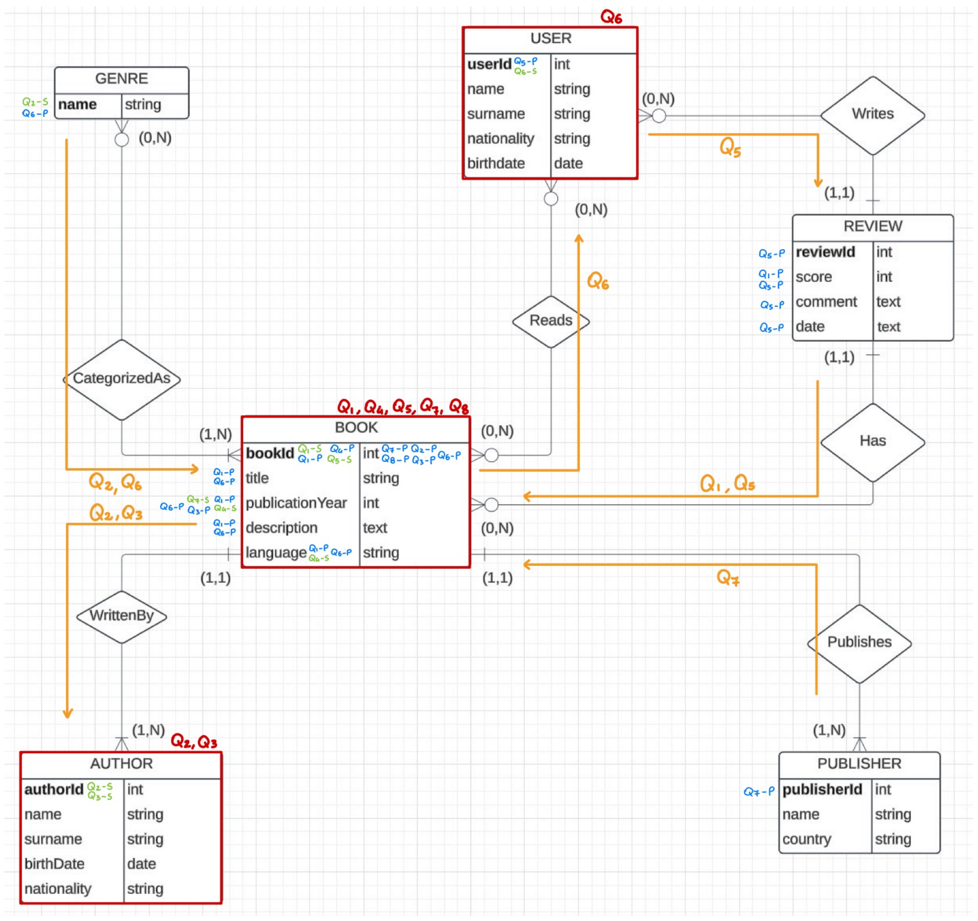


Figure 2: Annotated entity relationship diagram

5.3 Aggregates

```
Book: {
    bookId, title, publicationYear, description, language,
    reviews: [{ reviewId, score, comment, date, userId }],
    publisherId
}
```

```
Author: {
    authorId,
    books: [{
        bookId, publicationYear,
        genres: [{ name }]
    }]
}
```

```
User: {
    userId,
    books: [{
        bookId, title, publicationYear, description, language,
        genres: [{ name }]
    }]
}
```

6 Cassandra

6.1 Schema design

6.1.1 Book aggregate

Q1. **(BOOK, [BOOK(bookId)_!], [BOOK_!, REVIEW(score)_H])**

Q4. **(BOOK, [BOOK(publicationYear, language)_!], [BOOK(bookId)_!])**

Q5. **(BOOK, [BOOK(bookId)_!], [REVIEW_H, USER(userId)_HW])**

Q7. **(BOOK, [BOOK(publicationYear)_!],[BOOK(bookId)_!, PUBLISHER(publisherId)_P])**

Q8. **(BOOK, [], [BOOK(bookId)_!])**

Since the **LS sets** are disjoint, we define the following tables:

- Q1, Q5, Q8: Table with `BOOK(bookId)` as unique **partition key** and **primary key**.
- Q4, Q7: Table with `BOOK(publicationYear)` as **partition key** and `BOOK(language)` as **clustering column** together with `BOOK(bookId)` (identifier of the aggregate).
- Define a new type for the nested attribute `reviews` inside the aggregate.

Defining a new type for reviews:

```
CREATE TYPE IF NOT EXISTS Review_t (
    reviewId     int,
    score        int,
    comment      text,
    date         varchar,
    userId       int
);
```

Tables for BOOK:

```
CREATE TABLE IF NOT EXISTS Book1 (
    bookId          int PRIMARY KEY,
    title           varchar,
    publicationYear int,
    description     text,
    language        varchar,
    reviews         set<frozen<Review_t>>,
    publisherId     int
);
```

```
CREATE TABLE IF NOT EXISTS Book2 (
    bookId          int,
    title           varchar,
    publicationYear int,
    description     text,
    language        varchar,
    reviews         set<frozen<Review_t>>,
    publisherId     int,
    PRIMARY KEY (publicationYear, language, bookId)
);
```

6.1.2 Author aggregate

Q2. (AUTHOR, [GENRE_WbCa, AUTHOR(authorId)_!], [BOOK(bookId)_Wb])

Q3. (AUTHOR, [AUTHOR(authorId)_!], [BOOK(bookId, publicationYear)_Wb])

For the Author aggregate, given these two queries from the workload, we must pay attention.

There are no problems for Q3 since we can just create a table with AuthorId as the partition key (primary key), instead for Q2, due to the logical architecture of Cassandra we cannot perform the query.

In fact, for Q2 we should have to create a table with `GENRE(name)` as a clustering column, but this is not possible because `GENRE(name)` is a nested attribute in the aggregate, and that means we cannot specify conditions on `GENRE(name)`, neither creating some index nor specifying the `ALLOWING FILTERING` clause!

In this case, we should modify the aggregation in such a way we can perform the query efficiently but if we chose `GENRE` as the new aggregation entity, the same problem would persist on `AUTHOR(AuthorId)` because the cardinality is always **many to many**, so the problem of nested attribute persists.

Hence, we create a single table with `AUTHOR(AuthorId)` as the **partition key** (**primary key**) and to perform Q2 we must retrieve, at the application level, all the information that we need. In particular, we can obtain all the books written by a specific author, but we can filter by the genre only at the application level.

In short:

- for **Q3**, we can use `AUTHOR(authorId)` as **partition key**.
- **Q2** cannot directly be implemented in Cassandra due to the nested attribute `GENRE(name)`. We solve this by handling filtering at the **application level**.
- So `AUTHOR(authorId)` will be both **partition key** and **primary key**.

Defining a new type for books:

```
CREATE TYPE IF NOT EXISTS Book_at (
    bookId          int,
    publicationYear int,
    genres          set<text>
);
```

Table for AUTHOR:

```
CREATE TABLE IF NOT EXISTS Author (
    authorId    int PRIMARY KEY,
    books       set<frozen<Book_at>>
);
```

6.1.3 User aggregate

Q6. **(USER, [USER(userId)_!], [BOOK_R, GENRE_RCa])**

Since we have only one query, we create a single table with `USER(userId)` as the **partition key** (**primary key**) to perform this query efficiently.

Defining a new type for books:

```
CREATE TYPE IF NOT EXISTS Book_ut (
    bookId          int,
    title           varchar,
    publicationYear int,
    description     text,
    language        varchar,
    genres          set<text>
);
```

Table for USER:

```
CREATE TABLE IF NOT EXISTS User (
    userId     int PRIMARY KEY,
    books      set<frozen<Book_ut>>
);
```

6.2 Query Modeling in CQL

Q1. Given the book with ID `218619`, determine the average score and all the book information.

```
SELECT *
FROM   Book1
WHERE  bookId = 218619
```

The score is a nested attribute inside `reviews`, so we can only get the reviews of the book and compute the average of the score at the application level.

Q2. Determine all books written by `737939` of the genre "Science fiction".

```
SELECT books
FROM   Author
WHERE  authorId = 737939
```

As we already said, we cannot filter by genre, since it is a nested attribute. We have to perform this query to retrieve all the books written by the author `737939` and then filter by the genre at the application level.

Q3. Find all books written by the author with ID 206880 along with their publication year.

```sql
SELECT books
FROM   Author
WHERE  authorId = 206880
```

In this case, we also get the genres of all the books, because `books` is a nested attribute. If we are not interested in genres, we have to project just what we need at the application level.

Q4. Find all books published in 2017 in English.

```sql
SELECT bookId
FROM   Book2
WHERE  publicationYear = 2017 AND language = 'English'
```

Q5. Provide all the review information for the book with ID 350392, including the users who wrote them.

```sql
SELECT reviews
FROM   Book1
WHERE  bookId = 350392
```

Q6. Given the user ID 54, determine all the information about the books read by them and their genres.

```sql
SELECT books
FROM   User
WHERE  userId = 54
```

Q7. Provide all the books published in the last year along with their publishers.

```sql
SELECT bookId, publisherId
FROM   Book2
WHERE  publicationYear = 2024
```

Q8. Show 10 books for the home page.

```sql
SELECT bookId
FROM   Book1
LIMIT  10
```

7 Neo4j

7.1 Schema design

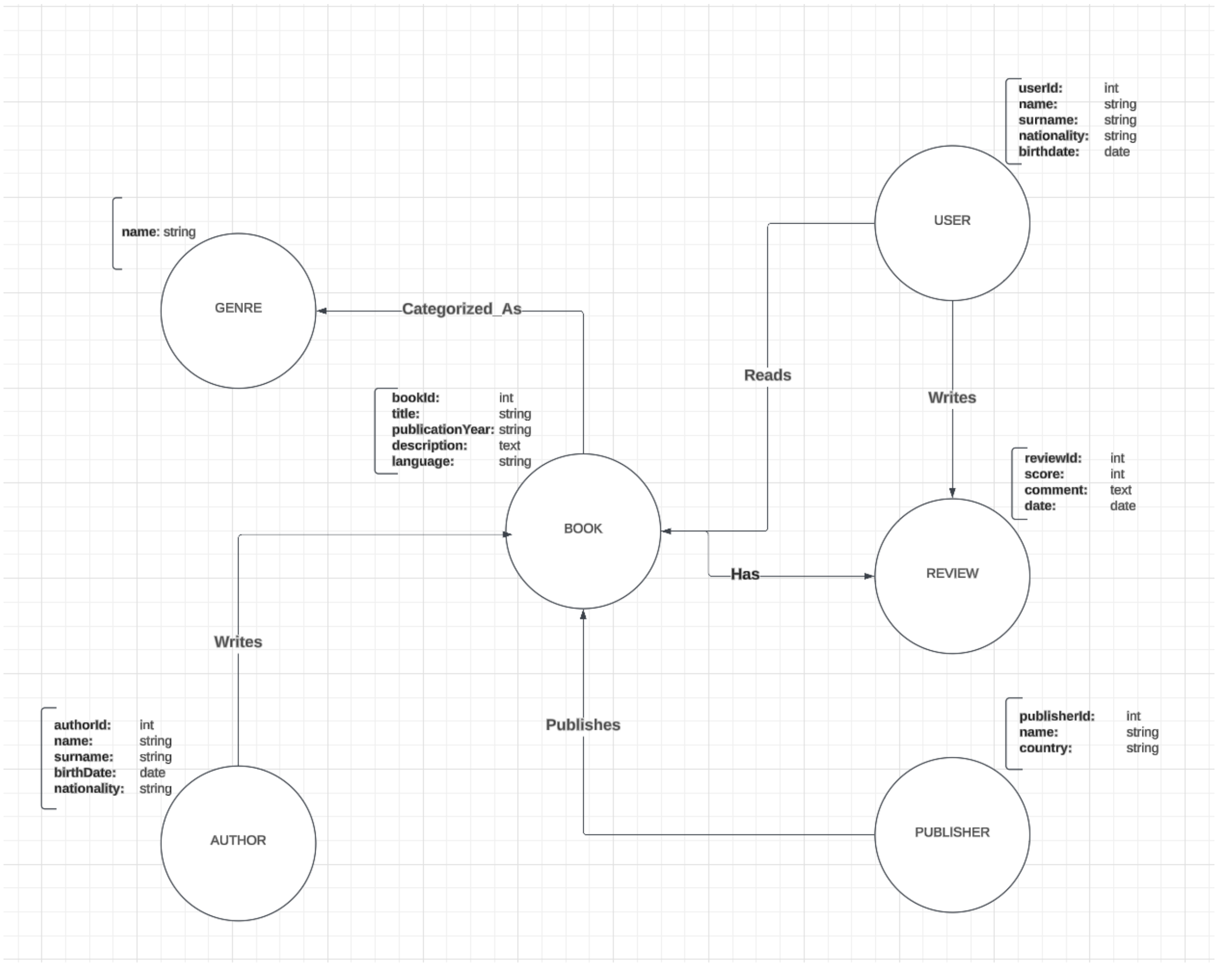


Figure 3: Neo4j graph schema

Graph-based NoSQL systems, such as Neo4j, are designed to handle complex, highly interconnected data.

Starting from our ER diagram we chose to model the graph in such a way we can handle all the queries in an efficient way.

There are many different ways to design the graph. For example, one might consider merging `genre` into the book node as an attribute. However, this would not be a good idea for our application, as each book can belong to multiple genres. Keeping `genre` as a separate node provides greater flexibility, allows for more complex queries, and prevents data duplication since the same genre node can be linked to multiple books at the same time.

One might also think to insert years or dates as labels for the relationship, but for our workload, we chose to construct the model avoiding useless traversal, even if they are very efficient in graph DB.

7.2 Query modeling in Cypher

Q1. Given the book with ID 1598407, determine the average score and all the book information.

```
MATCH   (b:Book{bookId:1598407})-[:Has]->(r:Review)
RETURN  b, AVG(r.score)
```

Q2. Determine all the books written by 110620 of the genre "Fiction".

```
MATCH   (:Genre{name:"Fiction"})<-[:Categorized_As]-(b:Book)
        <-[:Writes]-(:Author{authorId:110620})
RETURN  b.bookId
```

Q3. Find all the books written by the author with ID 402411, along with their publication year.

```
MATCH   (b: Book)<-[:Writes]-(:Author{authorId: 402411})
RETURN  b.bookId, b.publicationYear
```

Q4. Find all the books published in 2008 in English.

```
MATCH   (b:Book)
WHERE   b.publicationYear = 2008 AND b.language = "English"
RETURN  b.bookId
```

Q5. Provide all the review information for the book with ID 1598428, including the users who wrote them.

```
MATCH   (:Book{bookId:1598428})-[:Has]->(r:Review)<-[:Writes]-
        (u:User)
RETURN  r, u.userId
```

Q6. Given the user ID 24259, determine all the information about the books they read and their genres.

```
MATCH   (:User{userId:24259})-[:Reads]->(b:Book)-
        [:categorized_As]->(g:Genre)
RETURN  b, g
```

Q7. Provide all the books published in the last year along with their publishers.

```
MATCH   (b:Book{publicationYear:2024})<-[:Publishes]-
        (p:Publisher)
RETURN  b.bookId, p.publisherId
```

Q8. Show 10 books for the home page.

```
MATCH   (b:Book)
RETURN  b.bookId
LIMIT   10
```

8 Why Cassandra?

According to what we have already said from the beginning of this document, most of all considering point 2 and point 4, we are interested in availability more than consistency (an eventual consistency level is enough) and we decided to use Cassandra as our database for our application.

Cassandra, according to the CAP theorem, is focused on partition tolerance and availability, provided by its P2P structure, both key features of our application. It is the best suite for a read/write intensive application such as our library platform, and with its consistent hashing, it offers us a very high scalability to scale our platform for more and more new users. Talking about the workload, it might be quite complex for the Cassandra system because of its limits in querying, however, thanks to the speed in resolving queries, its high write throughput, fault tolerance, high availability, and the possibility of filtering the result at application level, Cassandra is the best choice for our application.

MongoDB is more flexible in solving queries because we can use nested attributes and lookups and there are fewer limitations concerning Cassandra. In fact, at the application level, everything is visible in Mongo.
Despite that, MongoDB is focused on consistency and partition tolerance (CP) and this is not what our application needs.

Neo4j is very fast and flexible and allows us to make complex queries with Cypher which is declarative and provides clauses to filter, return, create, delete, and set data; it is very fast thanks to its traversal speed, due to its index-free adjacency, where relationships between nodes are persistent and not calculated at query time.
Despite that, since Neo4j is well suited mostly for read-intensive and analytical applications, it is not the best database structure for our application, in fact, we need also to manage lots of writes from users.
For sure, it will be the best fit for our possible future implementation as we have already said in point 1: a recommendation system for our platform.

9 Cassandra configuration

For deploying our database, we relied on the docker features: we created a 3-node cluster using docker-compose.

The Docker Compose file sets up a Cassandra cluster with three nodes for fault-tolerant distributed database management with some key features:

- Persistent storage is defined for each node (cassandra-node-1, cassandra-node-2, cassandra-node-3) to ensure data durability even if containers are restarted.
- Each node has a health check that uses `nodetool status` to verify that the node is operational.
- The health check ensures dependent services (e.g., cassandra-2 depends on a healthy cassandra-1) only start when the required services are ready.
- The `restart: on-failure` directive ensures that any node will automatically restart if it crashes, contributing to fault tolerance by minimizing downtime.

The replication strategy is defined at the keyspace level. During the creation of keyspace, we can define the class and the replication factor that we want to use. In our case, we have decided to use the `SimpleStrategy` with a replication factor equal to 3 (the data are replicated in all three nodes) to ensure the maximum availability level.

10 Cassandra logical schema

10.1 Keyspace

```
CREATE KEYSPACE IF NOT EXISTS library
WITH REPLICATION = {'class':'SimpleStrategy', 'replication_factor':'1'};
```

10.2 Types

Three different types are used to perform queries efficiently:

- Review_t: 6.1.1 Book aggregate
- Book_at: 6.1.2 Author aggregate
- Book_ut: 6.1.3 User aggregate

10.3 Tables

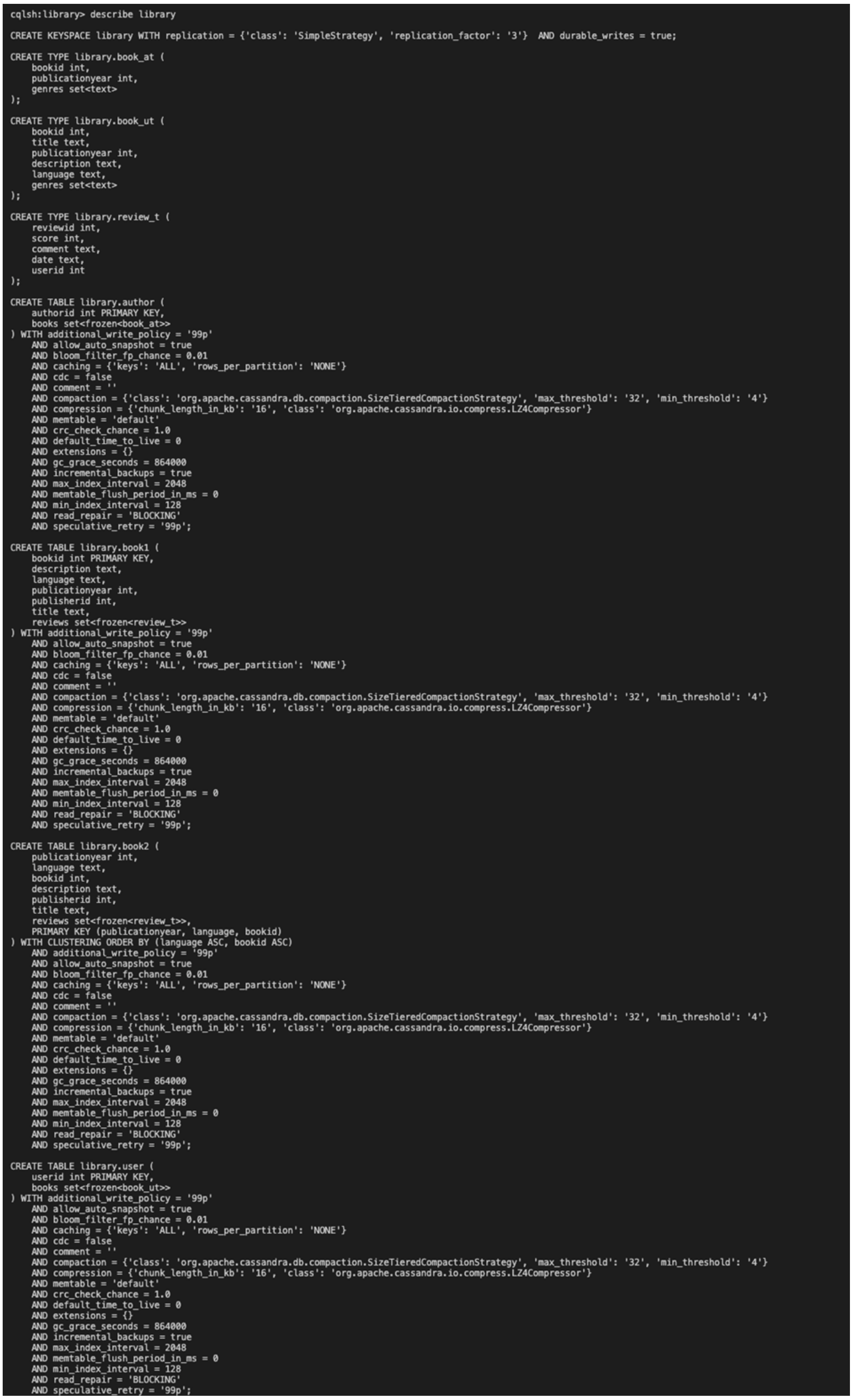
Four different tables to store efficiently all the data:

- Book1: 6.1.1 Book aggregate
- Book2: 6.1.1 Book aggregate
- Author: 6.1.2 Author aggregate
- User: 6.1.3 User aggregate

10.4 Copy the dataset from CSV files

```
COPY library.book1 (bookId, title, publicationYear, description,
     language, reviews, publisherId)
FROM '/csv/books.csv'
WITH HEADER = TRUE;

COPY library.book2 (bookId, title, publicationYear, description,
     language, reviews, publisherId)
FROM '/csv/books.csv'
WITH HEADER = TRUE;

COPY library.author (authorid, books)
FROM '/csv/authors.csv'
WITH HEADER = TRUE AND MINBATCHSIZE=1 AND MAXBATCHSIZE=1
                   AND PAGESIZE=10;

COPY library.user (userid, books)
FROM '/csv/users.csv'
WITH HEADER = TRUE AND MINBATCHSIZE=1 AND MAXBATCHSIZE=1
                   AND PAGESIZE=10;
```

11 Cassandra instance

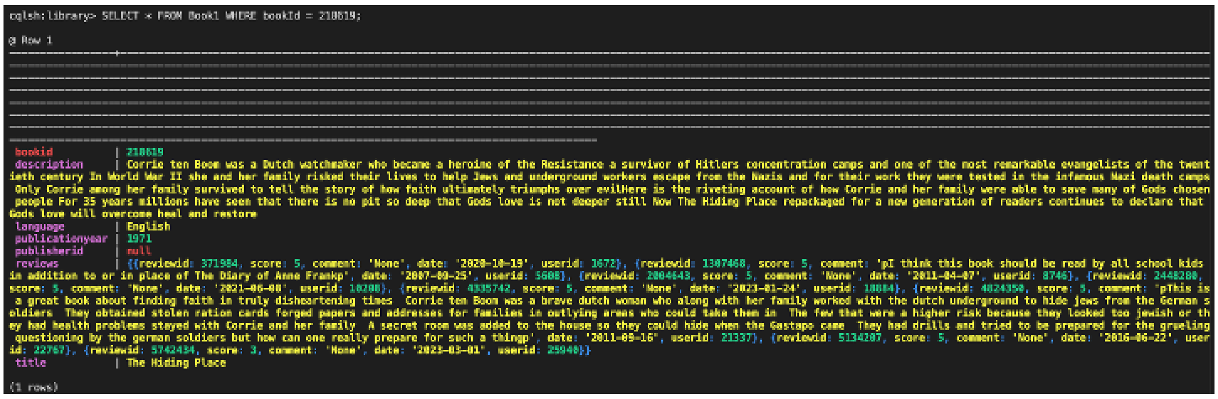


```
cqlsh:library> describe library

CREATE KEYSPACE library WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '3'}  AND durable_writes = true;

CREATE TYPE library.book_at (
    bookid int,
    publicationyear int,
    genres set<text>
);

CREATE TYPE library.book_ut (
    bookid int,
    title text,
    publicationyear int,
    description text,
    language text,
    genres set<text>
);

CREATE TYPE library.review_t (
    reviewid int,
    score int,
    comment text,
    date text,
    userid int
);

CREATE TABLE library.author (
    authorid int PRIMARY KEY,
    books set<frozen<book_at>>
) WITH additional_write_policy = '99p'
    AND allow_auto_snapshot = true
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND incremental_backups = true
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';

CREATE TABLE library.book1 (
    bookid int PRIMARY KEY,
    description text,
    language text,
    publicationyear int,
    publisherid int,
    title text,
    reviews set<frozen<review_t>>
) WITH additional_write_policy = '99p'
    AND allow_auto_snapshot = true
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND incremental_backups = true
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';

CREATE TABLE library.book2 (
    publicationyear int,
    language text,
    bookid int,
    description text,
    publisherid int,
    title text,
    reviews set<frozen<review_t>>,
    PRIMARY KEY (publicationyear, language, bookid)
) WITH CLUSTERING ORDER BY (language ASC, bookid ASC)
    AND additional_write_policy = '99p'
    AND allow_auto_snapshot = true
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND incremental_backups = true
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';

CREATE TABLE library.user (
    userid int PRIMARY KEY,
    books set<frozen<book_ut>>
) WITH additional_write_policy = '99p'
    AND allow_auto_snapshot = true
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND incremental_backups = true
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';
```

Figure 4: Library KEYSPACE

12 Workload implementation

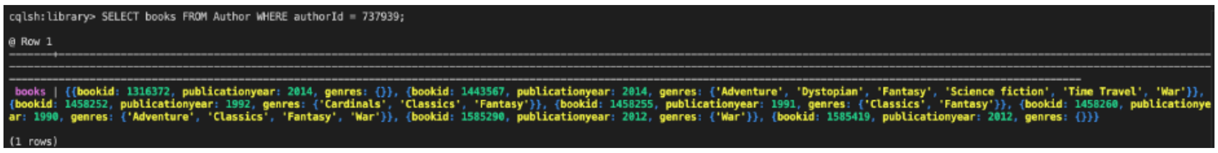
12.1 Q1



```
cqlsh:library> SELECT * FROM Book1 WHERE bookId = 210619;

@ Row 1
-----------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
 bookid          | 210619
 description     | Corrie ten Boom was a Dutch watchmaker who became a heroine of the Resistance a survivor of Hitlers concentration camps and one of the most remarkable evangelists of the twent
ieth century In World War II she and her family risked their lives to help Jews and underground workers escape from the Nazis and for their work they were tested in the infamous Nazi death camps
 Only Corrie among her family survived to tell the story of how faith ultimately triumphs over evilHere is the riveting account of how Corrie and her family were able to save many of Gods chosen
 people For 35 years millions have seen that there is no pit so deep that Gods love is not deeper still Now The Hiding Place repackaged for a new generation of readers continues to declare that
Gods love will overcome heal and restore
 language        | English
 publicationyear | 1971
 publisherid     | null
 reviews         | {{reviewid: 371984, score: 5, comment: 'None', date: '2020-10-19', userid: 1672}, {reviewid: 1307468, score: 5, comment: 'pI think this book should be read by all school kids
in addition to or in place of The Diary of Anne Frankp', date: '2007-09-25', userid: 5608}, {reviewid: 2004643, score: 5, comment: 'None', date: '2011-04-07', userid: 8746}, {reviewid: 2448280,
score: 5, comment: 'None', date: '2021-06-08', userid: 10208}, {reviewid: 4335742, score: 5, comment: 'None', date: '2023-01-24', userid: 18884}, {reviewid: 4824350, score: 5, comment: 'pThis is
 a great book about finding faith in truly disheartening times  Corrie ten Boom was a brave dutch woman who along with her family worked with the dutch underground to hide jews from the German s
oldiers  They obtained stolen ration cards forged papers and addresses for families in outlying areas who could take them in  The few that were a higher risk because they looked too jewish or th
ey had health problems stayed with Corrie and her family  A secret room was added to the house so they could hide when the Gestapo came  They had drills and tried to be prepared for the grueling
 questioning by the german soldiers but how can one really prepare for such a thingp', date: '2011-09-16', userid: 21337}, {reviewid: 5134207, score: 5, comment: 'None', date: '2016-06-22', user
id: 22767}, {reviewid: 5742434, score: 3, comment: 'None', date: '2023-03-01', userid: 25940}}
 title           | The Hiding Place

(1 rows)
```

Figure 5: First query

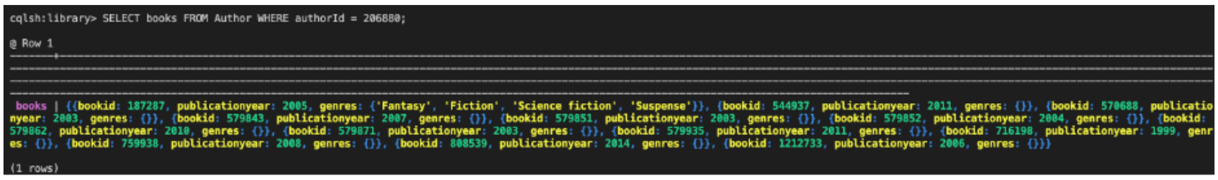
12.2 Q2



```
cqlsh:library> SELECT books FROM Author WHERE authorId = 737939;

@ Row 1
-------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 books | {{bookid: 1316372, publicationyear: 2014, genres: {}}, {bookid: 1443567, publicationyear: 2014, genres: {'Adventure', 'Dystopian', 'Fantasy', 'Science fiction', 'Time Travel', 'War'}},
{bookid: 1458252, publicationyear: 1992, genres: {'Cardinals', 'Classics', 'Fantasy'}}, {bookid: 1458255, publicationyear: 1991, genres: {'Classics', 'Fantasy'}}, {bookid: 1458260, publicationye
ar: 1990, genres: {'Adventure', 'Classics', 'Fantasy', 'War'}}, {bookid: 1585290, publicationyear: 2012, genres: {'War'}}, {bookid: 1585419, publicationyear: 2012, genres: {}}}

(1 rows)
```

Figure 6: Second query

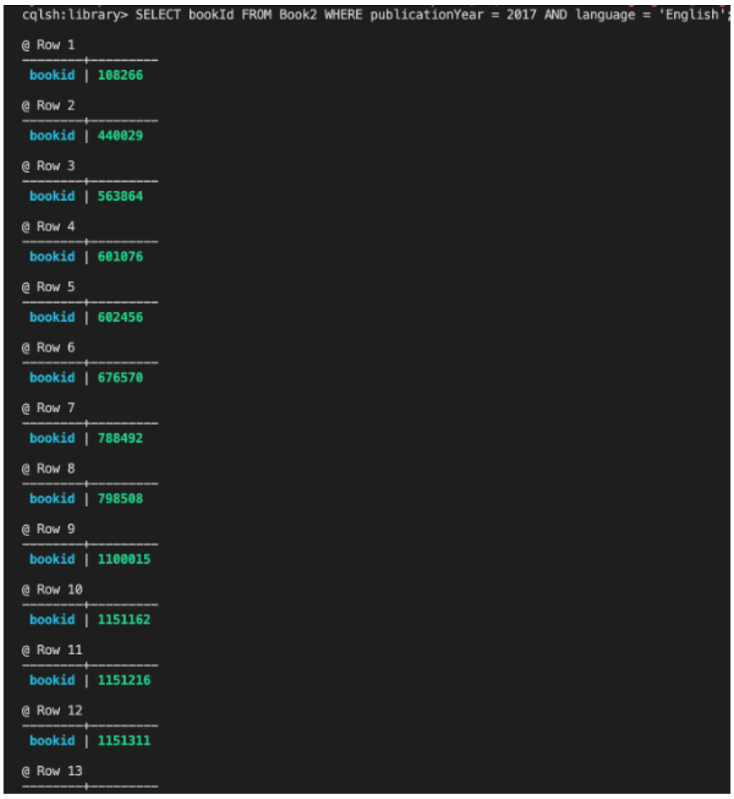
12.3 Q3



```
cqlsh:library> SELECT books FROM Author WHERE authorId = 206880;

@ Row 1
-------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
 books | {{bookid: 187287, publicationyear: 2005, genres: {'Fantasy', 'Fiction', 'Science fiction', 'Suspense'}}, {bookid: 544937, publicationyear: 2011, genres: {}}, {bookid: 570688, publicatio
nyear: 2003, genres: {}}, {bookid: 579843, publicationyear: 2007, genres: {}}, {bookid: 579851, publicationyear: 2003, genres: {}}, {bookid: 579852, publicationyear: 2004, genres: {}}, {bookid:
579862, publicationyear: 2010, genres: {}}, {bookid: 579871, publicationyear: 2003, genres: {}}, {bookid: 579935, publicationyear: 2011, genres: {}}, {bookid: 716198, publicationyear: 1999, genr
es: {}}, {bookid: 759938, publicationyear: 2008, genres: {}}, {bookid: 808539, publicationyear: 2014, genres: {}}, {bookid: 1212733, publicationyear: 2006, genres: {}}}

(1 rows)
```

Figure 7: Third query

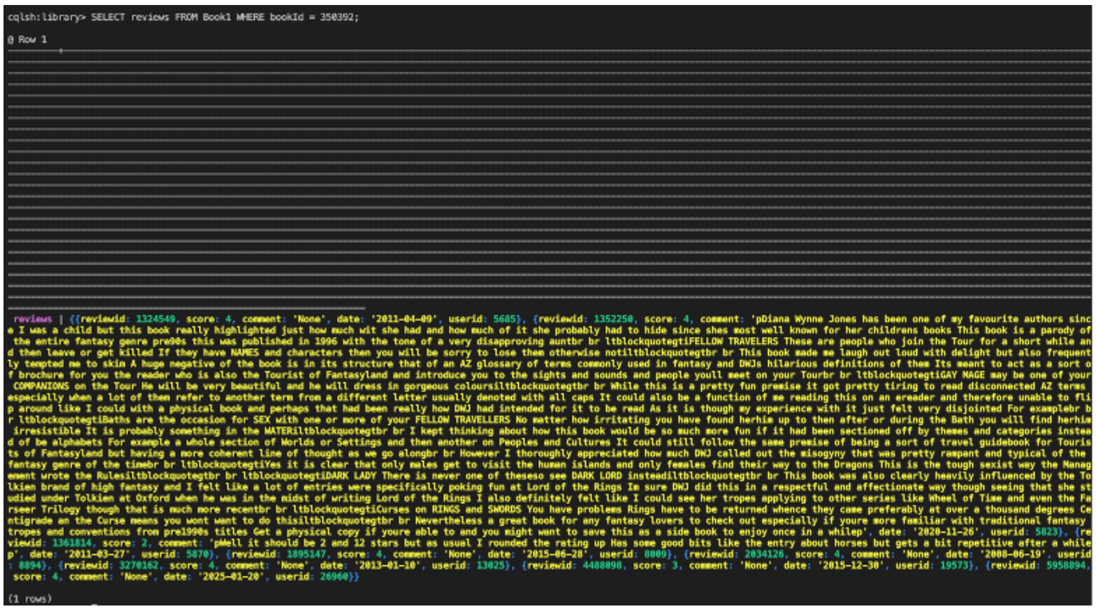
12.4 Q4



```
cqlsh:library> SELECT bookId FROM Book2 WHERE publicationYear = 2017 AND language = 'English';

@ Row 1
--------+---------
 bookid | 108266

@ Row 2
--------+---------
 bookid | 440029

@ Row 3
--------+---------
 bookid | 563864

@ Row 4
--------+---------
 bookid | 601076

@ Row 5
--------+---------
 bookid | 602456

@ Row 6
--------+---------
 bookid | 676570

@ Row 7
--------+---------
 bookid | 788492

@ Row 8
--------+---------
 bookid | 798508

@ Row 9
--------+---------
 bookid | 1100015

@ Row 10
--------+---------
 bookid | 1151162

@ Row 11
--------+---------
 bookid | 1151216

@ Row 12
--------+---------
 bookid | 1151311

@ Row 13
--------+---------
```

Figure 8: Fourth query

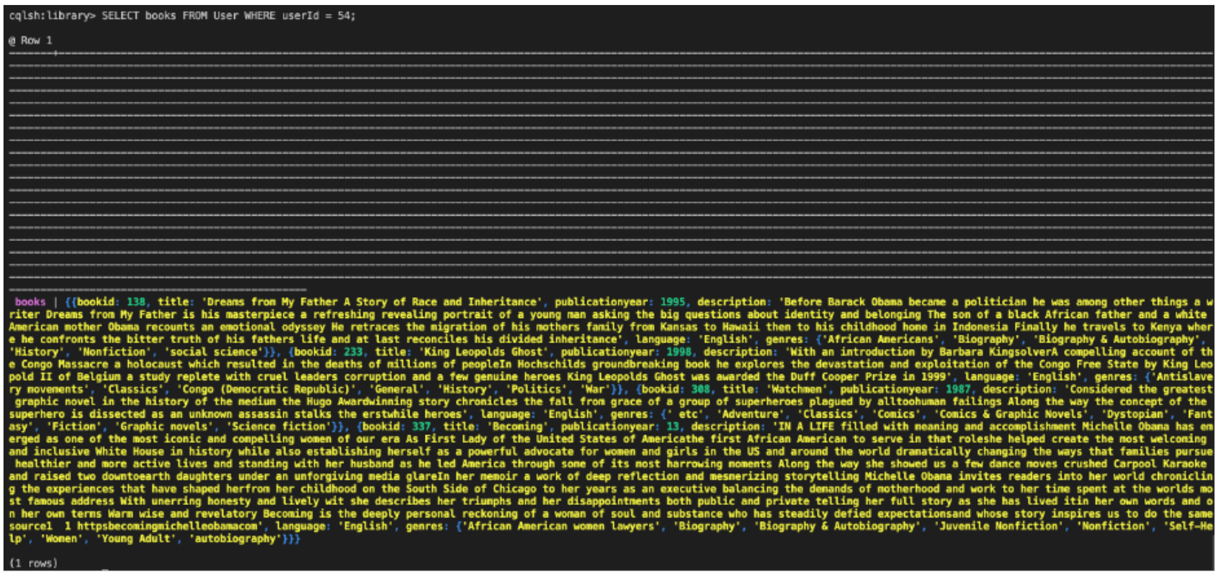
12.5 Q5



```
cqlsh:library> SELECT reviews FROM Book1 WHERE bookId = 350392;

@ Row 1
---------+--------------------------------------------------------------------------------------------------------
 reviews | {{reviewid: 1324549, score: 4, comment: 'None', date: '2011-04-09', userid: 5685}, {reviewid: 1352250, score: 4, comment: 'pDiana Wynne Jones has been one of my favourite authors sinc
e I was a child but this book really highlighted just how much wit she had and how much of it she probably had to hide since shes most well known for her childrens books This book is a parody of
 the entire fantasy genre pre90s this was published in 1996 with the tone of a very disapproving auntbr br ltblockquotegtiFELLOW TRAVELERS These are people who join the Tour for a short while an
d then leave or get killed If they have NAMES and characters then you will be sorry to lose them otherwise notiltblockquotegtbr br This book made me laugh out loud with delight but also frequent
ly tempted me to skim A huge negative of the book is in its structure that of an AZ glossary of terms commonly used in fantasy and DWJs hilarious definitions of them Its meant to act as a sort o
f brochure for you the reader who is also the Tourist of Fantasyland and introduce you to the sights and sounds and people youll meet on your Tourbr br ltblockquotegtiGAY MAGE may be one of your
 COMPANIONS on the Tour He will be very beautiful and he will dress in gorgeous coloursiltblockquotegtbr br While this is a pretty fun premise it got pretty tiring to read disconnected AZ terms
especially when a lot of them refer to another term from a different letter usually denoted with all caps It could also be a function of me reading this on an ereader and therefore unable to fli
p around like I could with a physical book and perhaps that had been really how DWJ had intended for it to be read As it is though my experience with it just felt very disjointed For examplebr b
r ltblockquotegtiBaths are the occasion for SEX with one or more of your FELLOW TRAVELLERS No matter how irritating you have found herhim up to then after or during the Bath you will find herhim
 irresistible It is probably something in the WATERiltblockquotegtbr br I kept thinking about how this book would be so much more fun if it had been sectioned off by themes and categories instea
d of be alphabets For example a whole section of Worlds or Settings and then another on Peoples and Cultures It could still follow the same premise of being a sort of travel guidebook for Touris
ts of Fantasyland but having a more coherent line of thought as we go alongbr br However I thoroughly appreciated how much DWJ called out the misogyny that was pretty rampant and typical of the
fantasy genre of the timebr br ltblockquotegtiYes it is clear that only males get to visit the human islands and only females find their way to the Dragons This is the tough sexist way the Manag
ement wrote the Rulesiltblockquotegtbr br ltblockquotegtiDARK LADY There is never one of theseso see DARK LORD insteadiltblockquotegtbr br This book was also clearly heavily influenced by the To
lkien brand of high fantasy and I felt like a lot of entries were specifically poking fun at Lord of the Rings Im sure DWJ did this in a respectful and affectionate way though seeing that she st
udied under Tolkien at Oxford when he was in the midst of writing Lord of the Rings I also definitely felt like I could see her tropes applying to other series like Wheel of Time and even the Fa
rseer Trilogy though that is much more recentbr br ltblockquotegtiCurses on RINGS and SWORDS You have problems Rings have to be returned whence they came preferably at over a thousand degrees Ce
ntigrade an the Curse means you wont want to do thisiltblockquotegtbr br Nevertheless a great book for any fantasy lovers to check out especially if youre more familiar with traditional fantasy
tropes and conventions from pre1990s titles Get a physical copy if youre able to and you might want to save this as a side book to enjoy once in a whilep', date: '2020-11-26', userid: 5823}, {re
viewid: 1361814, score: 2, comment: 'pWell it should be 2 and 12 stars but as usual I rounded the rating up Has some good bits like the entry about horses but gets a bit repetitive after a while
p', date: '2011-03-27', userid: 5870}, {reviewid: 1895147, score: 4, comment: 'None', date: '2015-06-28', userid: 8009}, {reviewid: 2034126, score: 4, comment: 'None', date: '2008-06-19', userid
: 8894}, {reviewid: 3270162, score: 4, comment: 'None', date: '2013-01-10', userid: 13025}, {reviewid: 4488098, score: 3, comment: 'None', date: '2015-12-30', userid: 19573}, {reviewid: 5958894,
 score: 4, comment: 'None', date: '2025-01-20', userid: 26960}}

(1 rows)
```

Figure 9: Fifth query

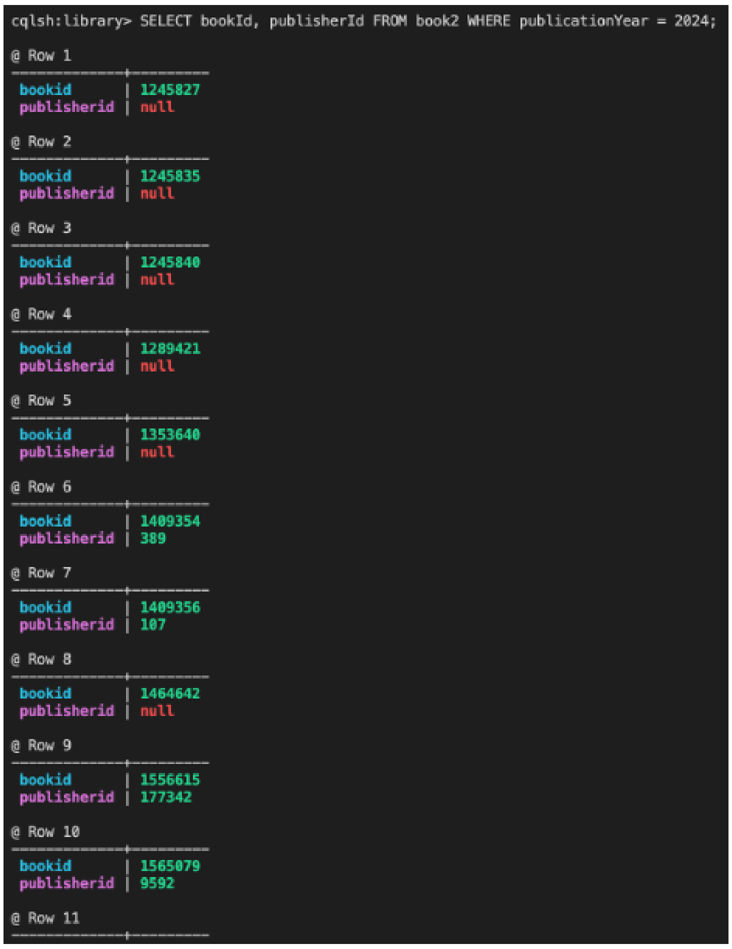
12.6 Q6



```
cqlsh:library> SELECT books FROM User WHERE userId = 54;

@ Row 1
-------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------
 books | {{bookid: 138, title: 'Dreams from My Father A Story of Race and Inheritance', publicationyear: 1995, description: 'Before Barack Obama became a politician he was among other things a w
riter Dreams from My Father is his masterpiece a refreshing revealing portrait of a young man asking the big questions about identity and belonging The son of a black African father and a white
American mother Obama recounts an emotional odyssey He retraces the migration of his mothers family from Kansas to Hawaii then to his childhood home in Indonesia Finally he travels to Kenya wher
e he confronts the bitter truth of his fathers life and at last reconciles his divided inheritance', language: 'English', genres: {'African Americans', 'Biography', 'Biography & Autobiography',
'History', 'Nonfiction', 'social science'}}, {bookid: 233, title: 'King Leopolds Ghost', publicationyear: 1998, description: 'With an introduction by Barbara KingsolverA compelling account of th
e Congo Massacre a holocaust which resulted in the deaths of millions of peopleIn Hochschilds groundbreaking book he explores the devastation and exploitation of the Congo Free State by King Leo
pold II of Belgium a study replete with cruel leaders corruption and a few genuine heroes King Leopolds Ghost was awarded the Duff Cooper Prize in 1999', language: 'English', genres: {'Antislave
ry movements', 'Classics', 'Congo (Democratic Republic)', 'General', 'History', 'Politics', 'War'}}, {bookid: 308, title: 'Watchmen', publicationyear: 1987, description: 'Considered the greatest
 graphic novel in the history of the medium the Hugo Awardwinning story chronicles the fall from grace of a group of superheroes plagued by alltoohuman failings Along the way the concept of the
superhero is dissected as an unknown assassin stalks the erstwhile heroes', language: 'English', genres: {' etc', 'Adventure', 'Classics', 'Comics', 'Comics & Graphic Novels', 'Dystopian', 'Fant
asy', 'Fiction', 'Graphic novels', 'Science fiction'}}, {bookid: 337, title: 'Becoming', publicationyear: 13, description: 'IN A LIFE filled with meaning and accomplishment Michelle Obama has em
erged as one of the most iconic and compelling women of our era As First Lady of the United States of Americathe first African American to serve in that roleshe helped create the most welcoming
and inclusive White House in history while also establishing herself as a powerful advocate for women and girls in the US and around the world dramatically changing the ways that families pursue
 healthier and more active lives and standing with her husband as he led America through some of its most harrowing moments Along the way she showed us a few dance moves crushed Carpool Karaoke
and raised two downtoearth daughters under an unforgiving media glareIn her memoir a work of deep reflection and mesmerizing storytelling Michelle Obama invites readers into her world chroniclin
g the experiences that have shaped herfrom her childhood on the South Side of Chicago to her years as an executive balancing the demands of motherhood and work to her time spent at the worlds mo
st famous address With unerring honesty and lively wit she describes her triumphs and her disappointments both public and private telling her full story as she has lived itin her own words and o
n her own terms Warm wise and revelatory Becoming is the deeply personal reckoning of a woman of soul and substance who has steadily defied expectationsand whose story inspires us to do the same
source1  1 httpsbecomingmichelleobamacom', language: 'English', genres: {'African American women lawyers', 'Biography', 'Biography & Autobiography', 'Juvenile Nonfiction', 'Nonfiction', 'Self-He
lp', 'Women', 'Young Adult', 'autobiography'}}}

(1 rows)
```

Figure 10: Sixth query

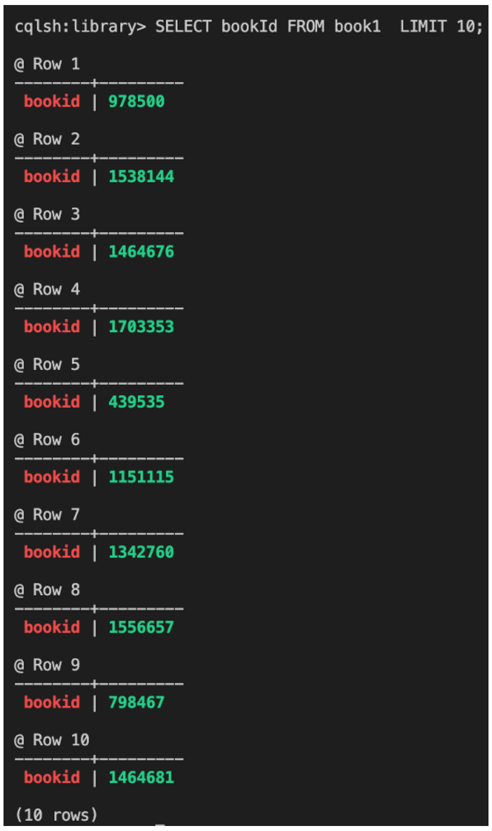
12.7 Q7



```
cqlsh:library> SELECT bookId, publisherId FROM book2 WHERE publicationYear = 2024;

@ Row 1
-------------+---------
 bookid      | 1245827
 publisherid | null

@ Row 2
-------------+---------
 bookid      | 1245835
 publisherid | null

@ Row 3
-------------+---------
 bookid      | 1245840
 publisherid | null

@ Row 4
-------------+---------
 bookid      | 1289421
 publisherid | null

@ Row 5
-------------+---------
 bookid      | 1353640
 publisherid | null

@ Row 6
-------------+---------
 bookid      | 1409354
 publisherid | 389

@ Row 7
-------------+---------
 bookid      | 1409356
 publisherid | 107

@ Row 8
-------------+---------
 bookid      | 1464642
 publisherid | null

@ Row 9
-------------+---------
 bookid      | 1556615
 publisherid | 177342

@ Row 10
-------------+---------
 bookid      | 1565079
 publisherid | 9592

@ Row 11
-------------+---------
```

Figure 11: Seventh query

12.8 Q8



```
cqlsh:library> SELECT bookId FROM book1  LIMIT 10;

@ Row 1
--------+---------
 bookid | 978500

@ Row 2
--------+---------
 bookid | 1538144

@ Row 3
--------+---------
 bookid | 1464676

@ Row 4
--------+---------
 bookid | 1703353

@ Row 5
--------+---------
 bookid | 439535

@ Row 6
--------+---------
 bookid | 1151115

@ Row 7
--------+---------
 bookid | 1342760

@ Row 8
--------+---------
 bookid | 1556657

@ Row 9
--------+---------
 bookid | 798467

@ Row 10
--------+---------
 bookid | 1464681

(10 rows)
```

Figure 12: Eighth query

13 Conclusions

In this report, we explored the architecture of a distributed database system using Apache Cassandra. We defined a query model, designed and implemented dataset loading from CSV files, and discussed best practices for system implementation. Additionally, we examined using Docker to create a multi-node Cassandra cluster with replication for high availability and fault tolerance.
We also analyzed and implemented several queries for extracting data from the database, considering Cassandra's NoSQL characteristics. The solutions for handling complex queries, such as filtering and aggregation, were tailored to the system's specific features and business needs.

Future Perspectives:
As we already discussed, looking ahead, we could explore the implementation of a recommendation system using graph databases like Neo4j. This would allow for more efficient handling of relationships between users and items, enabling personalized recommendations based on complex graph structures.