

Database Management Systems

Database System for Library

Introduction:

Knowledge is the key to success, and one can get knowledge from different resources. Still the leading resource of scholarly books, the library is the best way to find any book related to anything.

The above statement is so correct that in this piece of a research paper, we used case **3**, which is related to the library. A library is a place where we used to get lots of books simultaneously of different genres and not only books in the library, but we also have other things. A library's collection includes books, periodicals, newspapers, manuscripts, films, maps, prints, documents, etc.

Books are not the only things connected to the library with books; several readers usually come to read the books or publish the book from the library. That there are several eBooks and magazines in the library. In the library, there are so many things that a person is not easy to do simultaneously, so in this paper, we made a library database by understanding things happening in the library. Creating tables like books, magazines, publishers, readers, and good records stored library information and books and other things information.

Use Case 3- Library Management System:

Entity Relational Model

Entities:

Here we have used the third Use case of Library Management System and after carefully going through the details of the use case we were able to come up with the list of entities, the relationships and the necessary attributes in the library management system.

Entities:

Readers
Books
eBooks
Magazines
Publishers
Fine
Staff
Authentication System

Relations:

Borrows
Publish
Downloads
Reserve
Keep Track of
Pays

Attributes:

Names of Reader
Addresses
Registration Dates
Title of Books
Author
Year of publish
Genre
No. of Pages
Issue Number
URL
Name of Publisher
Country
Phone number
Email Address

Cardinality Constraints:

One to Many relationships-

A book/magazine can have **only one** publisher.

Many to Many relationships-

can download an **unlimited** number of e-books

Participation Constraints:

Optional Participation-

A reader **may** borrow/reserve a book or magazine.

Mandatory Participation-

A book/magazine **must be** published by a Publisher.

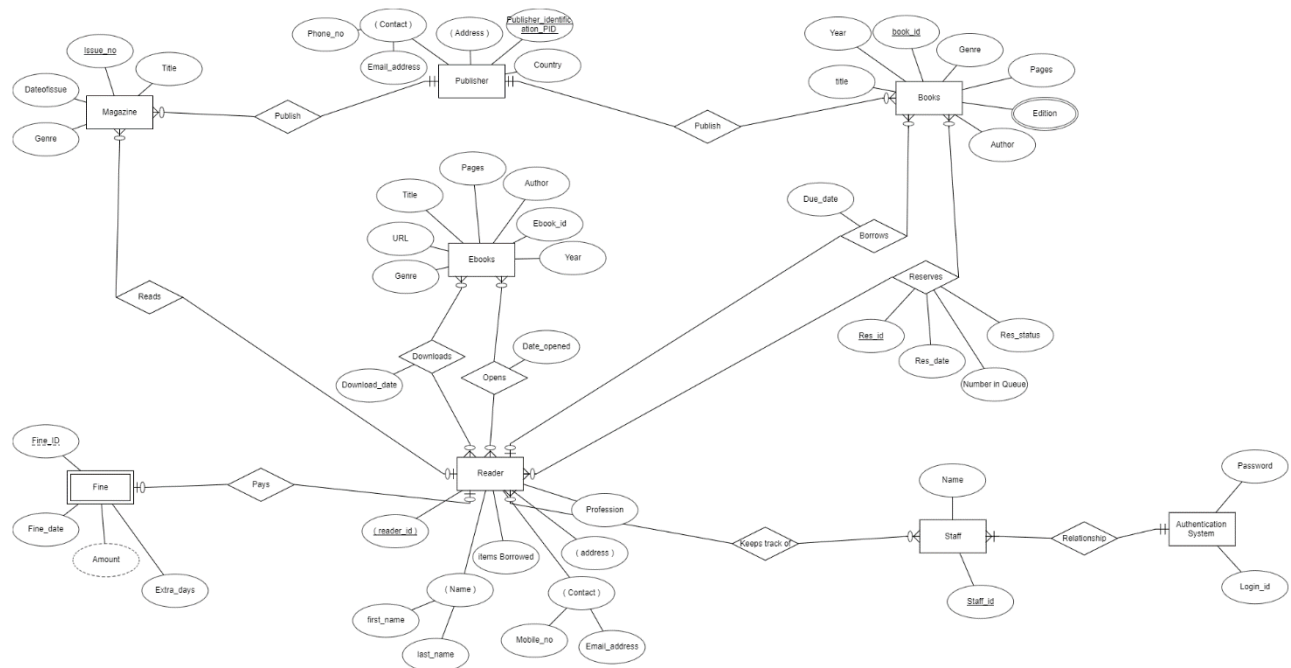


Figure 1: Entity Relational Diagram

Relational Schema:

In the next task we have converted the ER diagram of Library Management System into Relational Schema. (Geek for Geeks)

Now let us explain the steps we followed to convert the ER diagram to Relational Schema-

Step 1: We have created 5 main entities as Reader, Books, Magazine, ebooks and publisher. And then created relationships for all of the entities also added attributes for the respective entities and chose one attribute as a primary key in RS.

Reader(reader_id, Name, Contact, Address, Profession

Books(book_id, title, author, pages, genre, year

Magazines(Issue_no, title, genre, DateofIssue

Ebooks(ebook_id, title, url, genre, year, author

Publisher(Publisher_identification, country, address, Contact

Step 2: We have one weak entity which is fine which is dependent on the date of return. So we converted the weak into a relation and added a foreign.

Reader(reader_id, Name, Contact, Address, Profession

Books(book_id, title, author, pages, genre, year

Magazines(Issue_no, title, genre, DateofIssue

Ebooks(ebook_id, title, url, genre, year, author

Publisher(Publisher_identification, country, address, Contact

Fine(Fine_id, fine date, extra days, Reader_id*

Step 3: We don't have one to one relationship for Mandatory participation so skipped this step.

Step 4: We have 3 one to many relationships and they are Publish, Borrows and Reads. For them, we need to put the primary key of the relation on the one side into the relation on the many side. Eg. For Reads the one side is reader and primary key is Reader_id and many side is magazine so we put Reader_id onto other side as a foreign key. The same process is repeated for Borrows and Publish relationships.

Reader(reader_id, Name, Contact, Address, Profession

Books(book_id, title, author, pages, genre, year, Reader_id*, Publisher_identification*

Magazines(Issue_no, title, genre, DateofIssue, Publisher_identification*

Ebooks(ebook_id, title, url, genre, year, author

Publisher(Publisher_identification, country, address, Contact

Fine(Fine_id, fine date, extra days, Reader_id*

Step 5: We have 3 many to many relationships and they are Downloads, Opens and Reserves. Here the same process is repeated for Opens and Reserves.

Reader(reader_id, Name, Contact, Address, Profession

Books(book_id, title, author, pages, genre, year, Reader_id*, Publisher_identification*

Magazines(Issue_no, title, genre, DateofIssue, Publisher_identification*

Ebooks(ebook_id, title, url, genre, year, author

Publisher(Publisher_identification, country, address, Contact

Fine(Fine_id, fine date, extra days, Reader_id*

Downloads(Reader_id*, Ebook_id*

Opens(Reader_id*, Ebook_id*

Reserves(Reader_id*, book_id*

Step 6: There is one multi-valued attribute, Edition of the Books. For this, we need to create a separate relation, put this attribute in it, and add the primary key of Books there as a foreign key. The primary key will be the combination of both attributes.

Reader(reader_id, Name, Contact, Address, Profession)

Books(book_id, title, author, pages, genre, year, Reader_id*, Publisher_identification*)

Magazines(Issue_no, title, genre, DateofIssue, Publisher_identification*)

Ebooks(ebook_id, title, url, genre, year, author)

Publisher(Publisher_identification, country, address, Contact)

Fine(Fine_id, fine date, extra days, Reader_id*)

Downloads(Reader_id*, Ebook_id*)

Opens(Reader_id*, Ebook_id*)

Reserves(Reader_id*, book_id*)

Book_Edition(Edition, Book_id*)

This is the final result,

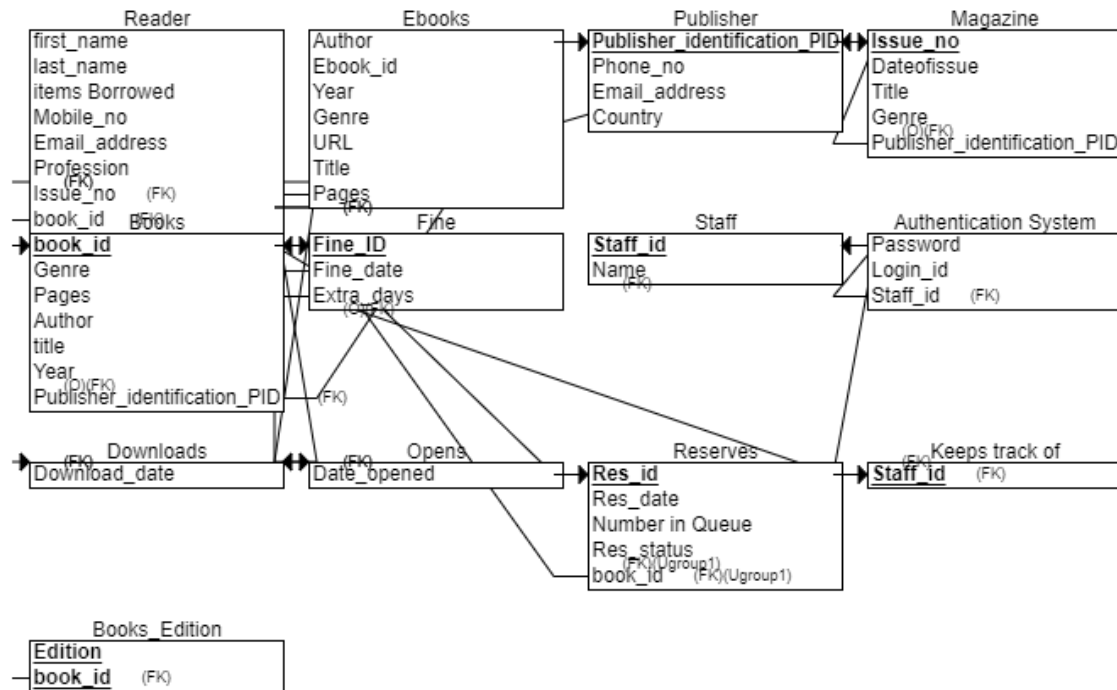


Figure 2: Relational Schema

The above use case is implemented across three different technologies: SQL, Neo4j and MongoDB.

Let us study the technologies and their uses in detail.

SQL:

Introduction:

SQL, or Structured Query Language, is a valuable resource for data workers. It is without a doubt the most important language for obtaining a career in data analysis or data sciences. Every minute, millions of data points are generated, and raw data has no story to tell. Professionals use SQL to extract data from databases after it has been stored for further study. SQL is the most commonly used language for extracting and organizing data from relational databases. A database is a table made up of rows and columns. SQL is the database language. It makes it easier to retrieve specific information from databases, which can then be analyzed. Even when the research is conducted on a different platform, such as Python or R, SQL would be needed to extract the data you need from a company's database.

SQL manages a big amount of data, especially when a large amount of data is being written concurrently and there are too many data transactions. (Yishan Li, Sathiamoorthy Manoharan, 2013); An Oracle database (Oracle) is a grouping of data treated as a single unit. The purpose of a database is to store and retrieve important information. A database server is required to solve information management problems. In a multiuser scenario, a server consistently processes a large volume of data, allowing numerous users to access the same data at the same time. All of this is accomplished while maintaining excellent performance. A database server also protects against unauthorized access and offers effective failure recovery solutions. The database has logical and physical structures. Because the physical and logical structures are different, physical data storage can be restricted while logical storage structure access remains unaffected. (Alan Beaulieu, Sanjay Mishra, 2004)

Below are the steps in which we have designed our Database corresponding to our ER diagram in SQL.

Creating SQL Database

Creation of Reader Tables:

“The below query refers to the creation of reader table where the primary key is Reader with subsequently all the attributes are shown below it with their corresponding data types.”

```
create table Reader(  
  
    reader_id char(4)primary key,  
  
    firstname varchar2(8),  
  
    lastname varchar(8),
```

```
items_borrowed number(4),  
mobile char(10),  
email varchar(20),  
profession varchar(10),  
fine_id char(3)  
);
```

Inserting values into the Reader table:

```
insert into Reader
```

```
values('A111', 'John', 'Hopes', 2, '9898989898', 'john@gmail.com', 'student', '000');
```

```
insert into Reader
```

```
values('M112', 'Lewis', 'smith', 0, '9894489898', 'lewis@gmail.com', 'm_student', '000');
```

```
insert into Reader
```

```
.
```

```
.
```

```
.
```

select * from Reader;

Below is the snapshot of the reader table created:

	READER_ID	FIRSTNAME	LASTNAME	ITEMS_BORROWED	MOBILE	EMAIL	PROFESSION	FINE_ID
1	A111	John	Hopes		2 9898989898	john@gmail.com	student	000
2	M112	Lewis	smith		0 9894489898	lewis@gmail.com	m student	000
3	M113	Sid	Hopes		3 9898339898	sid@gmail.com	m student	121
4	M114	oliver	Brown		1 9898966898	oliver@gmail.com	m student	000
5	A115	olivia	Devis		4 9898779898	olivia@gmail.com	student	000
6	A116	sophia	Jhonson		1 9898988898	sophia@gmail.com	student	000
7	A117	lily	Robert		1 9898922898	lily@gmail.com	student	897
8	T118	JOHNY	Devis		2 9894589898	johny@gmail.com	Assistant	000
9	T119	maia	smith		0 9898559898	maia@gmail.com	Assistant	000
10	P112	malik	green		2 9898669898	malik@gmail.com	professor	543
11	P111	nishant	sharma		1 9897789898	nishant@gmail.com	professor	000
12	P113	tushar	yadav		4 9897889898	tushar@gmail.com	professor	000

Creation of books table:

The below query refers to the creation of books table where the primary key is book_id with subsequently all the attributes are shown below it with their corresponding data types.

```
create table Books(
    book_id char(4) not null primary key,
    title varchar(20),
    author varchar2(20),
    publishing date,
    No_of_copies number(4),
    genere varchar(10),
    reader_id char(4),
    FOREIGN KEY (reader_id) REFERENCES Reader(reader_id)
);
```

Inserting values into the Reader table:

```
insert into Books
```

```
values('bf01', 'The great Gatsby', 'F.ScottFitzgerald', '10-JAN-18', 1000 , 'Tragedy', 'A115');
```


insert into Books

values('bf02', 'Brave New World', 'Aldous Huxley', '10-JAN-32', 102, 'Science', '');

insert into Books

.
.
.

select * from books;

Below is the snapshot of the books table created:

BOOK_ID	TITLE	AUTHOR	PUBLISHING	NO_OF_COPIES	GENRE	READER_ID
1bf01	The great Gatsby	F.ScottFitzgerald	10-JAN-18	1000	Tragedy	A115
2bf02	Brave New World	Aldous Huxley	10-JAN-32	102	Science	(null)
3bf03	The Graphic Novel	Margaret Atwood	01-FEB-85	45	Tragedy	T118
4ed01	Make it Stick	Henry Roediger	14-APR-14	10	StudyGuide	M112
5ed02	Artificial Int	Peter Norvig	13-DEC-94	19	Science	M114
6ed03	Practical Statistics	Andrew Bruce	10-MAY-17	100	Science	P111
7ed04	The C Programming	Brian Kernighan	10-JAN-78	19	Science	P113
8ph02	Machine Learning	Alex	20-JAN-15	19	Science	(null)
9ph77	JAVA	F.Scott	10-OCT-19	12	Coding	A115

Creation of magazine table:

The below query refers to the creation of magazine table where the primary key is issue_no with subsequently all the attributes are shown below it with their corresponding data types.

create table magazine(

issue_no char(4) not null primary key,

title varchar(20),

date_of_issue Date,

genere varchar(10),

reader_id char(4),

FOREIGN KEY (reader_id) REFERENCES Reader(reader_id)

);

Inserting values into the Magazine table:

insert into magazine

values('mg01', 'Wired', '01-JAN-21', 'Technology','A115');

insert into magazine

values('mg02', 'PgMag', '05-JUL-19', 'Technology','A111');

insert into magazine

.

.

.

select * from magazine;

Below is the snapshot of the Magazine table created:

	ISSUE_NO	TITLE	DATE_OF_ISSUE	GENERE	READER_ID
1	mg01	Wired	01-JAN-21	Technology	A115
2	mg02	PgMag	05-JUL-19	Technology	A111
3	mg03	Retro Gamer	28-MAR-21	Technology	M112
4	ms01	BBC Wildlife	11-JAN-17	Science	A116
5	ms02	The Lancet	22-DEC-19	Science	M114
6	ms03	All About Space	09-SEP-18	Science	A115
7	mf01	Apex Magazine	22-SEP-22	Ficton	T119
8	mf02	Locus	28-NOV-17	Ficton	P111
9	mf03	Lightspeed	06-MAY-22	Ficton	P113
10	ma01	Apollo	29-SEP-13	Art	T118
11	ma02	Wallpaper	05-APR-15	Art	M114
12	ma03	Antiques	31-MAR-19	Art	T119

Creation of Publisher Table:

The below query refers to the creation of publisher table where the primary key is publisher_identification with subsequently all the attributes are shown below it with their corresponding data types.

create table publisher(

publisher_identification varchar(4)not null primary key,

```

Publisher_name char(50),
phone_no char(15),
email_address varchar(100),
country varchar(25)
);

```

Inserting values into the Publisher table:

```
insert into publisher
```

```
values('P101','Wordsworth-Edition Limited','0120909203','enquiries@wordsworth-
editions.com','United Kingdom');
```

```
insert into publisher
```

```
values('P102','Penguin Random
House','012091234','enquiries@penguinrandomhouse.com','United States');
```

```
insert into publisher
```

```

.
.
.
.

```

```
select * from publisher;
```

Below is the snapshot of the Publisher table created:

	PUBLISHER_IDENTIFICATION	PUBLISHER_NAME	PHONE_NO	EMAIL_ADDRESS	COUNTRY
1	P101	Wordsworth-Edition Limited	0120909203	enquiries@wordsworth-editions.com	United Kingdom
2	P102	Penguin Random House	012091234	enquiries@penguinrandomhouse.com	United States
3	P103	Pearson Limited	0120128470	enquiries@pearson.com	United Kingdom
4	P104	O-reilly Media	012038475	enquiries@oreilly.com	United States
5	P105	Harvard University Press	012012349	enquiries@harvardunipress.com	United States

Creation of EBooks Table:

The below query refers to the creation of ebooks table where the primary key is Ebook_id with subsequently all the attributes is shown below it with their corresponding data types.

```
create table Ebooks(
```

```

Ebook_id char(5) not null primary key,
title varchar(50),
url_ varchar(300),
publish_year date,
author varchar2(20),
No_of_pages int,
genere varchar(15),
reader_id char(4),
FOREIGN KEY (reader_id) REFERENCES Reader(reader_id)
);

```

Inserting values into the EBooks table:

```
insert into EBooks
```

```
values('EB001', 'The Keeper of Stories', 'https://www.kobo.com/gb/en/ebook/the-keeper-of-
stories-1', '05-JAN-1996', 'Sally Page', 343 , 'Adventure','A115');
```

```
insert into EBooks
```

```
values('EB002', 'Pieces of Her', 'https://www.kobo.com/gb/en/ebook/pieces-of-her-the-stunning-
new-psychological-crime-thriller-from-the-no-1-global-bestselling-suspense-author', '10-JAN-
1987', 'Karin Slaughter', 522 , 'Romantic','');
```

```
insert into EBooks
```

```

.
.
.
.

```

```
select * from EBooks;
```

Below is the snapshot of the Ebooks table created:

BOOK_ID	TITLE	URL	PUBLISH_YEAR	AUTHOR	NO_OF_PAGES	GENRE	READER_ID
1 EB001	The Keeper of Stories	https://www.kobo.com/gb/en/eb...	05-JAN-96	Sally Page	343	Adventure	A115
2 EB002	Pieces of Her	https://www.kobo.com/gb/en/eb...	10-JAN-87	Karin Slaughter	522	Romantic	(null)
3 EB003	The Midnight Lock	https://www.kobo.com/gb/en/eb...	01-FEB-85	Jeffery Deaver	454	Mystery	T118
4 EB004	Jailhouse Lawyer	https://www.kobo.com/gb/en/eb...	14-APR-14	James Patterson	1001	Mystery	M112
5 EB005	God is Dead	https://www.kobo.com/gb/en/eb...	13-DEC-94	Andy Mcgrath	199	Biography	M114
6 EB006	The Kaiju Preservation	https://www.kobo.com/gb/en/eb...	10-MAY-17	Kaiju Preservation	450	science fiction	P111
7 EB007	All the White Friends I Couldnt Keep	https://www.kobo.com/gb/en/eb...	10-JAN-78	The White Friends	519	Non-Fiction	P113
8 EB008	Mission Total Resilience	https://www.kobo.com/gb/en/eb...	20-JAN-15	Mission Total	519	Kids	(null)
9 EB009	My Part in his Downfall	https://www.kobo.com/gb/en/eb...	10-OCT-19	Spike Milligan	312	History	A115

Creation of Publish Table:

The below query refers to the creation of publish table where two foreign keys are added one from Books table and other from Publisher table.

```
create table publish(
book_id CHAR(4),
publisher_identification varchar(4),
FOREIGN KEY (Book_id) REFERENCES BOOKS(book_id),
FOREIGN KEY (publisher_identification) REFERENCES publisher(publisher_identification)
);
```

Inserting values into the Publish table:

```
insert into publish
values('bf01', 'P101');
insert into publish
values('bf02', 'P101');
insert into publish
.
.
.
.
select * from publish;
```

Below is the snapshot of the Publish table created:

	BOOK_ID	PUBLISHER_IDENTIFICATION
1	bf01	P101
2	bf02	P101
3	bf03	P102
4	ed01	P102
5	ed02	P103
6	ed03	P104
7	ed04	P104
8	ph02	P105
9	ph77	P105

Explanation of the above database created:

In this database, we have taken Reader, Books, Ebooks, Magazine, Publisher and Publish tables. Now in a Library Management System, the most important entities are readers, the books which are present in the library, which include books, magazines and ebooks available in the library. The additional entity is the Publisher, as it contains details about the Publisher who publishes the books. Here, the Publisher data is necessary if the readers need to find the books from the publisher's information. Here the Publisher entity is supported by a weak entity of Publish to link the data of books with the publisher.

Now the choices we have made regarding the entities for the Library Management System fits the characteristics because the Library system consists of readers information for keeping track of the data for each book borrowed by the reader or for reserving the books required or for the reader details for the book borrowed.

Test Cases:

- 1) **Show all the names of the reader who has no fine and who are Masters's student or Assistant professor.**

SQL Query:

```
select firstname,lastname from reader where fine_id = '000' and (profession = 'm_student' or profession = 'Assistant');
```

Snapshot of Result:

	FIRSTNAME	LASTNAME
1	Lewis	smith
2	oliver	Brown
3	JOHny	Devis
4	maia	smith

- 2) Show all the names of the reader who are professor & they have issued 'C-Programming' book.

SQL Query:

```
select * from reader A join books B on A.reader_id = B.reader_id
where A.profession = 'professor';
```

Snapshot of Result:

READER_ID	FIRSTNAME	LASTNAME	ITE...	MOBILE	EMAIL	PROFESSION	FINE_ID	BOOK_ID	TITLE	AUTHOR	PUBLISHING	NO_OF_COPIES	GENRE	READER_ID_1
1P111	nishant sharma		19897789898	nishant@...	professor	000	ed03	Practical Statistics	Andrew Bruce	10-MAY-17	100	Science	P111	
2P113	tushar yadav		49897889898	tushar@g...	professor	000	ed04	The C Programming	Brian Kernighan	10-JAN-78	19	Science	P113	

- 3) Show all the magazine who have issues before January 2020 & the genre is Science.

SQL Query:

```
select title from magazine where date_of_issue < '01-JAN-20' and genere = 'Science';
```

Snapshot of Result:

	TITLE
1	BBC Wildlife
2	The Lancet
3	All About Space

4) Show all the publisher names which start with 'PE'.

SQL Query:

```
select Publisher_name from publisher where Publisher_name like 'Pe%';
```

Snapshot of Result:

PUBLISHER_NAME
1 Penguin Random House
2 Pearson Limited

5) Show all the info of readers who reads books published by UK based publishers.

SQL Query:

```
select * from reader where reader_id IN(  
select reader_id from books where book_id IN(  
select book_id from publish where publisher_identification IN (  
select publisher_identification from publisher where country = 'United Kingdom')));
```

Snapshot of Result:

READER_ID	FIRSTNAME	LASTNAME	ITEMS_BORROWED	MOBILE	EMAIL	PROFESSION	FINE_ID
1 M114	oliver	Brown	1	9898966898	oliver@gmail.com	m student	000
2 A115	olivia	Devis	4	9898779898	olivia@gmail.com	student	000

NEO4j

Introduction

In the last few years, there has been an explosion of innovative database paradigms. Previously, the relational database management system (RDBMS), as exemplified by Microsoft's SQLServer or Oracle's MySQL, was really the de facto option for database users. In this article, we'll dwell deeper into Neo4j (Neo4j, 2022), the Graph Database, to see what it can do and offer some examples of how it might be used.

What is Graph?

Graph databases, as the name implies, arrange data in the shape of a graph using graph theory as a mathematical basis. A graph can be thought of as a collection of nodes and edges in its most basic form. Edges are used to represent the relationships between nodes, whereas nodes are used to represent the relations between nodes. What makes this valuable for databases is that nodes can carry data that describes the item, but edges can also hold data that describes the nature and details of the relationship. That information could be as basic as the sort of relationship or it could be really complex.

Enter Neo4j

Neo4j is the most popular graph database engine, supporting ACID transactions as well as native graph data storage and processing. It is available in both a free Open-Source and a commercial Enterprise licenced edition. The open-source version only supports one node, whereas the corporate version includes clustering and hot backups. Both versions employ the Cypher query language, which uses a 'ASCII-art' style syntax to make graph data storage and querying as easy and logical as feasible.

Creating NEO4j Database

To begin, we'll need to establish our graph, presuming you have Neo4j (Neo4j, n.d.) installed or are using the online sandbox they provide. We have a few options here: We can load data from a file like CSV or JSON, or we can utilise Neo4j's built-in ETL functionality to load data from a JDBC linked RDBMS.

To start with the project, we have to create nodes (Neo4j Team.)

Creating reader nodes:

The below query refers to the creation of reader table with subsequently all the attributes are shown below it with their corresponding data types.

“In Neo4j the nodes and the attributes of nodes are input within the same queries”

```
create(:reader{reader_id:'A111',firstname:'JOhn',lastname:'Hopes',items_borrowed:
'2',mobile:'9898989898',email:'john@gmail.com',profession:'student',fine_id:'000'}
)
```

```
create(:reader{reader_id:'M112',firstname:'Lewis',lastname:'smith',items_borrowed
:'0',mobile:'9894489898',email:'lewis@gmail.com',profession:'m_student',fine_id:'
000'})
```

.

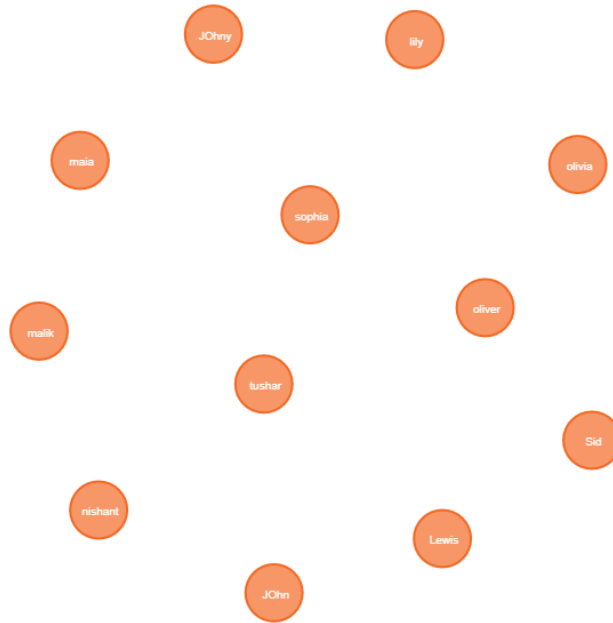
.

Below is the snapshot of the reader nodes.

Reader Nodes

Query:

```
match(r:reader) return r
```



Creation of Books nodes:

The below query refers to the creation of books table with subsequently all the attributes are shown below it with their corresponding data types.

```
create(:books{book_id:'bf01',title:'The great  
Gatsby',author:'F.ScottFitzgerald',publishing:'10-JAN-  
18',No_of_copies:'1000',genre:'Tragedy',reader_id:'A115'})
```

```
create(:books{book_id:'bf02',title:'Brave New World',author:'Aldous  
Huxley',publishing:'10-JAN-32',No_of_copies:'102',genre:'Science',reader_id:''})
```

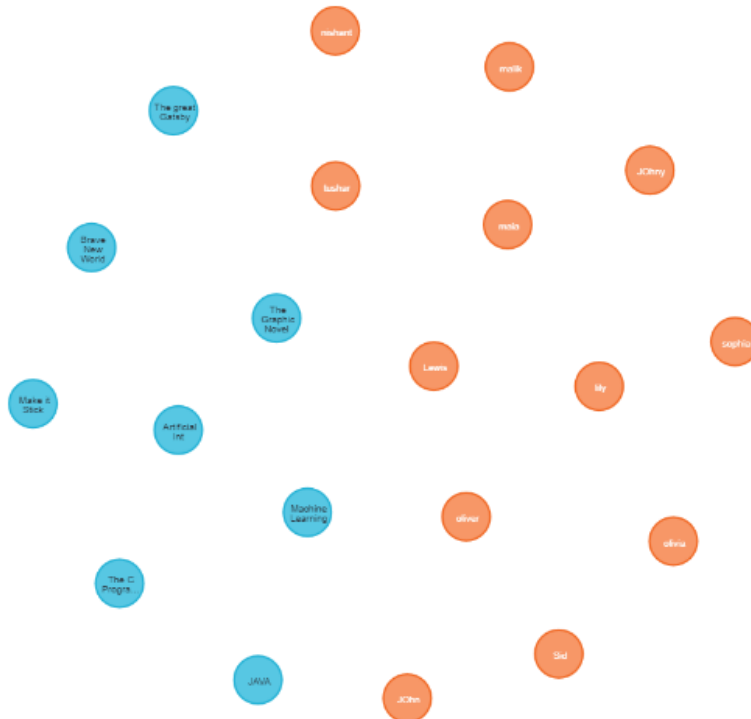
.
. .
.

Below is the snapshot of the books nodes.

Books Nodes

Query:

```
match(b:books) return b
```



Here Blue colour represents Books and orange colour represents reader

Creation of Magazine nodes:

The below query refers to the creation of magazine nodes with subsequently all the attributes are shown below it with their corresponding data types.

```
create(:magazine{ magazine_id:'mg01',title:'Wired',date_of_issue:'01-JAN-21',genre:'Technology',reader_id:'A115'})
```

```
create(:magazine{ magazine_id:'mg02',title:'PgMag',date_of_issue:'05-JUL-19',genre:'Technology',reader_id:'A111'})
```

.

.

.

Below is the snapshot of the magazine nodes

magazine Nodes

query:

```
match(m:magazine) return m
```


House',phone_no:'012091234',email_address:'enquiries@penguinrandomhouse.com',country:'United states'})

.

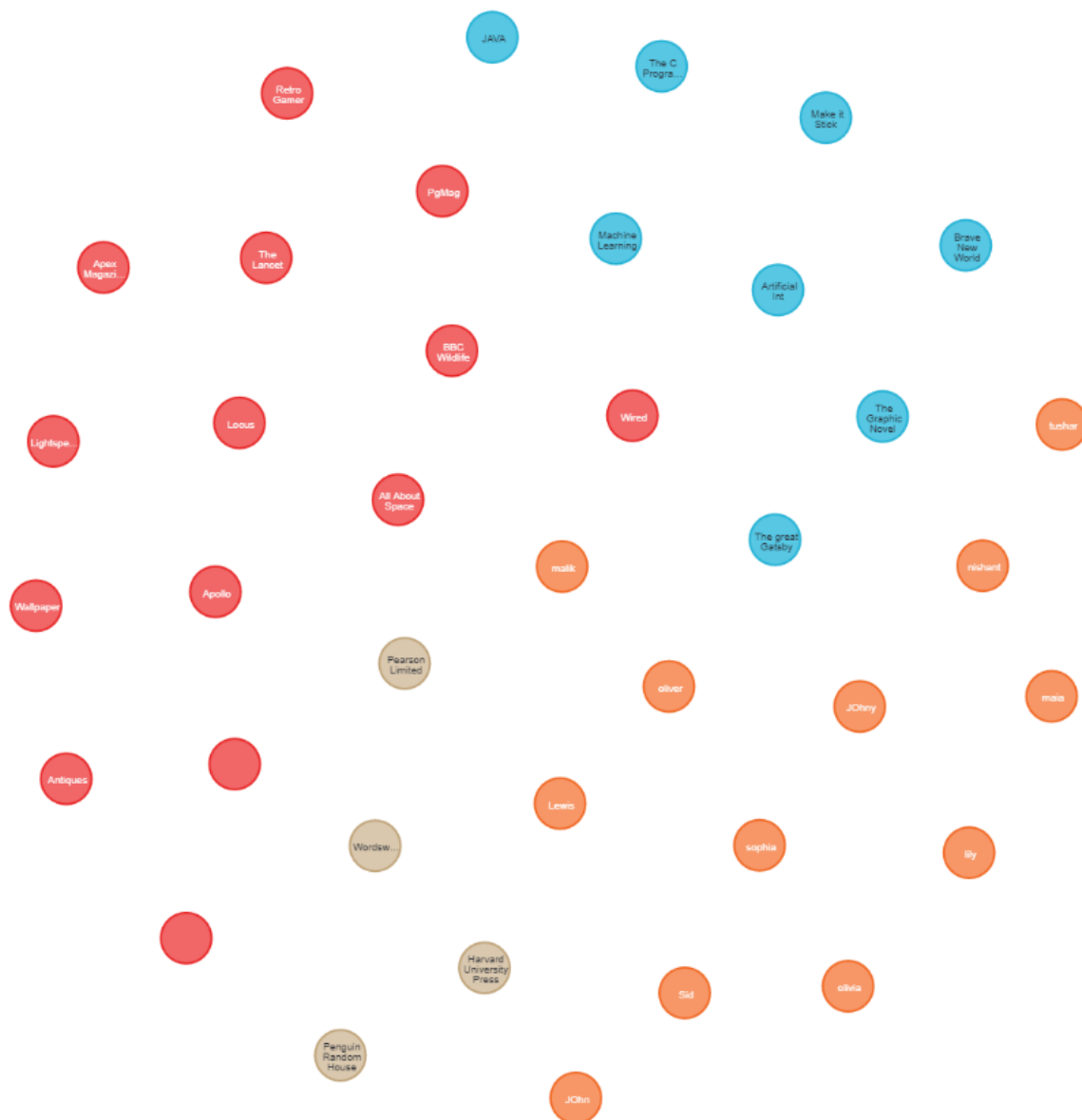
.

Below is the snapshot of the magazine nodes

Publisher Nodes

Query:

```
match(p:publisher) return p
```



Blue colour nodes denote books, orange colour nodes denote reader, red colour nodes denote magazine and light brown colour nodes denote publisher

After creating nodes, we have created the relation between the nodes

Relation Between Books and Publisher:

“Relationship of books and publisher is connection of those books which is published by its publisher”

Relationship 1-

```
Match(b:books),(p:publisher) where b.book_id='bf01' and p.publisher_identification='P101'  
CREATE (b)-[bp:published]->(p) return bp
```

Below is the snapshot of the relationship

Relationship

2-



```
Match(b:books),(p:publisher) where b.book_id='bf02' and p.publisher_identification='P101'  
CREATE (b)-[bp:published]->(p) return bp
```

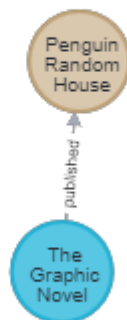
Below is the snapshot of the relationship



Relationship 3-

Match(b:books),(p:publisher) where b.book_id='bf03' and p.publisher_identification='P102'
CREATE (b)-[bp:published]->(p) return bp

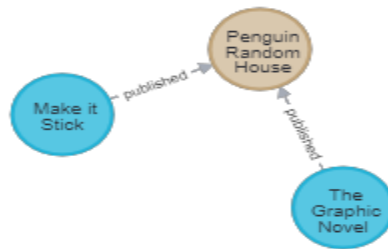
Below is the snapshot of the relationship



Relationship 4-

Match(b:books),(p:publisher) where b.book_id='ed01' and p.publisher_identification='P102'
CREATE (b)-[bp:published]->(p) return bp

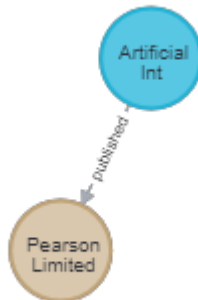
Below is the snapshot of the relationship



Relationship 5-

Match(b:books),(p:publisher) where b.book_id='ed02' and p.publisher_identification='P103'
CREATE (b)-[bp:published]->(p) return bp

Below is the snapshot of the relationship



Relation between book and reader:

Relationship 1-

Match(b:books),(r:reader) where b.book_id='bf01' and r.reader_id='A115' CREATE (r)-[br:reads]->(b) return br

Below is the snapshot of the relationship



Relationship 2-

Match(b:books),(r:reader) where b.book_id='bf03' and r.reader_id='T118' CREATE (r)-[br:reads]->(b) return br

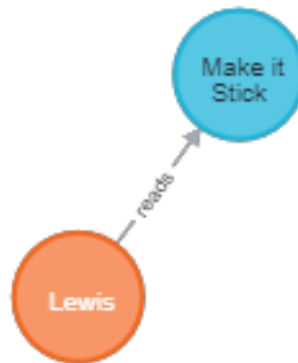
Below is the snapshot of the relationship



Relationship 3-

Match(b:books),(r:reader) where b.book_id='ed01' and r.reader_id='M112' CREATE (r)-[br:reads]->(b) return br

Below is the snapshot of the relationship



Relationship 4-

Match(b:books),(r:reader) where b.book_id='ed02' and r.reader_id='M114' CREATE (r)-[br:reads]->(b) return br

Below is the snapshot of the relationship

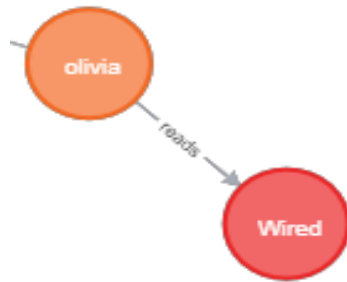


Relation between magazine and reader:

Relationship 1-

Match(m:magazine),(r:reader) where m.magazine_id='mg01' and r.reader_id='A115' CREATE (r)-[br:reads]->(m) return br

Below is the snapshot of the relationship



Relationship 2-

Match(m:magazine),(r:reader) where m.magazine_id='mg02' and r.reader_id='A111' CREATE (r)-[br:reads]->(m) return br

Below is the snapshot of the relationship



Relationship 3-

Match(m:magazine),(r:reader) where m.magazine_id='mg03' and r.reader_id='M112' CREATE (r)-[br:reads]->(m) return br

Below is the snapshot of the relationship



Relationship 4-

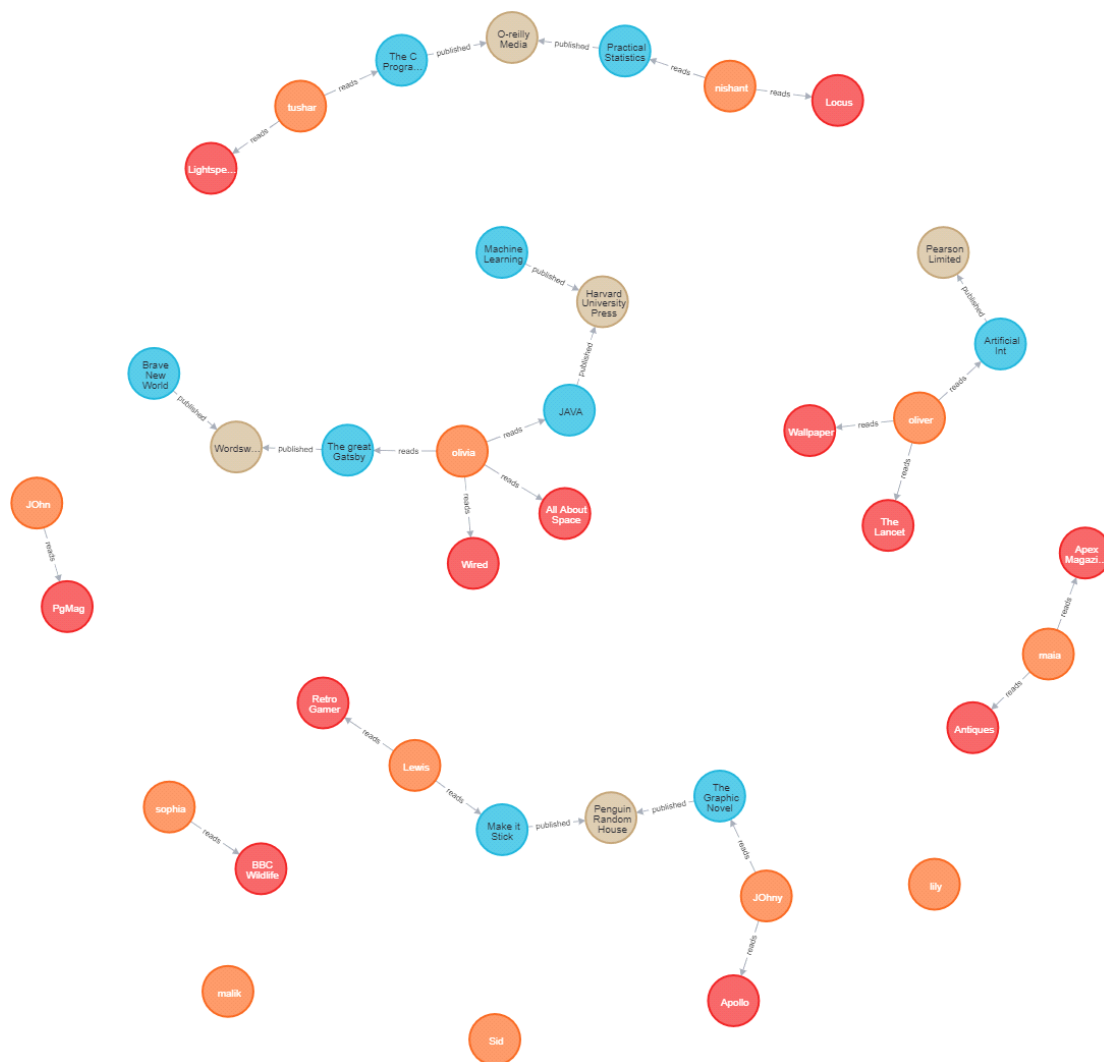
Match(m:magazine),(r:reader) where m.magazine_id='ms01' and r.reader_id='A116' CREATE (r)-[br:reads]->(m) return br

Below is the snapshot of the relationship



To See all the relations of all nodes:

match(abc) return abc



All the nodes are created and relations are made between them.

Explanation of the above database created:

In this database we created four nodes which are (Books,Publisher,Magazine,Reader) and in Library Management System, the most important entities are readers and the books which are present in the library,magazines and ebooks available in the library. The Publisher is an additional entity that contains information on the Publisher who publishes the books. If readers need to find books using the publisher's information, the Publisher data is required. To link the data of books with the publisher, the Publisher entity is supported by a weak Publish entity.

Now, the entities we chose for the Library Management System suit the characteristics since the library system consists of readers' information for keeping track of data for each book rented by the reader, or for reserving the books needed, or for the reader's details for the book rented.

Test Case:

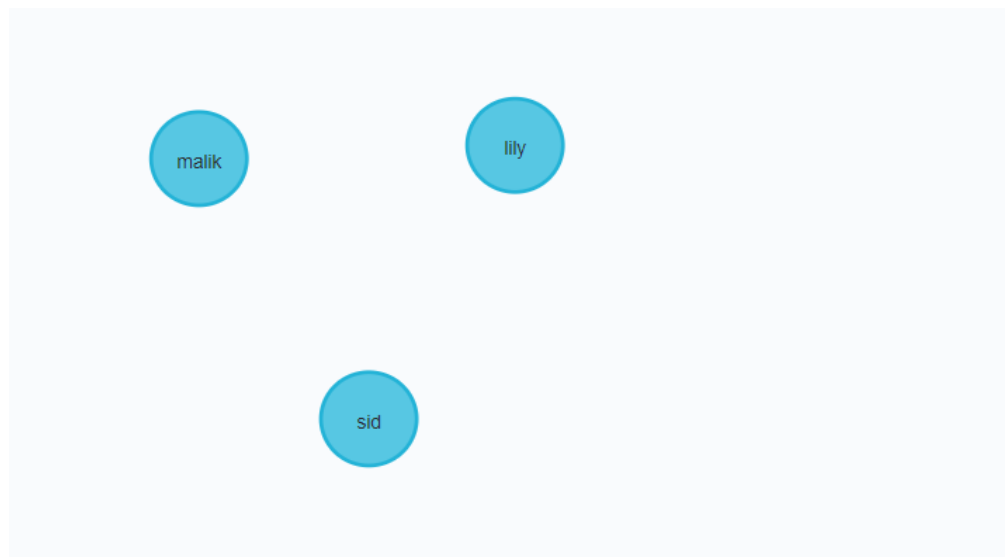
Q1- find all the readers who have to pay the fine

Answer-

```
MATCH (r:Reader)
```

```
WHERE NOT r.fine_id = "000"
```

```
RETURN r
```



Q2: reverse the relation between publisher and books

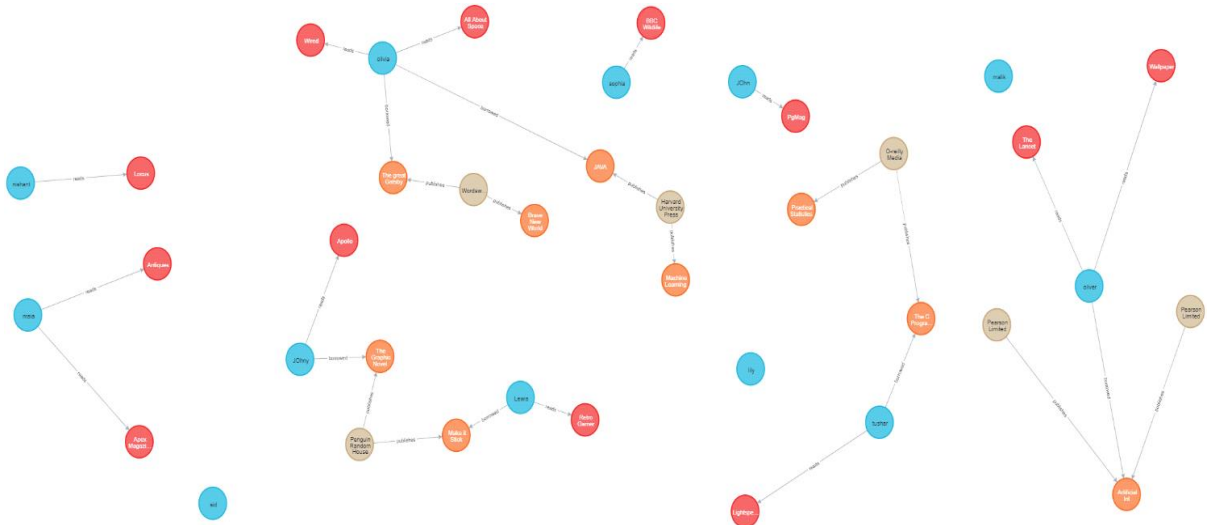
Answer-

```
match (b:books)-[rel:publishes]->(p:publisher)
```

```
CALL apoc.refactor.invert(rel)
```

```
yield input, output
```

```
RETURN input, output
```



Q3: find the science books which has more than 10 copies in library

Answer-

MATCH (b:books) WHERE b.No_of_copies>'10' and b.genre='science' RETURN b



Q4: show all the books and number of copies in descending order

Answer-

MATCH (b:books)

RETURN b.title, b.No_of_copies

ORDER BY b.title

"b.title"	"b.No_of_copies"
"The great Gatsby"	"1000"
"The Graphic Novel "	"45"
"The C Programming "	"19"
"Practical Statistics "	"100"
"Make it Stick "	"10"
"Machine Learning "	"19"
"JAVA "	"12"
"Brave New World"	"102"
"Artificial Int "	"19"

Q5 find all the magazine which starts with 'A'

Answer-

MATCH (M:magazine) WHERE M.title STARTS WITH 'A' RETURN M.title

M.title
"All About Space"
"Apex Magazine"
"Apollo"
"Antiques"

MONGO DB

Introduction:

MongoDB (Amit Phaltankar, Juned Ahsan, Michael Harrison, 2020) is a document-oriented database that operates on a number of platforms and provides superior efficiency, high availability, and high scalability. MongoDB is based on the collection and document principles. MongoDB is also a NoSQL database operator. NoSQL databases are taking the place of traditional relational databases. NoSQL databases make it easy to work with vast amounts of scattered data. MongoDB is a document-oriented database management system that can store and retrieve data.

The primary goal of MongoDB:

1. Scalability
2. Performance
3. Document Based
4. Indexing

Why Different from another Database:

We use MongoDB to manage unstructured data and massive documents, which we do by turning them to Jason files.

Creating MongoDB Database:

Mongo supports both GUI and CMD (MongoDB Team, n.d.). We used command prompt to create the collection and fields, and Mongo Compass to view the collection in GUI style.

Creation of Database;

Use Library

Creating Collection:

Collection: In theory, we could retain the same set of documents in MongoDB, but we can have a collection with a new set of documents. In a perfect world, we maintain the same set of documents in a single collection. In MongoDB, a collection is equivalent like a table in a relational SQL database, as well as a document is similar to a row in a relational SQL database. It is not required to get a collection in a database before inserting a document in MongoDB.

In this database we use “insert.Many” command to fill the multiple details on the collections

Creation Of Reader Collection:

a- Query

```
db.createCollection('Reader')
```

b- Insert Data In Reader Collection:

```
db.Reader.insertMany([  
  {  
    'reader_id' : 'A111',  
    'First_Name': 'John',  
    'Last_Name' : 'Hopes',  
    'Items_borrowed' : '2',  
    'Mobile_no' : '9898989898',  
    'Email' : 'john@gmail.com',  
    'Profession' : 'Student',  
    'Fine_Id' : '000'  
  },  
  {  
    'reader_id' : 'M112',  
    'First_Name': 'Lewis',  
    'Last_Name' : 'Smith',  
    'Items_borrowed' : '0',  
    'Mobile_no' : '9894489858',  
    'Email' : 'lewis@gmail.com',  
    'Profession' : 'm_Student',
```

```
'Fine_Id' : '000'
```

```
}}
```

Reader					
	_id ObjectId	reader_id String	First_Name String	Last_Name String	Items_borrowed Int32
1	ObjectId('6251d990c78b013d76c...')	"A111"	"John"	"Hopes"	2
2	ObjectId('6251d990c78b013d76c...')	"M112"	"Lewis"	"Smith"	0
3	ObjectId('6251d990c78b013d76c...')	"M113"	"Sid"	"Hopes"	3
4	ObjectId('6251d990c78b013d76c...')	"M114"	"Oliver"	"Brown"	4
5	ObjectId('6251d990c78b013d76c...')	"A115"	"Oliva"	"Devis"	4
6	ObjectId('6251d990c78b013d76c...')	"A116"	"Sophia"	"Jhonson"	1
7	ObjectId('6251d990c78b013d76c...')	"A117"	"Lily"	"Robert"	1
8	ObjectId('6251d990c78b013d76c...')	"T118"	"Johnny"	"Devis"	2
9	ObjectId('6251d990c78b013d76c...')	"T119"	"Maia"	"Smith"	0
10	ObjectId('6251d990c78b013d76c...')	"P112"	"Malik"	"Green"	2

Creation of Book Collection:

a- Query

```
db.createCollection('Book')
```

b- Insert Of Data In Collection

```
db.Book.insertMany([  
  {  
    'Book_Id' : 'bf01',  
    'Title' : 'The Great Gatsby',  
    'Author' : 'f.ScottFitzgerald',  
    'Publishing_date' : '10-JAN-18',  
    'No_Of_Copies' : 1000,  
    'Genere' : 'Tragedy',  
    'publisher_id' : 'P101',  
    'Reader_Id' : 'A115'  
  },  
  {  
    'Book_Id' : 'bf02',  
    'Title' : 'Brave New World',  
    'Author' : 'Aldous Husley',
```



```

'Publishing_date' : '10-Jan-32',
'No_Of_Copies'   : 102,
'Genre'          : 'Science',
'publisher_id'   : 'P102',
'Reader_Id'      : 'M112'
})

```

Book					
	_id ObjectId	Book_Id String	Title String	Author String	Publishing_date String
1	ObjectId('62640d1cda0f89d4b1b...	"bf01"	"The Great Gatsby"	"f.ScottFitzgerald"	"10-Jan-18"
2	ObjectId('62640d1cda0f89d4b1b...	"bf02"	"Brave New World"	"Aldous Husley"	"10-Jan-32"
3	ObjectId('62640d1cda0f89d4b1b...	"bf03"	"The Graphic Novel"	"Margaret Atwood"	"1-Feb-85"
4	ObjectId('62640d1cda0f89d4b1b...	"ed01"	"Make It Stick"	"Henry Roediger"	"14-Apr-14"
5	ObjectId('62640d1cda0f89d4b1b...	"ed02"	"Artificial Int"	"peter Norvig"	"13-Dec-94"
6	ObjectId('62640d1cda0f89d4b1b...	"ed03"	"Practical Statistics"	"Andrew Bruce"	"10-May-17"
7	ObjectId('62640d1cda0f89d4b1b...	"ed04"	"The C Programming"	"Brain Kernighan"	"10-Jan-78"
8	ObjectId('62640d1cda0f89d4b1b...	"ph02"	"Machine Learning"	"Alex"	"20-Jan-15"
9	ObjectId('62640d1cda0f89d4b1b...	"ph77"	"JAVA"	"F.Scott"	"10-Oct-19"

Creation of Ebook Collection:

a- Query

```
db.createCollection('Ebook')
```

b- Insert Of Data In Collection:

```

db.Ebooks.insertMany([
  {
    'Ebook_id' : 'EB001',
    'title' : 'The Keeper of Stories',
    'url' : 'https://www.kobo.com/gb/en/ebook/the-midnight-lock-2',
    'year' : '05-Jan-1996',
    'author' : 'Sally Page',
    'pages' : 343,
    'genre' : 'Adventure',
    'reader_id' : '001',
    'downloadreader_id' : 'A115'
  },

```

```
{
  'Ebook_id' : 'EB002',
  'title' : 'Piece of Her',
  'url' : 'https://www.kobo.com/gb/en/ebook/jailhouse-lawyer-1',
  'year' : '10-Jan-1987',
  'author' : 'Karin Slaughter',
  'pages' : 522,
  'genre' : 'Romantic',
  'reader_id' : '524',
  'downloadreader_id' : 'B552'
}
```

Ebooks					
	_id ObjectId	Ebook_id String	title String	url String	year String
1	ObjectId('62640f97da0f89d4b1b...')	"EB001"	"The Keeper of Stories"	"https://www.kobo.com/gb/en/e..."	"05-Jan-1996"
2	ObjectId('62640f97da0f89d4b1b...')	"EB002"	"Piece of Her"	"https://www.kobo.com/gb/en/e..."	"10-Jan-1987"
3	ObjectId('62640f97da0f89d4b1b...')	"EB003"	"The Midenight Lock"	"https://www.kobo.com/gb/en/e..."	"1-Feb-1985"
4	ObjectId('62640f97da0f89d4b1b...')	"EB004"	"Jailhouse Lawyer"	"https://www.kobo.com/gb/en/e..."	"14-Apr-2014"
5	ObjectId('62640f97da0f89d4b1b...')	"EB005"	"God Is Dead"	"https://www.kobo.com/gb/en/e..."	"13-Dec-1994"
6	ObjectId('62640f97da0f89d4b1b...')	"EB006"	"The Kaiju Preservation Socie..."	"https://www.kobo.com/gb/en/e..."	"10-May-2017"
7	ObjectId('62640f97da0f89d4b1b...')	"EB007"	"All The White Friends I Coul..."	"https://www.kobo.com/gb/en/e..."	"10-Jan-1978"
8	ObjectId('62640f97da0f89d4b1b...')	"EB008"	"Mission Total Resilience"	"https://www.kobo.com/gb/en/e..."	"20-Jan-2015"
9	ObjectId('62640f97da0f89d4b1b...')	"EB009"	"My Part in Hid Downfall"	"https://www.kobo.com/gb/en/e..."	"10-Oct-1919"

Creation of Publisher Collection:

a- Query

```
db.createCollection('Publisher')
```

b- Insert Of Data in Collection:

```
{
  "publisher_id": "P101",
```

```

    "Publisher_name": "Wordsworth-Edition Limited",
    "phone_no": "0120909230",
    "email_address": "Wordsworth-Edition.com",
    "country": "United Kingdom"
  },
  {
    "publisher_id": "P102",
    "Publisher_name": "Penguin Random House",
    "phone_no": "0120912345",
    "email_address": "enquiries@penguinrandomhouse.com",
    "country": "United States"
  },
  {
    "publisher_id": "P103",
    "Publisher_name": "Pearson Limited",
    "phone_no": "0120128470",
    "email_address": "enquiries@pearson.com",
    "country": "United Kingdom"
  }
})

```

Publisher					
	_id ObjectId	publisher_id String	Publisher_name String	phone_no String	email_address String
1	ObjectId('62537405e6f62497134...')	"P101"	"Wordsworth-Edition Limited"	"0120909230"	"Wordsworth-Edition.co
2	ObjectId('62537405e6f62497134...')	"P102"	"Penguin Random House"	"0120912345"	"enquiries@penguinrand
3	ObjectId('62537405e6f62497134...')	"P103"	"Pearson Limited"	"0120128470"	"enquiries@pearson.com
4	ObjectId('62537405e6f62497134...')	"P104"	"O-reilly Media"	"012030475"	"enquiries@oreilly.com
5	ObjectId('62537405e6f62497134...')	"P105"	"Harvard University Press"	"012012349"	"enquiries@harvardunip

Creation Of Magazine Collection:

a- Query

```
db.createCollection('Magazine')
```

b- Insert Of Data In Collection:

```
db.Magazine.insertMany([  
  {'magazine_id' : 'mg01',  
    'title' : 'Wired',  
    'date_of_issue' : '01-Jan-21',  
    'genre' : 'Technology',  
    'reader_id' : 'A115'  
  },  
  {  
    'magazine_id' : 'mg02',  
    'title' : 'PgMag',  
    'date_of_issue' : '05-Jul-19',  
    'genre' : 'Technology',  
    'reader_id' : 'A111'  
  },  
  {  
    'magazine_id' : 'mg03',  
    'title' : 'Retro Gamer',  
    'date_of_issue' : '28-Mar-21',  
    'genre' : 'Technology',  
    'reader_id' : 'M112'  
  }  
])
```

Magazine					
	_id ObjectId	magazine_id String	title String	date_of_issue String	genre String
1	ObjectId('626410c4da0f89d4b1b...')	"mg01"	"Wired"	"01-Jan-21"	"Technology"
2	ObjectId('626410c4da0f89d4b1b...')	"mg02"	"PgMag"	"05-Jul-19"	"Technology"
3	ObjectId('626410c4da0f89d4b1b...')	"mg03"	"Retro Gamer"	"28-Mar-21"	"Technology"
4	ObjectId('626410c4da0f89d4b1b...')	"ms01"	"BBC Wildlife"	"11-Jan-17"	"Science"
5	ObjectId('626410c4da0f89d4b1b...')	"ms02"	"The Lancet"	"22-Dec-19"	"Science"
6	ObjectId('626410c4da0f89d4b1b...')	"ms03"	"All About Space"	"09-Sep-18"	"Science"
7	ObjectId('626410c4da0f89d4b1b...')	"mf01"	"Apex Magazine"	"22-Sep-22"	"Fiction"
8	ObjectId('626410c4da0f89d4b1b...')	"mf02"	"Locus"	"28-Nov-17"	"Fiction"
9	ObjectId('626410c4da0f89d4b1b...')	"ma01"	"Apollo"	"29-Sep-13"	"Art"
10	ObjectId('626410c4da0f89d4b1b...')	"ma02"	"Wallpaper"	"05-Apr-15"	"Art"

Explanation of the above database created:

We used the Reader, Books, Ebooks, Magazine, Publisher, and Publish tables in this database. The most significant entities in a Mongo DB Library Management System are readers and the books that are offered in the library, which also include books, magazines, and ebooks. We don't need to define variables in Mongo DB like we do in other databases. The Publisher is an additional entity that contains information on the Publisher who publishes the books. If readers need to find books using the publisher's information, the Publisher data is required.

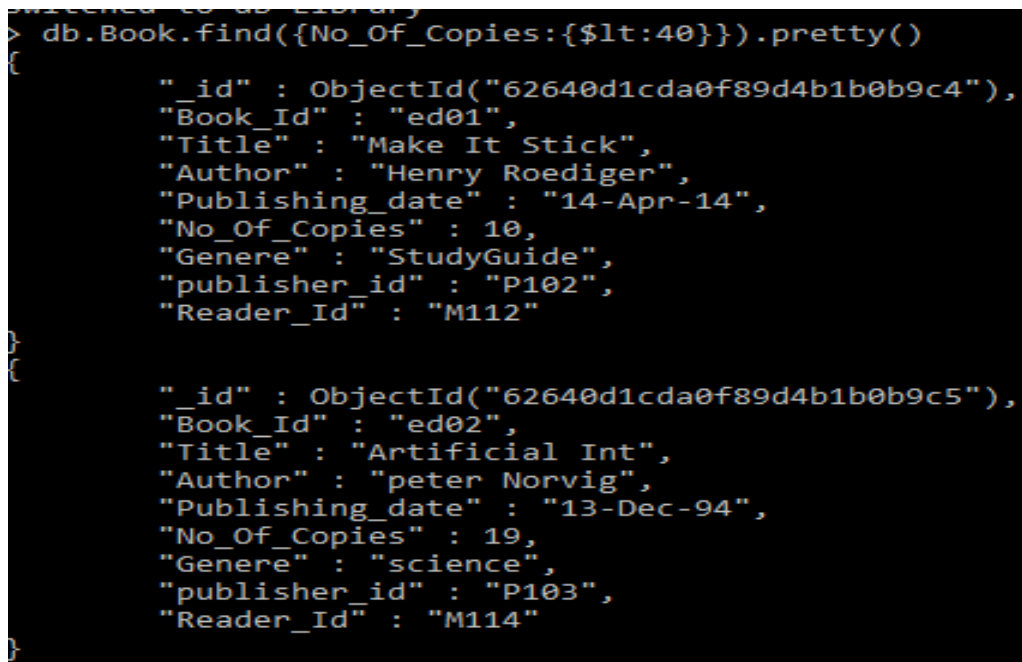
The entities we chose for the Library Management System now fulfil the requirements because the Library system is made up of reader information for keeping track of the books.

Test Cases:

Q1. Find the books which have less than 40 copies in the library

Query:

```
db.Book.find({No_Of_Copies:{<40}}).pretty()
```

A screenshot of a MongoDB terminal window with a black background and green text. The prompt is '>'. The command entered is 'db.Book.find({No_Of_Copies:{<40}}).pretty()'. The output shows two JSON objects. The first object has fields: '_id' (ObjectId), 'Book_Id' (ed01), 'Title' (Make It Stick), 'Author' (Henry Roediger), 'Publishing_date' (14-Apr-14), 'No_Of_Copies' (10), 'Genre' (StudyGuide), 'publisher_id' (P102), and 'Reader_Id' (M112). The second object has fields: '_id' (ObjectId), 'Book_Id' (ed02), 'Title' (Artificial Int), 'Author' (peter Norvig), 'Publishing_date' (13-Dec-94), 'No_Of_Copies' (19), 'Genre' (science), 'publisher_id' (P103), and 'Reader_Id' (M114).

```
> db.Book.find({No_Of_Copies:{<40}}).pretty()
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c4"),
  "Book_Id" : "ed01",
  "Title" : "Make It Stick",
  "Author" : "Henry Roediger",
  "Publishing_date" : "14-Apr-14",
  "No_Of_Copies" : 10,
  "Genre" : "StudyGuide",
  "publisher_id" : "P102",
  "Reader_Id" : "M112"
}
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c5"),
  "Book_Id" : "ed02",
  "Title" : "Artificial Int",
  "Author" : "peter Norvig",
  "Publishing_date" : "13-Dec-94",
  "No_Of_Copies" : 19,
  "Genre" : "science",
  "publisher_id" : "P103",
  "Reader_Id" : "M114"
}
```

Whole output:

```
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c4"),
  "Book_Id" : "ed01",
  "Title" : "Make It Stick",
  "Author" : "Henry Roediger",
  "Publishing_date" : "14-Apr-14",
  "No_Of_Copies" : 10,
  "Genre" : "StudyGuide",
  "publisher_id" : "P102",
```

```

    "Reader_Id" : "M112"
  }
  {
    "_id" : ObjectId("62640d1cda0f89d4b1b0b9c5"),
    "Book_Id" : "ed02",
    "Title" : "Artificial Int",
    "Author" : "peter Norvig",
    "Publishing_date" : "13-Dec-94",
    "No_Of_Copies" : 19,
    "Genre" : "science",
    "publisher_id" : "P103",
    "Reader_Id" : "M114"
  }
  {
    "_id" : ObjectId("62640d1cda0f89d4b1b0b9c7"),
    "Book_Id" : "ed04",
    "Title" : "The C Programming",
    "Author" : "Brain Kernighan",
    "Publishing_date" : "10-Jan-78",
    "No_Of_Copies" : 19,
    "Genre" : "Science",
    "publisher_id" : "P103",
    "Reader_Id" : "P111"
  }
  {
    "_id" : ObjectId("62640d1cda0f89d4b1b0b9c8"),
    "Book_Id" : "ph02",
    "Title" : "Machine Learning",

```

```

    "Author" : "Alex",
    "Publishing_date" : "20-Jan-15",
    "No_Of_Copies" : 19,
    "Genre" : "Science",
    "publisher_id" : "P104",
    "Reader_Id" : "P113"
  }
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c9"),
  "Book_Id" : "ph77",
  "Title" : "JAVA",
  "Author" : "F.Scott",
  "Publishing_date" : "10-Oct-19",
  "No_Of_Copies" : 12,
  "Genre" : "Coding",
  "publisher_id" : "P105",
  "Reader_Id" : "A115"
}

```

Q2. Total number of books borrowed from library by readers

Query:

```
db.Reader.aggregate({$count:'Items_borrowed'})
```

```

{
  "Items_borrowed" : 15
}

```

Q3. Get the readers who studying science genere

Query:


```
db.Book.aggregate( [{ $lookup: { from: 'Reader', localField: 'Reader_Id', foreignField: 'Reader_Id', as: 'genre' } }, { $match: { Genre: 'Science' } } ]).pretty()
```

```
> db.Book.aggregate( [{ $lookup: { from: 'Reader', localField: 'Reader_Id', foreignField: 'Reader_Id', as: 'genre' } }, { $match: { Genre: 'Science' } } ]).pretty()
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c2"),
  "Book_Id" : "bf02",
  "Title" : "Brave New World",
  "Author" : "Aldous Husley",
  "Publishing_date" : "10-Jan-32",
  "No_Of_Copies" : 102,
  "Genre" : "Science",
  "publisher_id" : "P102",
  "Reader_Id" : "M112",
  "genre" : [ ]
}
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c6"),
  "Book_Id" : "ed03",
  "Title" : "Practical Statistics",
  "Author" : "Andrew Bruce",
  "Publishing_date" : "10-May-17",
  "No_Of_Copies" : 100,
  "Genre" : "Science",
  "publisher_id" : "P104",
  "Reader_Id" : "P111",
  "genre" : [ ]
}
```

Whole output:

```
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c2"),
  "Book_Id" : "bf02",
  "Title" : "Brave New World",
  "Author" : "Aldous Husley",
  "Publishing_date" : "10-Jan-32",
  "No_Of_Copies" : 102,
  "Genre" : "Science",
  "publisher_id" : "P102",
```

```

    "Reader_Id" : "M112",
    "genre" : [ ]
}
{
    "_id" : ObjectId("62640d1cda0f89d4b1b0b9c6"),
    "Book_Id" : "ed03",
    "Title" : "Practical Statistics",
    "Author" : "Andrew Bruce",
    "Publishing_date" : "10-May-17",
    "No_Of_Copies" : 100,
    "Genre" : "Science",
    "publisher_id" : "P104",
    "Reader_Id" : "P111",
    "genre" : [ ]
}
{
    "_id" : ObjectId("62640d1cda0f89d4b1b0b9c7"),
    "Book_Id" : "ed04",
    "Title" : "The C Programming",
    "Author" : "Brain Kernighan",
    "Publishing_date" : "10-Jan-78",
    "No_Of_Copies" : 19,
    "Genre" : "Science",
    "publisher_id" : "P103",
    "Reader_Id" : "P111",
    "genre" : [ ]
}
{

```

```
"_id" : ObjectId("62640d1cda0f89d4b1b0b9c8"),  
"Book_Id" : "ph02",  
"Title" : "Machine Learning",  
"Author" : "Alex",  
"Publishing_date" : "20-Jan-15",  
"No_Of_Copies" : 19,  
"Genre" : "Science",  
"publisher_id" : "P104",  
"Reader_Id" : "P113",  
"genre" : [ ]  
}
```

Q4: Find the all the books which id starts with 'ed'.

Query:

```
db.Book.find({Book_Id: /^ed/}).pretty()
```

```
> db.Book.find({Book_Id: /^ed/}).pretty()
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c4"),
  "Book_Id" : "ed01",
  "Title" : "Make It Stick",
  "Author" : "Henry Roediger",
  "Publishing_date" : "14-Apr-14",
  "No_Of_Copies" : 10,
  "Genre" : "StudyGuide",
  "publisher_id" : "P102",
  "Reader_Id" : "M112"
}
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c5"),
  "Book_Id" : "ed02",
  "Title" : "Artificial Int",
  "Author" : "peter Norvig",
  "Publishing_date" : "13-Dec-94",
  "No_Of_Copies" : 19,
  "Genre" : "science",
  "publisher_id" : "P103",
  "Reader_Id" : "M114"
}
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c6"),
  "Book_Id" : "ed03",
  "Title" : "Practical Statistics",
  "Author" : "David J. Spiegelhalter",
  "Publishing_date" : "10-Nov-94",
  "No_Of_Copies" : 10,
  "Genre" : "science",
  "publisher_id" : "P103",
  "Reader_Id" : "M114"
}
```

Whole output:

```
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c4"),
  "Book_Id" : "ed01",
  "Title" : "Make It Stick",
  "Author" : "Henry Roediger",
  "Publishing_date" : "14-Apr-14",
  "No_Of_Copies" : 10,
  "Genre" : "StudyGuide",
  "publisher_id" : "P102",
  "Reader_Id" : "M112"
}
```

```

}
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c5"),
  "Book_Id" : "ed02",
  "Title" : "Artificial Int",
  "Author" : "peter Norvig",
  "Publishing_date" : "13-Dec-94",
  "No_Of_Copies" : 19,
  "Genre" : "science",
  "publisher_id" : "P103",
  "Reader_Id" : "M114"
}
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c6"),
  "Book_Id" : "ed03",
  "Title" : "Practical Statistics",
  "Author" : "Andrew Bruce",
  "Publishing_date" : "10-May-17",
  "No_Of_Copies" : 100,
  "Genre" : "Science",
  "publisher_id" : "P104",
  "Reader_Id" : "P111"
}
{
  "_id" : ObjectId("62640d1cda0f89d4b1b0b9c7"),
  "Book_Id" : "ed04",
  "Title" : "The C Programming",
  "Author" : "Brain Kernighan",

```

```

    "Publishing_date" : "10-Jan-78",
    "No_Of_Copies" : 19,
    "Genre" : "Science",
    "publisher_id" : "P103",
    "Reader_Id" : "P111"
}

```

Q5. Find the ‘Mystery and suspense’ eBooks with pages between 400 and 1500.

Query:

```
db.Ebooks.find({ $and: [{"genre": "Mystery And Suspence"}, {"pages": { $gte: 400, $lte: 1500}}]}).pretty()
```

```

> db.Ebooks.find({ $and: [{"genre": "Mystery And Suspence"}, {"pages": { $gte: 400, $lte: 1500}}]}).pretty()
{
  "_id" : ObjectId("62640f97da0f89d4b1b0b9d5"),
  "Ebook_id" : "EB003",
  "title" : "The Midenight Lock",
  "url" : "https://www.kobo.com/gb/en/ebook/the-midnight-lock-2",
  "year" : "1-Feb-1985",
  "author" : "Jeffery Deaver",
  "pages" : 454,
  "genre" : "Mystery And Suspence",
  "reader_id" : "819",
  "downloadreader_id" : "T118"
}
{
  "_id" : ObjectId("62640f97da0f89d4b1b0b9d6"),
  "Ebook_id" : "EB004",
  "title" : "Jailhouse Lawyer",
  "url" : "https://www.kobo.com/gb/en/ebook/jailhouse-lawyer-1",
  "year" : "14-Apr-2014",
  "author" : "James Patterson",
  "pages" : 1001,
  "genre" : "Mystery And Suspence",
  "reader_id" : "452",
  "downloadreader_id" : "M112"
}
>

```

Whole output:

```

{
  "_id" : ObjectId("62640f97da0f89d4b1b0b9d5"),

```

```

    "Ebook_id" : "EB003",
    "title" : "The Midenight Lock",
    "url" : "https://www.kobo.com/gb/en/ebook/the-midnight-lock-2",
    "year" : "1-Feb-1985",
    "author" : "Jeffery Deaver",
    "pages" : 454,
    "genre" : "Mystery And Suspence",
    "reader_id" : "819",
    "downloadreader_id" : "T118"
  }
  {
    "_id" : ObjectId("62640f97da0f89d4b1b0b9d6"),
    "Ebook_id" : "EB004",
    "title" : "Jailhouse Lawyer",
    "url" : "https://www.kobo.com/gb/en/ebook/jailhouse-lawyer-1",
    "year" : "14-Apr-2014",
    "author" : "James Patterson",
    "pages" : 1001,
    "genre" : "Mystery And Suspence",
    "reader_id" : "452",
    "downloadreader_id" : "M112"
  }

```