ORACLE
MONGODB
NEO4J

Anik Muhib

CST 4724

City College of Technology

Table of Contents

Part 1: Relational Databases – Use ORACLE 2

Normalizing the data and EER diagram 2

Create Table Statements 2

Insert Statement 4

Queries for Oracle 10

Part 2: Document Stores – Use MongoDB 14

Created collection studentinfo and inserted data 14

Showing the collecting have been created 14

MongoDB Queries 15

Part 3: Graph Store – Use Neo4j 16

Graphically represent the relationship between author and book 16

Create statement for author and book Nodes 16

create relationship statement between author and book 18

Graphically represent the relationship between publisher and book 19

Create statement for publisher Nodes 19

create relationships statement between publisher and book 20

Graphically represent the relationship between publisher and author 21

create relationships statement between publisher and author 21

Show all the nodes and relationships 22

Queries for Neo4j 23

Part 4: Experience Recap 24

Question 1 24

Question 2 24

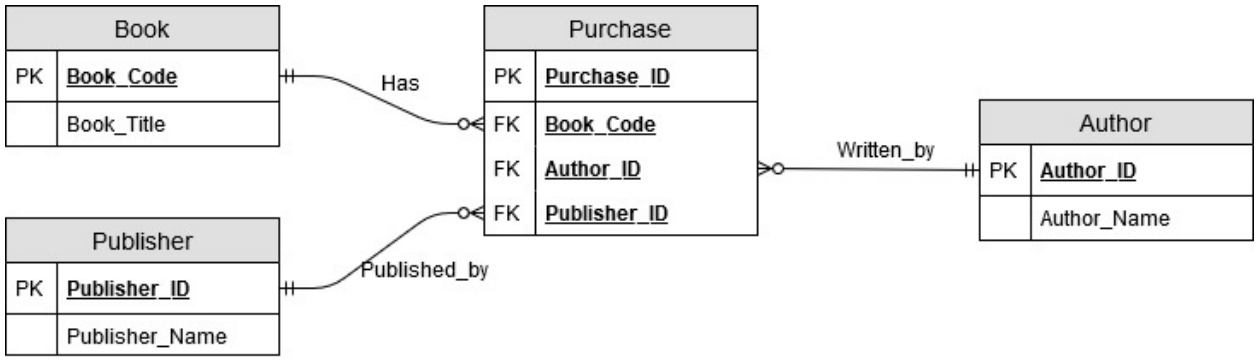
Question 3 25

Question 4 25

Question 5 26

Part 1: Relational Databases – Use ORACLE

Normalizing the data and EER diagram



Create Table Statements

Create Book Table:

```
1  Create table Book
2  (
3  Book_Code varchar2(2),
4  Book_Title varchar2(20),
5  Constraint Book_PK PRIMARY KEY (Book_Code)
6  );
```

```
Table created.
```

Create Publisher Table

```
Create table Publisher
(
Publisher_ID varchar2(10),
Publisher_Name varchar2(20),
Constraint Publisher_PK PRIMARY KEY (Publisher_ID)
);
```

```
Table created.
```

Create Author Table

```
Create table Author
(
Author_ID varchar2(10),
Author_Name varchar2(20),
Constraint Author_PK PRIMARY KEY (Author_ID)
);
```

```
Table created.
```

Create Purchase Table

```
Create table Purchase
(
Purchase_ID varchar2(10),
Book_Code varchar2(2),
Author_ID varchar2(10),
Publisher_ID varchar2(10),

Constraint Purchase_PK PRIMARY KEY(Purchase_ID),
Constraint Book_FK FOREIGN KEY(Book_Code) References Book(Book_Code),
Constraint Author_FK FOREIGN KEY(Author_ID) References Author(Author_ID),
Constraint Publisher_FK FOREIGN KEY(Publisher_ID) References Publisher(Publisher_ID)
);
```

```
Table created.
```

Insert Statement

Insert into Book table:

```
Insert into Book
Values (22,'Stranger')
```

1 row(s) inserted.

```
Insert into Book
Values (13,'Dreamcatcher')
```

1 row(s) inserted.

```
Insert into Book
Values (18,'Beloved')
```

1 row(s) inserted.

```
Insert into Book
Values (37,'Nine')
```

1 row(s) inserted.

```
Insert into Book
Values (57,'Catch 22')
```

1 row(s) inserted.

```
Insert into Book
Values (61,'Jazz')
```

1 row(s) inserted.

```
Insert into Book
Values (69,'Franny')
```

1 row(s) inserted.

```
Insert into Book
Values (75,'Fall')
```

1 row(s) inserted.

```
Insert into Book
Values (96,'Grapes')
```

1 row(s) inserted.

```
Insert into Book
Values (98,'Catcher')
```

1 row(s) inserted.

Book table with inserted data:

```
66  Select *
67  From Book;
68
```

BOOK_CODE	BOOK_TITLE
61	Jazz
96	Grapes
22	Stranger
69	Franny
98	Catcher
13	Dreamcatcher
57	Catch 22
18	Beloved
37	Nine
75	Fall

Download CSV

10 rows selected.

Insert into Publisher table

```
Insert into Publisher
Values (111,'Vintage')
```

1 row(s) inserted.

```
Insert into Publisher
Values (112,'Scribner')
```

1 row(s) inserted.

```
Insert into Publisher
Values (113,'Plume')
```

1 row(s) inserted.

```
Insert into Publisher
Values (114,'LB Books')
```

1 row(s) inserted.

```
Insert into Publisher
Values (115,'Penguin')
```

1 row(s) inserted.

Publisher table with inserted data:

```
80  Select *
81  From Publisher;
82
```

PUBLISHER_ID	PUBLISHER_NAME
113	Plume
111	Vintage
112	Scribner
114	LB Books
115	Penguin

Download CSV

5 rows selected.

Insert into Author table

```
Insert into Author
Values (222,'Camus')
```

1 row(s) inserted.

```
Insert into Author
Values (223,'King')
```

1 row(s) inserted.

```
Insert into Author
Values (224,'Morrison')
```

1 row(s) inserted.

```
Insert into Author
Values (225,'Salinger')
```

1 row(s) inserted.

```
Insert into Author
Values (226,'Heller')
```

1 row(s) inserted.

Author table with inserted data:

```
98  Select *
99  From Author;
```

AUTHOR_ID	AUTHOR_NAME
223	King
222	Camus
224	Morrison
225	Salinger
226	Heller

Download CSV

5 rows selected.

Insert data into Purchase table

```
Insert into Purchase
Values (333,22,222,111)
```

1 row(s) inserted.

```
Insert into Purchase
Values (334,13,223,112)
```

1 row(s) inserted.

```
Insert into Purchase
Values (335,18,224,113)
```

1 row(s) inserted.

```
Insert into Purchase
Values (336,37,225,114)
```

1 row(s) inserted.

```
Insert into Purchase
Values (337,57,226,112)
```

1 row(s) inserted.

```
Insert into Purchase
Values (338,61,224,113)
```

1 row(s) inserted.

```
Insert into Purchase
Values (339,69,225,114)
```

1 row(s) inserted.

```
Insert into Purchase
Values (340,75,222,111)
```

1 row(s) inserted.

```
Insert into Purchase
Values (341,96,224,115)
```

1 row(s) inserted.

```
Insert into Purchase
Values (342,98,225,114)
```

1 row(s) inserted.

Purchase table with inserted data:

```
32  Select *
33  From Purchase;
```

PURCHASE_ID	BOOK_CODE	AUTHOR_ID	PUBLISHER_ID
334	13	223	112
333	22	222	111
336	37	225	114
338	61	224	113
339	69	225	114
335	18	224	113
341	96	224	115
340	75	222	111
342	98	225	114
337	57	226	112

Download CSV

10 rows selected.

Queries for Oracle

5.a) For each author name, show the book title. Your output should include author name and book title. Order the output by author name in descending order

```
156 Select Author_Name, Book_Title
157 From Author
158 INNER JOIN Purchase
159 ON Author.Author_ID = Purchase.Author_ID
160 INNER JOIN Book
161 ON Purchase.Book_Code = Book.Book_Code
162 Order by Author_Name DESC;
```

AUTHOR_NAME	BOOK_TITLE
Salinger	Nine
Salinger	Franny
Salinger	Catcher
Morrison	Grapes
Morrison	Beloved
Morrison	Jazz
King	Dreamcatcher
Heller	Catch 22
Camus	Stranger
Camus	Fall

5.b) For each publisher, show the books they have published. Your output should include publisher name and book title. Order the output by publisher name in ascending order.

```
165 Select Publisher_Name, Book_Title
166 From Publisher
167 INNER JOIN Purchase
168 ON Publisher.Publisher_ID = Purchase.Publisher_ID
169 INNER JOIN Book
170 ON Purchase.Book_Code = Book.Book_Code
171 Order by Publisher_Name ASC;
```

PUBLISHER_NAME	BOOK_TITLE
LB Books	Nine
LB Books	Franny
LB Books	Catcher
Penguin	Grapes
Plume	Jazz
Plume	Beloved
Scribner	Dreamcatcher
Scribner	Catch 22
Vintage	Fall
Vintage	Stranger

5.c) For each author, show the publisher who has published his work. Your output should include the author's name and the publisher's name. Order the output by author's name in ascending order.

```
174  Select Author_Name, Publisher_Name
175  From Author
176  INNER JOIN Purchase
177  ON Author.Author_ID = Purchase.Author_ID
178  INNER JOIN Publisher
179  ON Purchase.Publisher_ID = Publisher.Publisher_ID
180  Order By Author Name ASC;
```

AUTHOR_NAME	PUBLISHER_NAME
Camus	Vintage
Camus	Vintage
Heller	Scribner
King	Scribner
Morrison	Plume
Morrison	Plume
Morrison	Penguin
Salinger	LB Books
Salinger	LB Books
Salinger	LB Books

5.d) List the title of all the books which Salinger has written.

```
183  Select Author_Name, Book_Title
184  From Author
185  INNER JOIN Purchase
186  ON Author.Author_ID = Purchase.Author_ID
187  INNER JOIN Book
188  ON Purchase.Book_Code = Book.Book_Code
189  Where Author_Name = 'Salinger';
```

AUTHOR_NAME	BOOK_TITLE
Salinger	Catcher
Salinger	Franny
Salinger	Nine

5.e) List the title of all the books published by publisher Vintage

```
192  Select Publisher_Name, Book_Title
193  From Publisher
194  INNER JOIN Purchase
195  ON Publisher.Publisher_ID = Purchase.Publisher_ID
196  INNER JOIN Book
197  ON Purchase.Book_Code = Book.Book_Code
198  Where Publisher_Name = 'Vintage';
```

PUBLISHER_NAME	BOOK_TITLE
Vintage	Stranger
Vintage	Fall

5.f) List the title of all the books and the publisher's name for all books published by Vintage, LB Books or Plume (use IN)

```
200  Select Publisher_Name, Book_Title
201  From Publisher
202  INNER JOIN Purchase
203  ON Publisher.Publisher_ID = Purchase.Publisher_ID
204  INNER JOIN Book
205  ON Purchase.Book_Code = Book.Book_Code
206  Where Publisher_Name IN ('Vintage','LB Books','Plume');
```

PUBLISHER_NAME	BOOK_TITLE
Vintage	Stranger
LB Books	Catcher
Plume	Beloved
LB Books	Franny
Vintage	Fall
LB Books	Nine
Plume	Jazz

5.g) List the title of the books and the publisher name for all books published by Scribner or Plume (use OR) ***NOTE: you should only have four lines of output**

```
208  Select Publisher_Name, Book_Title
209  From Publisher
210  INNER JOIN Purchase
211  ON Publisher.Publisher_ID = Purchase.Publisher_ID
212  INNER JOIN Book
213  ON Purchase.Book_Code = Book.Book_Code
214  Where Publisher_Name = 'Scribner' OR  Publisher_Name = 'Plume';
```

PUBLISHER_NAME	BOOK_TITLE
Scribner	Dreamcatcher
Scribner	Catch 22
Plume	Beloved
Plume	Jazz

5.h) List the title of the book and the publisher name for all books published by Penguin and written by Morrison (use AND)

```
217  Select Publisher_Name, Book_Title
218  From Publisher
219  INNER JOIN Purchase
220  ON Publisher.Publisher_ID = Purchase.Publisher_ID
221  INNER JOIN Book
222  ON Purchase.Book_Code = Book.Book_Code
223  INNER JOIN Author
224  ON Purchase.Author_ID = Author.Author_ID
225  Where Publisher_Name = 'Penguin' AND Author_Name = 'Morrison';
```

PUBLISHER_NAME	BOOK_TITLE
Penguin	Grapes

Part 2: Document Stores – Use MongoDB

1. Use MongoDB to enter the information displayed in table 1 into the collection (s) you have created

CREATED COLLECTION STUDENTINFO AND INSERTED DATA

```
MongoDB Web Shell

>>> use studentinfo
switched to db studentinfo
>>> db.bookinfo.insert({book_code:22,book_title:"stranger", publisher:"vintage", author:"camus"})
WriteResult({ "nInserted" : 1 })
>>> db.bookinfo.insert({book_code:13,book_title:"dreamcatcher", publisher:"scribner", author:"king"})
WriteResult({ "nInserted" : 1 })
>>> db.bookinfo.insert({book_code:18,book_title:"beloved", publisher:"plume", author:"morrison"})
WriteResult({ "nInserted" : 1 })
>>> db.bookinfo.insert({book_code:37,book_title:"nine", publisher:"lb books", author:"salinger"})
WriteResult({ "nInserted" : 1 })
>>> db.bookinfo.insert({book_code:57,book_title:"catch 22", publisher:"scribner", author:"heller"})
WriteResult({ "nInserted" : 1 })
>>> db.bookinfo.insert({book_code:61,book_title:"jazz", publisher:"plume", author:"morrison"})
WriteResult({ "nInserted" : 1 })
>>> db.bookinfo.insert({book_code:69,book_title:"franny", publisher:"lb books", author:"salinger"})
WriteResult({ "nInserted" : 1 })
>>> db.bookinfo.insert({book_code:75,book_title:"fall", publisher:"vintage", author:"camus"})
WriteResult({ "nInserted" : 1 })
>>> db.bookinfo.insert({book_code:96,book_title:"grapes", publisher:"penguin", author:"morrison"})
WriteResult({ "nInserted" : 1 })
>>> db.bookinfo.insert({book_code:98,book_title:"catcher", publisher:"lb books", author:"salinger"})
WriteResult({ "nInserted" : 1 })
>>>
```

2. Use MongoDB to show that the information has been entered into the collection(s) Here is all the data in collection studentinfo:

SHOWING THE COLLECTING HAVE BEEN CREATED

```
>>> db.bookinfo.find()
{ "_id" : ObjectId("5eae4ac690f2d74e1b229eed"), "book_code" : 22, "book_title" : "stranger", "publisher" : "vintage", "author" : "camus" }
{ "_id" : ObjectId("5eae4b2490f2d74e1b229eee"), "book_code" : 13, "book_title" : "dreamcatcher", "publisher" : "scribner", "author" : "king" }
{ "_id" : ObjectId("5eae4b5590f2d74e1b229eef"), "book_code" : 18, "book_title" : "beloved", "publisher" : "plume", "author" : "morrison" }
{ "_id" : ObjectId("5eae4b8490f2d74e1b229ef0"), "book_code" : 37, "book_title" : "nine", "publisher" : "lb books", "author" : "salinger" }
{ "_id" : ObjectId("5eae4c2290f2d74e1b229ef1"), "book_code" : 57, "book_title" : "catch 22", "publisher" : "scribner", "author" : "heller" }
{ "_id" : ObjectId("5eae4c5790f2d74e1b229ef2"), "book_code" : 61, "book_title" : "jazz", "publisher" : "plume", "author" : "morrison" }
{ "_id" : ObjectId("5eae4c8d90f2d74e1b229ef3"), "book_code" : 69, "book_title" : "franny", "publisher" : "lb books", "author" : "salinger" }
{ "_id" : ObjectId("5eae4cc490f2d74e1b229ef4"), "book_code" : 75, "book_title" : "fall", "publisher" : "vintage", "author" : "camus" }
{ "_id" : ObjectId("5eae4cf690f2d74e1b229ef5"), "book_code" : 96, "book_title" : "grapes", "publisher" : "penguin", "author" : "morrison" }
{ "_id" : ObjectId("5eae4d4390f2d74e1b229ef6"), "book_code" : 98, "book_title" : "catcher", "publisher" : "lb books", "author" : "salinger" }
>>>
```

MongoDB Queries

3a) Show all books written by author Salinger

```
>>> db.bookinfo.find({author:"salinger"})
{ "_id" : ObjectId("5eae4b8490f2d74e1b229ef0"), "book_code" : 37, "book_title" : "nine", "publisher" : "lb books", "author" : "salinger" }
{ "_id" : ObjectId("5eae4c8d90f2d74e1b229ef3"), "book_code" : 69, "book_title" : "franny", "publisher" : "lb books", "author" : "salinger" }
{ "_id" : ObjectId("5eae4d4390f2d74e1b229ef6"), "book_code" : 98, "book_title" : "catcher", "publisher" : "lb books", "author" : "salinger" }
```

3b) Show all books published by Vintage

```
>>> db.bookinfo.find({publisher:"vintage"})
{ "_id" : ObjectId("5eae4ac690f2d74e1b229eed"), "book_code" : 22, "book_title" : "stranger", "publisher" : "vintage", "author" : "camus" }
{ "_id" : ObjectId("5eae4cc490f2d74e1b229ef4"), "book_code" : 75, "book_title" : "fall", "publisher" : "vintage", "author" : "camus" }
```

3c) Show all books that are published by Vintage, LB Books or Plume (use IN)

```
>>> db.bookinfo.find({publisher:{ $in: ["vintage","lb books","plume"]}})
{ "_id" : ObjectId("5eae4ac690f2d74e1b229eed"), "book_code" : 22, "book_title" : "stranger", "publisher" : "vintage", "author" : "camus" }
{ "_id" : ObjectId("5eae4b5590f2d74e1b229eef"), "book_code" : 18, "book_title" : "beloved", "publisher" : "plume", "author" : "morrison" }
{ "_id" : ObjectId("5eae4b8490f2d74e1b229ef0"), "book_code" : 37, "book_title" : "nine", "publisher" : "lb books", "author" : "salinger" }
{ "_id" : ObjectId("5eae4c5790f2d74e1b229ef2"), "book_code" : 61, "book_title" : "jazz", "publisher" : "plume", "author" : "morrison" }
{ "_id" : ObjectId("5eae4c8d90f2d74e1b229ef3"), "book_code" : 69, "book_title" : "franny", "publisher" : "lb books", "author" : "salinger" }
{ "_id" : ObjectId("5eae4cc490f2d74e1b229ef4"), "book_code" : 75, "book_title" : "fall", "publisher" : "vintage", "author" : "camus" }
{ "_id" : ObjectId("5eae4d4390f2d74e1b229ef6"), "book_code" : 98, "book_title" : "catcher", "publisher" : "lb books", "author" : "salinger" }
```

3d) Show all books published by Scribner or Plume (use OR)

```
>>> db.bookinfo.find({ $or: [ {publisher:"scribner"},{publisher:"plume"}]})
{ "_id" : ObjectId("5eae4b2490f2d74e1b229eee"), "book_code" : 13, "book_title" : "dreamcatcher", "publisher" : "scribner", "author" : "king" }
{ "_id" : ObjectId("5eae4b5590f2d74e1b229eef"), "book_code" : 18, "book_title" : "beloved", "publisher" : "plume", "author" : "morrison" }
{ "_id" : ObjectId("5eae4c2290f2d74e1b229ef1"), "book_code" : 57, "book_title" : "catch 22", "publisher" : "scribner", "author" : "heller" }
{ "_id" : ObjectId("5eae4c5790f2d74e1b229ef2"), "book_code" : 61, "book_title" : "jazz", "publisher" : "plume", "author" : "morrison" }
>>>
```

3e) Show all books published by Penguin and written by Morrison (use AND)

```
>>> db.bookinfo.find({publisher:"penguin",author:"morrison"})
{ "_id" : ObjectId("5eae4cf690f2d74e1b229ef5"), "book_code" : 96, "book_title" : "grapes", "publisher" : "penguin", "author" : "morrison" }
>>>
```

Part 3: Graph Store – Use Neo4j

Graphically represent the relationship between author and book

Author	Book_Title	Book_Code
Camus	Stranger Fall	22 75
King	Dreamcatcher	13
Morrison	Beloved Jazz Grapes	18 61 96
Salinger	Nine Catcher Franny	37 98 69
Heller	Catch 22	57

Create statement for author and book Nodes

Create node for author

create (camus:author{name:'camus'})return camus

create (king:author{name:'king'})return king

create (morrison:author{name:'morrison'})return morrison

create (salinger:author{name:'salinger'})return salingercreate (heller:author{name:'heller'})return heller

Creating Node for Books:

create (stranger:book{name:'stranger',book_code:22})return stranger

create (fall:book{name:'fall',book_code:75})return fall

create (dream:book{name:'dreamchr',book_code:13})return dream

create (beloved:book{name:'beloved',book_code:18})return beloved

create (jazz:book{name:'jazz',book_code:61})return jazz

create (grapes:book{name:'grapes',book_code:96})return grapes

create (nine:book{name:'nine',book_code:37})return nine

create (catcher:book{name:'catcher',book_code:98})return catcher

create (franny:book{name:'franny',book_code:69})return franny

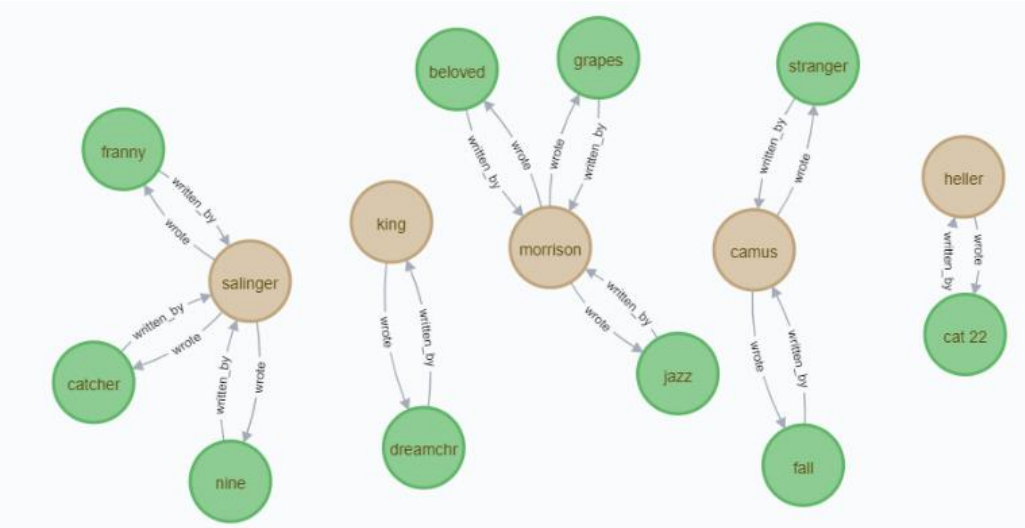
create (catch:book{name:'cat 22',book_code:57})return catch

CREATE RELATIONSHIP STATEMENT BETWEEN AUTHOR AND BOOK

Creating relationship between author and books:

match (a:author), (b:book) where a.name = 'camus' and b.name = 'stranger' create (a)-[r:wrote]->(b) -[:written_by]-> (a)
match (a:author), (b:book) where a.name = 'camus' and b.name = 'fall' create (a)-[r:wrote]->(b) -[:written_by]-> (a)
match (a:author), (b:book) where a.name = 'king' and b.name = 'dreamchr' create (a)-[r:wrote]->(b) -[:written_by]-> (a)
match (a:author), (b:book) where a.name = 'morrison' and b.name = 'beloved' create (a)-[r:wrote]->(b) -[:written_by]-> (a)
match (a:author), (b:book) where a.name = 'morrison' and b.name = 'jazz' create (a)-[r:wrote]->(b) -[:written_by]-> (a)
match (a:author), (b:book) where a.name = 'morrison' and b.name = 'grapes' create (a)-[r:wrote]->(b) -[:written_by]-> (a)
match (a:author), (b:book) where a.name = 'salinger' and b.name = 'nine' create (a)-[r:wrote]->(b) -[:written_by]-> (a)
match (a:author), (b:book) where a.name = 'salinger' and b.name = 'catcher' create (a)-[r:wrote]->(b) -[:written_by]-> (a)
match (a:author), (b:book) where a.name = 'salinger' and b.name = 'franny' create (a)-[r:wrote]->(b) -[:written_by]-> (a)
match (a:author), (b:book) where a.name = 'heller' and b.name = 'cat 22' create (a)-[r:wrote]->(b) -[:written_by]-> (a)

Show all the nodes and relationships



Graphically represent the relationship between publisher and book

Publisher	Book_Title
Vintage	Stranger Fall
Scribner	Dreamcatcher Catch 22
Plume	Beloved Jazz
LB Books	Nine Franny Catcher
Penguin	Grapes

Create statement for publisher Nodes

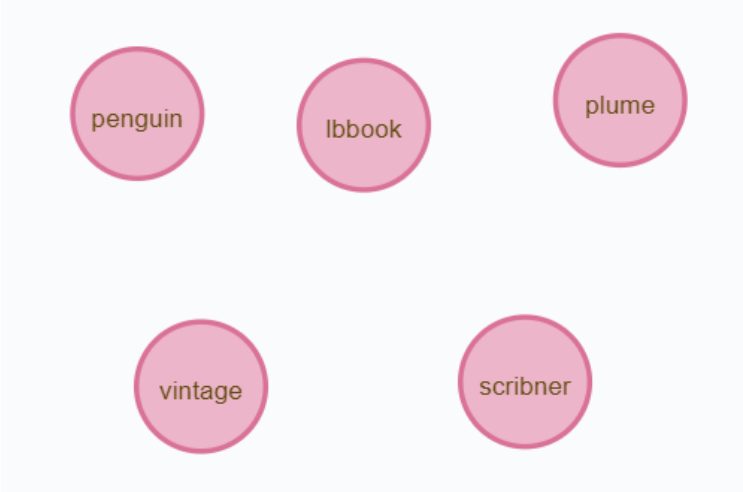
create (vintage:publisher{name:'vintage'})return vintage

create (scribner:publisher{name:'scribner'})return scribner

create (plume:publisher{name:'plume'})return plume

create (lbbook:publisher{name:'lbbook'})return lbbook

create (penguin:publisher{name:'penguin'})return penguin

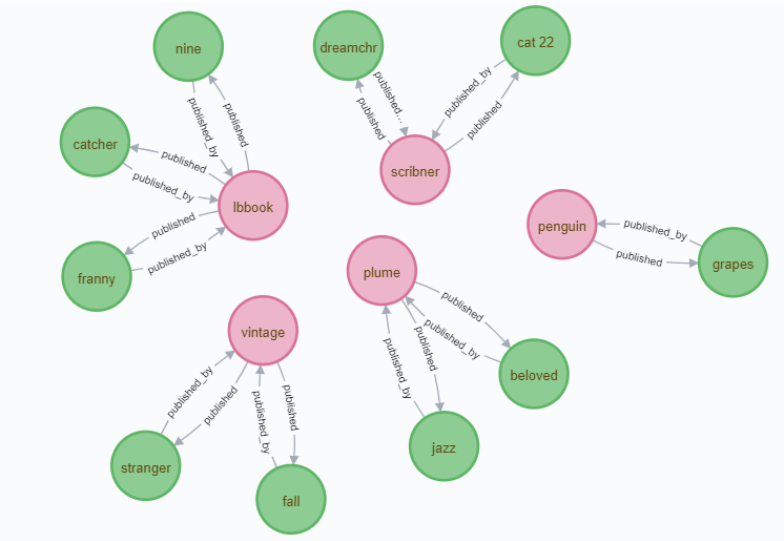


CREATE RELATIONSHIPS STATEMENT BETWEEN PUBLISHER AND BOOK

Relationship between publisher and book

match (a:publisher), (b:book) where a.name = 'vintage' and b.name = 'stranger' create (a)-[r:published]->(b) -[:published_by]-> (a)
match (a:publisher), (b:book) where a.name = 'vintage' and b.name = 'fall' create (a)-[r:published]->(b) -[:published_by]-> (a)
match (a:publisher), (b:book) where a.name = 'scribner' and b.name = 'dreamchr' create (a)-[r:published]->(b) -[:published_by]-> (a)
match (a:publisher), (b:book) where a.name = 'scribner' and b.name = 'cat 22' create (a)-[r:published]->(b) -[:published_by]-> (a)
match (a:publisher), (b:book) where a.name = 'plume' and b.name = 'beloved' create (a)-[r:published]->(b) -[:published_by]-> (a)
match (a:publisher), (b:book) where a.name = 'plume' and b.name = 'jazz' create (a)-[r:published]->(b) -[:published_by]-> (a)
match (a:publisher), (b:book) where a.name = 'lbbook' and b.name = 'nine' create (a)-[r:published]->(b) -[:published_by]-> (a)
match (a:publisher), (b:book) where a.name = 'lbbook' and b.name = 'franny' create (a)-[r:published]->(b) -[:published_by]-> (a)
match (a:publisher), (b:book) where a.name = 'lbbook' and b.name = 'catcher' create (a)-[r:published]->(b) -[:published_by]-> (a)
match (a:publisher), (b:book) where a.name = 'penguin' and b.name = 'grapes' create (a)-[r:published]->(b) -[:published_by]-> (a)

Show all the nodes and relationships



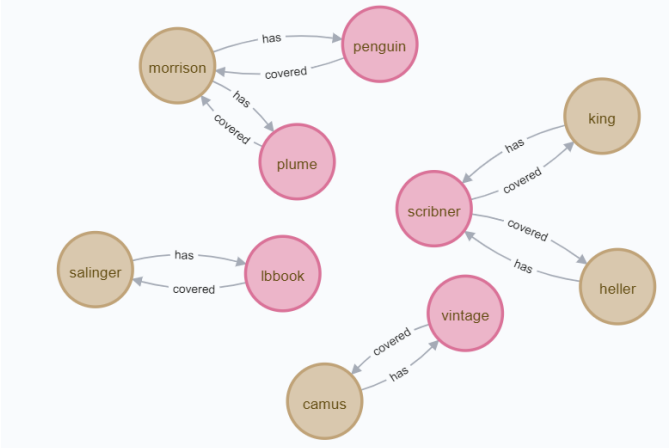
GRAPHICALLY REPRESENT THE RELATIONSHIP BETWEEN PUBLISHER AND AUTHOR

Publisher	Author
Vintage	Camus Camus
Scribner	King Heller
Plume	Morrison Morrison
LB Books	Salinger Salinger Salinger
Penguin	Morrison

CREATE RELATIONSHIPS STATEMENT BETWEEN PUBLISHER AND AUTHOR

Relationship between publisher and author

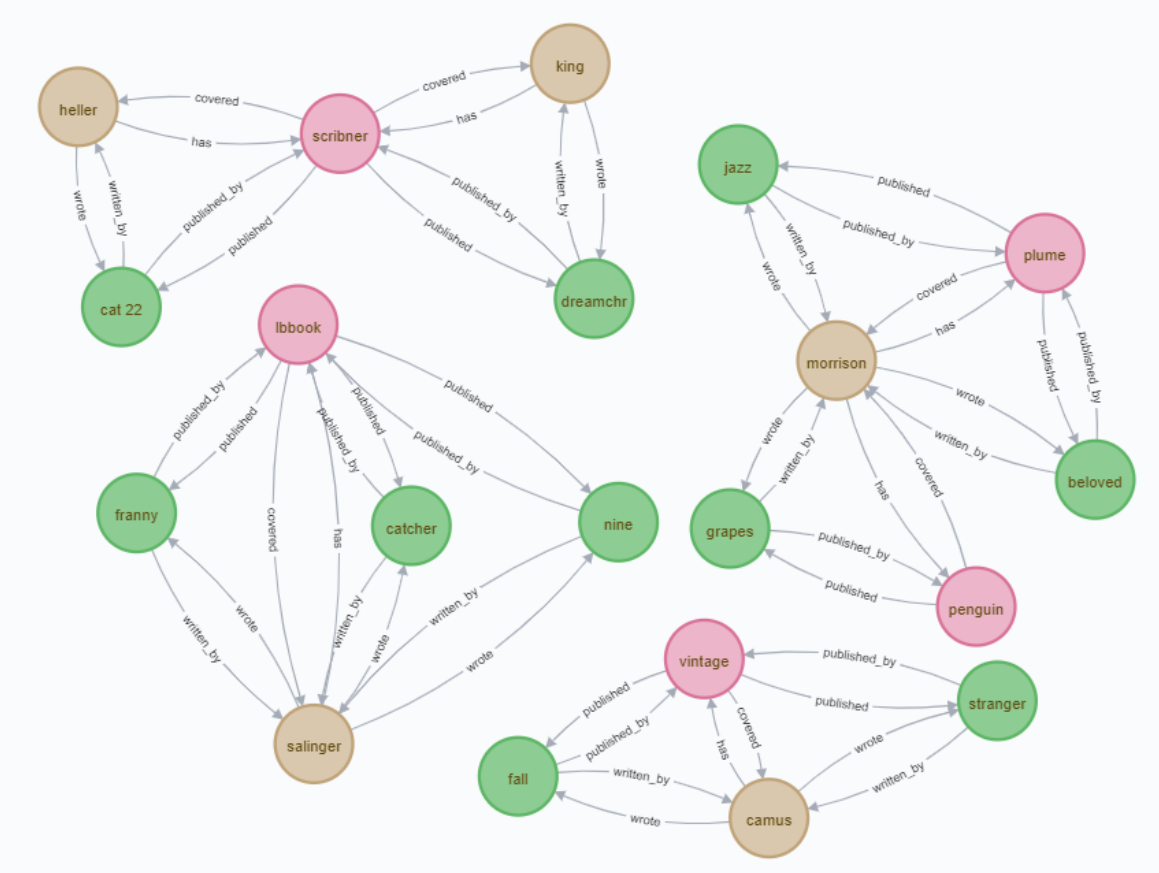
match (a:publisher), (b:author) where a.name = 'vintage' and b.name = 'camus' create (a)-[r:covered]->(b) -[:has]-> (a)
match (a:publisher), (b:author) where a.name = 'scribner' and b.name = 'king' create (a)-[r:covered]->(b) -[:has]-> (a)
match (a:publisher), (b:author) where a.name = 'scribner' and b.name = 'heller' create (a)-[r:covered]->(b) -[:has]-> (a)
match (a:publisher), (b:author) where a.name = 'plume' and b.name = 'morrison' create (a)-[r:covered]->(b) -[:has]-> (a)
match (a:publisher), (b:author) where a.name = 'lbbook' and b.name = 'salinger' create (a)-[r:covered]->(b) -[:has]-> (a)
match (a:publisher), (b:author) where a.name = 'penguin' and b.name = 'morrison' create (a)-[r:covered]->(b) -[:has]-> (a)



Show all the nodes and relationships

All nodes and relationships among them:

MATCH (n) RETURN n LIMIT 25



Queries for Neo4j

Sort the names in descending order

match(n) return n.name order by n.name DESC

n.name
"vintage"
"stranger"
"scribner"
"salinger"
"plume"
"penguin"
"nine"
"morrison"
"lbbook"
"king"
"jazz"
"heller"
"grapes"
"franny"
"fall"
"dreamchr"
"catcher"
"cat 22"
"camus"

Which publisher employs the author Morrison

match (author {name:'morrison' }) <- [:has|:covered] - (publisher) return publisher.name, author.name

publisher.name	author.name
"penguin"	"morrison"
"plume"	"morrison"

Part 4: Experience Recap

Question 1

What can you say about your experience using Oracle, MongoDB and Neo4j in this exercise?

I am a database track student therefore, most of my classes are based on data base related. All my prior database classes thought me about relational database only, I never knew there is a No SQL exist. MongoDB is more likely creating database by writing comment on MongoDB's ide and it has its own programing language. I found more fun writing code on Neo4j because its graphical representation makes it more visible and the relationship between the node shows the arrow which is visibly tells the user about the relationship among nodes. I liked all three-database platform and it's always a good experience.

Question 2

What can you say about the differences in use among the three systems (Oracle, MongoDB and Neo4j)?

Oracle is a relational database management system. Oracle environment is more organize by separating data into the table and it normalize data to reduce redundancy. oracle SQL would be a language you will use to query an oracle database. MongoDB is a folder system database, it will remotely store the data to the right folder. You can add picture, video or any information you like about that person or object. For example, anything you do in FB it will store in MongoDB in one folder. If someone looking for me on FB then MongoDB will set the file that saved from FB. Most company use Oracle and MongoDB. Neo4j is a graph data store, it uses for graphical representation to show relationship among data element. Its graphically shows all the nodes and the relationships. It is fun to use and its project data very well.

Question 3

Identify why you would use each of these systems:

a. **Oracle:** I will use Oracle because it is one of the most famous database management system. If I use Oracle then it will be easier for me to fit into majority company. Oracle is a database that delivers excellent performance when challenged with demanding tasks and its good for use relational database.
b. **MongoDB**: I will use MongoDB because, MongoDB language *is to* implement a data store that provides high performance, high availability, and automatic scaling. MongoDB is extremely simple to install and implement. MongoDB uses JSON or BSON documents to store data.
c. **Neo4j:** I will use Neo4j because, Neo4j is a hottest data store right now. Neo4j graph database allows me to connect network, data center, and IT assets in order to get important insights into the relationships between different operations within your network. Also, its graphical representation make life easier to understand the database better.

Question 4

What different view of the data do you get from using Oracle, MongoDB and Neo4j?

Oracle: Oracle database management system required to do normalization, we can do 2NF or 3NF that will separate the data by its category. Also, we can create logical data schema which is gives the user a good view of entire database management system.

MongoDB: MongoDB is another great way to create database. It is one of the popular Non-SQL database in the marker. We can use this for any big data store. It was a new experience for me to create a MongoDB and its data representation is easy to understand.

Neo4j: Neo4j also a Non-SQL database. I got totally new experience by creating Neo4j database. Because its graphical representation of data is amazing. All the nodes are labeled also the relationships between the node has arrow to show what kind of relationship do they have.

Question 5

Describe how a company can benefit from using all three systems. Use what you have learned in class and by completing this exercise to validate your opinion.

I think all three are very powerful database platform. If any company use all of them within the company then they will be benefited many way and company's data will be well developed and organized. Oracle database management system can maintain one part of the company's database in order to create a relational database. The company will be benefited economically by using oracle because, oracle reduce data redundancy which will help to minimize the storage. If the company need to work with big data then definitely, they should use Non-SQL database system MongoDB. It will help the company to get the data from outside of the company's database. MongoDB database service enables us to build applications that are faster, reliable, handles a diverse range of data and manage applications in an effective way. In order to maintain the level of efficiency company need to use Neo4j, it is also a Non-SQL database platform. It will help the company to gather large amount of data for individual employee or business object. Neo4j enables organizations to unlock the business value of connections, influences and relationships in data. Neo4j's vision will be to help the company make sense of data by graphical representation.