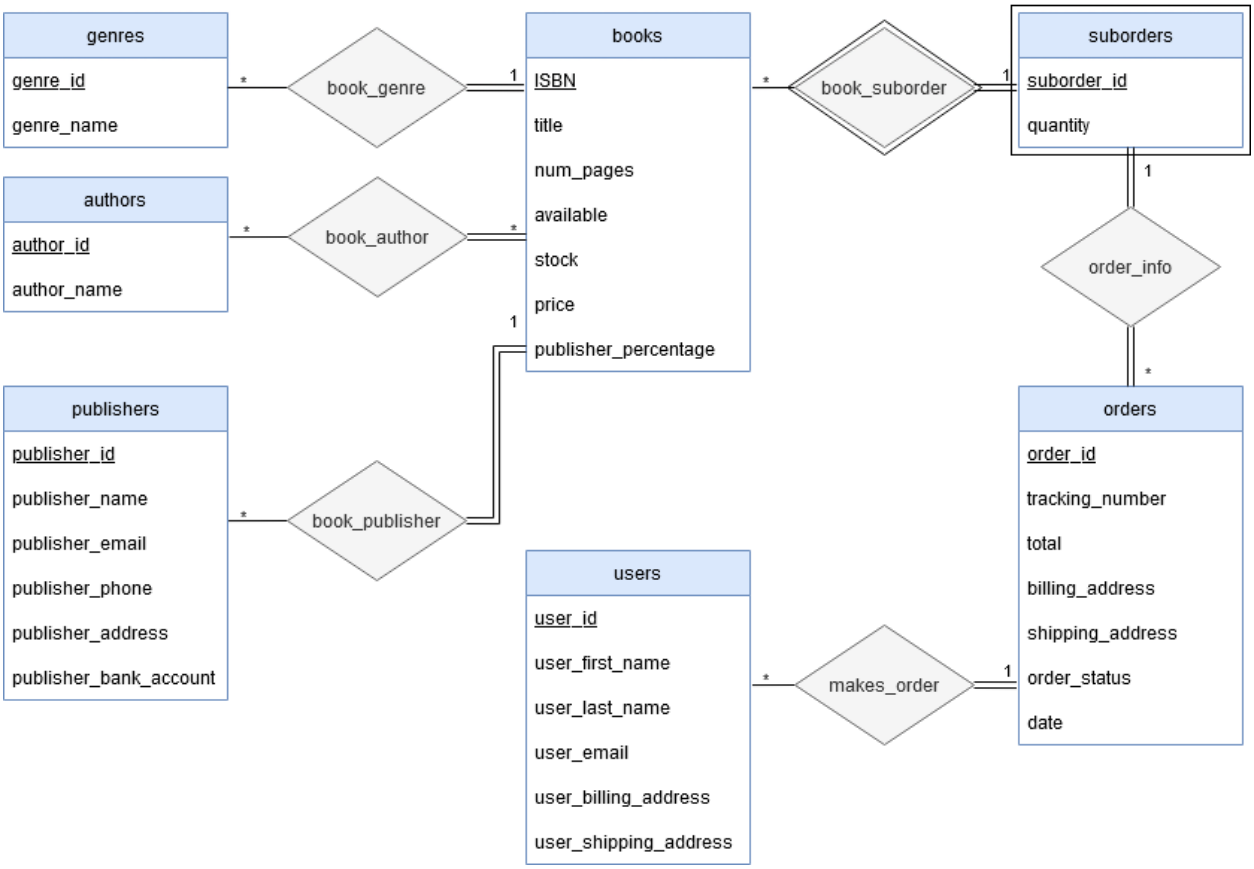
COMP 3005 — Project Report

Due: Sunday December 19, 23:59
Name: Michael Quach
Student Number: 101179729

1 Conceptual Design

ER Diagram



The following assumptions are made regarding cardinalities and participation in the ER diagram:

The ISBN of a book is unique, and thus no two books may have the same ISBN. A book must have exactly one genre, exactly one publisher, and at least one author. It may be the case that a book may have the same title, genre, publisher, etc, so long as it has different ISBNs.

A genre's id is unique and its name is also unique, no two genres may have the same name.

An author's id is unique. However, it may be the case that two authors may share the same first name and/or last name. An author can write many books.

A publisher's id is unique. In addition, a publisher's email address, phone number, and bank account number are unique, no two publishers may have the same email address, phone number, and/or bank account number. It may be the case that multiple publishers with the same name coexist.

A user's id is unique. However, it may be the case that two users may share the same first name, last name, billing address, and/or shipping address. Importantly, a user's email is unique – no two users may have the same email address. A user can make many orders. Users won't have phones as an email address is sufficient to contact them. Additionally, users won't have passwords as that is beyond the scope of the project.

An order's id is unique. It may be the case that two orders may share the same user id, total cost, billing address, shipping address, order status, and/or date. However, an order's tracking number must be unique, and thus no two orders may share the same tracking number. An order must be made by exactly one user. Additionally, each order must be correlated with at least one suborder. A suborder details exactly one book ordered (by ISBN) and the quantity of that book ordered. A suborder must belong to exactly one order. It may be the case that a book is present in many different suborders.

Addresses such as billing and shipping addresses are stored as a single string.

Further assumptions are made regarding the database:

When removing books from the store, it won't be the case that the books are deleted from the database, but instead said books are no longer visible to the user at the storefront. In doing so, financial records (e.g. book orders) are kept intact.

A genre cannot be deleted so long as at least one book is part of that genre, however it may be the case that a genre exists with no books falling under that description.

An author cannot be deleted so long as at least one book is written by that author, however it may be the case that an author is not the writer of any of the books in our database.

A publisher cannot be deleted so long as at least one book is published by that publisher, however it may be the case that a publisher has not published any books within our database.

A user can only be deleted from the database should no orders be associated with that user.

Should an order be deleted, its corresponding suborders will be deleted as well – however it generally will not be the case that orders will be deleted, as this would interfere with the bookstore's financial bookkeeping.

Generally books won't be deleted from the database as this would result in nulling or deleting corresponding suborders. The functionality of removing books from the bookstore has been discussed above.

In the scope of the project, these deletions won't be operations available to the user on either interface (removing books operates by hiding said books from the storefront as explained earlier). These assumptions have been made to give context to cascade effects (or lack thereof for some relations) in the database implementation.

All entities' attributes are non-null, meaning that when users or owners add a new entity (e.g. a new book or a new publisher), all attributes must be known and supplied.

The bookstore's expenditures consists only of the publisher percentages that it must pay for each book sale. For example, if a book's publisher percentage is 0.05 (5%), and the book's price is $20.00, the expenditure for that book is $1.00.

The status of an order refers to whether the order has been placed, in transit, or delivered.

2 Reduction to Relation Schemas

$books(\underline{ISBN}, title, genre\_id, publisher\_id, num\_pages, available, stock, price, publisher\_percentage)$

$genres(\underline{genre\_id}, genre\_name)$

$authors(\underline{author\_id}, author\_name)$

$publishers(\underline{publisher\_id}, publisher\_name, publisher\_email, publisher\_phone, publisher\_address, publisher\_bank\_account)$

$users(\underline{user\_id}, user\_first\_name, user\_last\_name, user\_email, user\_billing\_address, user\_shipping\_address)$

$orders(\underline{order\_id}, user\_id, tracking\_number, total, billing\_address, shipping\_address, order\_status, date)$

$suborders(\underline{order\_id}, \underline{suborder\_id}, ISBN, quantity)$

$book\_author(\underline{ISBN}, \underline{author\_id})$

3 Normalization of Relation Schemas

The following are the set of nontrivial functional dependencies for the books relation:

$F = \{$
$ISBN \rightarrow title$
$ISBN \rightarrow genre\_id$
$ISBN \rightarrow publisher\_id$
$ISBN \rightarrow num\_pages$
$ISBN \rightarrow available$
$ISBN \rightarrow stock$
$ISBN \rightarrow price$
$ISBN \rightarrow publisher\_percentage\}$

$ISBN$ is the sole key in each nontrivial functional dependency. To examine whether $ISBN$ is a superkey for the relation, we determine $(ISBN)^+$, the closure of $ISBN$ under F:

$result = ISBN$
$ISBN \rightarrow genre\_id : result = ISBN, genre\_id$
$ISBN \rightarrow publisher\_id : result = ISBN, genre\_id, publisher\_id$
$ISBN \rightarrow num\_pages : result = ISBN, genre\_id, publisher\_id, num\_pages$
$ISBN \rightarrow available : result = ISBN, genre\_id, publisher\_id, num\_pages, available$
$ISBN \rightarrow stock : result = ISBN, genre\_id, publisher\_id, num\_pages, available, stock$
$ISBN \rightarrow price : result = ISBN, genre\_id, publisher\_id, num\_pages, available, stock, price$
$ISBN \rightarrow publisher\_percentage : result = ISBN, genre\_id, publisher\_id, num\_pages, available,$
$stock, price, publisher\_percentage$
$(ISBN)^+ = (ISBN, genre\_id, publisher\_id, num\_pages, available, stock, price, publisher\_percentage)$

Thus we find that $ISBN$ is a superkey for the books relation. As for all nontrivial functional dependencies, $\alpha \rightarrow \beta, \alpha$ is a superkey for the relation, the books relation is thus in BCNF.

The following are the set of nontrivial functional dependencies for the genres relation:

$F = \{$
$genre\_id \rightarrow genre\_name$
$genre\_name \rightarrow genre\_id\}$

$genre\_id$ and $genre\_name$ are the keys in the nontrivial functional dependency. To examine whether $genre\_id$ is a superkey for the relation, we determine $(genre\_id)^+$, the closure of $genre\_id$ under F:

$result = genre\_id$
$genre\_id \rightarrow genre\_name : result = genre\_id, genre\_name$
$(genre\_id)^+ = (genre\_id, genre\_name)$

Next, examining whether $genre\_name$ is a superkey for the relation, we determine $(genre\_name)^+$, the closure of genre_name under F:

$result = genre\_name$
$genre\_name \rightarrow genre\_id : result = genre\_id, genre\_name$
$(genre\_name)^+ = (genre\_id, genre\_name)$

Thus we find that $genre\_name$ is a superkey for the genre relation. As for all nontrivial functional dependencies, $\alpha \rightarrow \beta, \alpha$ is a superkey for the relation, the genres relation is thus in BCNF.

The following are the set of nontrivial functional dependencies for the authors relation:

$F = \{$
$author\_id \rightarrow author\_name\}$

$author\_id$ is the sole key in each nontrivial functional dependency. To examine whether $author\_id$ is a superkey, we determine $(author\_id)^+$, the closure of $author\_id$ under F:

$result = author\_id$
$author\_id \rightarrow author\_first\_name : result = author\_id, author\_name$
$(author\_id)^+ = (author\_id, author\_name)$

Thus we find that $author\_id$ is a superkey for the authors relation. As for all nontrivial functional dependencies, $\alpha \rightarrow \beta, \alpha$ is a superkey for the relation, the authors relation is thus in BCNF.

The following are the set of nontrivial functional dependencies for the publishers relation:

$F = \{$
$publisher\_id \rightarrow publisher\_name$
$publisher\_id \rightarrow publisher\_email$
$publisher\_id \rightarrow publisher\_phone$
$publisher\_id \rightarrow publisher\_address$
$publisher\_id \rightarrow publisher\_bank\_account$
$publisher\_email \rightarrow publisher\_id$
$publisher\_phone \rightarrow publisher\_id$
$publisher\_bank\_account \rightarrow publisher\_id\}$

$publisher\_id, publisher\_email, publisher\_phone, publisher\_bank\_account$ are the keys the nontrivial functional dependencies. We first examine whether $publisher\_id$ is a superkey for the relation, determining $(publisher\_id)^+$, the closure of $publisher\_id$ under F:

$result = publisher\_id$
$publisher\_id \rightarrow publisher\_name : result = publisher\_id, publisher\_name$
$publisher\_id \rightarrow publisher\_email : result = publisher\_id, publisher\_name, publisher\_email$
$publisher\_id \rightarrow publisher\_phone : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_phone$
$publisher\_id \rightarrow publisher\_address : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_phone, publisher\_address$
$publisher\_id \rightarrow publisher\_bank\_account : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_phone, publisher\_address, publisher\_bank\_account$
$(publisher\_id)^+ = (publisher\_id, publisher\_name, publisher\_email, publisher\_phone, publisher\_address,$
$publisher\_bank\_account)$

Thus we find that $publisher\_id$ is a superkey for the publishers relation.

Next we examine whether $publisher\_email$ is a superkey for the publishers relation, determining $(publisher\_email)^+$, the closure of $publisher\_email$ under F:

$result = publisher\_email$
$publisher\_email \rightarrow publisher\_id : result = publisher\_id, publisher\_email$
$publisher\_id \rightarrow publisher\_name : result = publisher\_id, publisher\_name, publisher\_email$
$publisher\_id \rightarrow publisher\_phone : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_phone$
$publisher\_id \rightarrow publisher\_address : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_phone, publisher\_address$
$publisher\_id \rightarrow publisher\_bank\_account : result = publisher\_id, publisher\_name, publisher\_email,$

$publisher\_phone, publisher\_address, publisher\_bank\_account$
$(publisher\_email)^+ = (publisher\_id, publisher\_name, publisher\_email, publisher\_phone,$
$publisher\_address, publisher\_bank\_account)$

Thus we find that *publisher_email* is a superkey for the publishers relation.

Next we examine whether *publisher_phone* is a superkey for the publishers relation, determining $(publisher\_phone)^+$, the closure of *publisher_phone* under F:

$result = publisher\_phone$
$publisher\_phone \rightarrow publisher\_id : result = publisher\_id, publisher\_phone$
$publisher\_id \rightarrow publisher\_name : result = publisher\_id, publisher\_name, publisher\_phone$
$publisher\_id \rightarrow publisher\_email : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_phone$
$publisher\_id \rightarrow publisher\_address : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_phone, publisher\_address$
$publisher\_id \rightarrow publisher\_bank\_account : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_phone, publisher\_address, publisher\_bank\_account$
$(publisher\_phone)^+ = (publisher\_id, publisher\_name, publisher\_email, publisher\_phone,$
$publisher\_address, publisher\_bank\_account)$

Thus we find that *publisher_phone* is a superkey for the publishers relation.

Finally we examine whether publisher_bank_account is a superkey for the publishers relation, determining $(publisher\_bank\_account)^+$, the closure of *publisher_bank_account* under F:

$result = publisher\_bank\_account$
$publisher\_bank\_account \rightarrow publisher\_id : result = publisher\_id, publisher\_bank\_account$
$publisher\_id \rightarrow publisher\_name : result = publisher\_id, publisher\_name, publisher\_bank\_account$
$publisher\_id \rightarrow publisher\_email : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_bank\_account$
$publisher\_id \rightarrow publisher\_phone : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_phone, publisher\_bank\_account$
$publisher\_id \rightarrow publisher\_address : result = publisher\_id, publisher\_name, publisher\_email,$
$publisher\_phone, publisher\_address, publisher\_bank\_account$
$(publisher\_bank\_account)^+ = (publisher\_id, publisher\_name, publisher\_email, publisher\_phone,$
$publisher\_address, publisher\_bank\_account)$

Thus we find that *publisher_bank_account* is a superkey for the publishers relation. As for all nontrivial functional dependencies, $\alpha \rightarrow \beta, \alpha$ is a superkey for the relation, the publishers relation is thus in BCNF.

The following are the set of nontrivial functional dependencies for the users relation:

$F = \{$
$user\_id \rightarrow user\_first\_name$
$user\_id \rightarrow user\_last\_name$
$user\_id \rightarrow user\_email$
$user\_id \rightarrow user\_billing\_address$
$user\_id \rightarrow user\_shipping\_address$
$user\_email \rightarrow user\_id$
$user\_id \rightarrow publisher\_percentage\}$

$user\_id$ and $user\_email$ are the keys in the nontrivial functional dependencies. To examine whether $user\_id$ is a superkey, we determine $(user\_id)^+$, the closure of $user\_id$ under F:

$result = user\_id$
$user\_id \rightarrow user\_first\_name : result = user\_id, user\_first\_name$
$user\_id \rightarrow user\_last\_name : result = user\_id, user\_first\_name, user\_last\_name$
$user\_id \rightarrow user\_email : result = user\_id, user\_first\_name, user\_last\_name, user\_email$
$user\_id \rightarrow user\_billing\_address : result = user\_id, user\_first\_name, user\_last\_name, user\_email,$
$user\_billing\_address$
$user\_id \rightarrow user\_shipping\_address : result = user\_id, user\_first\_name, user\_last\_name, user\_email,$
$user\_billing\_address, user\_shipping\_address$
$(user\_id)^+ = (user\_id, user\_first\_name, user\_last\_name, user\_email, user\_billing\_address,$
$user\_shipping\_address)$

We find that $user\_id$ is a superkey for the users relation. Next we examine whether $user\_email$ is a superkey, determining $(user\_email)^+$, the closure of $user\_email$ under F:

$result = user\_email$
$user\_email \rightarrow user\_id : result = user\_id, user\_email$
$user\_id \rightarrow user\_first\_name : result = user\_id, user\_email, user\_first\_name$
$user\_id \rightarrow user\_last\_name : result = user\_id, user\_email, user\_first\_name, user\_last\_name$
$user\_id \rightarrow user\_billing\_address : result = user\_id, user\_first\_name, user\_last\_name, user\_email,$
$user\_billing\_address$
$user\_id \rightarrow user\_shipping\_address : result = user\_id, user\_first\_name, user\_last\_name, user\_email,$
$user\_billing\_address, user\_shipping\_address$
$(user\_email)^+ = (user\_id, user\_first\_name, user\_last\_name, user\_email, user\_billing\_address,$
$user\_shipping\_address)$

We find that $user\_email$ is a superkey for the users relation. As for all nontrivial functional dependencies, $\alpha \rightarrow \beta, \alpha$ is a superkey for the relation, the users relation is thus in BCNF.

The following are the set of nontrivial functional dependencies for the orders relation:

$F = \{$
$order\_id \rightarrow user\_id$
$order\_id \rightarrow tracking\_number$
$order\_id \rightarrow total$
$order\_id \rightarrow billing\_address$
$order\_id \rightarrow shipping\_address$
$order\_id \rightarrow order\_status$
$order\_id \rightarrow date$
$tracking\_number \rightarrow order\_id\}$

$order\_id$ and $tracking\_number$ are the keys in the nontrivial functional dependencies. To examine whether $order\_id$ is a superkey:

$result = order\_id$
$order\_id \rightarrow user\_id : result = order\_id, user\_id$
$order\_id \rightarrow tracking\_number : result = order\_id, user\_id, tracking\_number$
$order\_id \rightarrow total : result = order\_id, user\_id, tracking\_number, total$
$order\_id \rightarrow billing\_address : result = order\_id, user\_id, tracking\_number, total, billing\_address$
$order\_id \rightarrow shipping\_address : result = order\_id, user\_id, tracking\_number, total, billing\_address,$
$shipping\_address$
$order\_id \rightarrow order\_status : result = order\_id, user\_id, tracking\_number, total, billing\_address,$
$shipping\_address, order\_status$
$order\_id \rightarrow date : result = order\_id, user\_id, tracking\_number, total, billing\_address, shipping\_address,$
$order\_status, date$
$(order\_id)^+ = (order\_id, user\_id, tracking\_number, total, billing\_address, shipping\_address,$
$order\_status, date)$

We find that $order\_id$ is a superkey for the orders relation. Next we examine whether $tracking\_number$ is a superkey:

$result = tracking\_number$
$tracking\_number \rightarrow order\_id : result = order\_id, tracking\_number$
$order\_id \rightarrow user\_id : result = order\_id, user\_id, tracking\_number$
$order\_id \rightarrow total : result = order\_id, user\_id, tracking\_number, total$
$order\_id \rightarrow billing\_address : result = order\_id, user\_id, tracking\_number, total, billing\_address$
$order\_id \rightarrow shipping\_address : result = order\_id, user\_id, tracking\_number, total, billing\_address,$
$shipping\_address$
$order\_id \rightarrow order\_status : result = order\_id, user\_id, tracking\_number, total, billing\_address,$
$shipping\_address, order\_status$
$order\_id \rightarrow date : result = order\_id, user\_id, tracking\_number, total, billing\_address,$
$shipping\_address, order\_status, date$
$(tracking\_number)+ = (order\_id, user\_id, tracking\_number, total, billing\_address,$
$shipping\_address, order\_status, date)$

Thus we find that $tracking\_number$ is a superkey for the orders relation. As for all nontrivial functional dependencies, $\alpha \rightarrow \beta, \alpha$ is a superkey for the relation, the orders relation is thus in BCNF.

The following are the set of nontrivial functional dependencies for the suborders relation:

$F = \{$
$order\_id, suborder\_id \rightarrow ISBN$
$order\_id, suborder\_id \rightarrow quantity\}$

$order\_id$, $suborder\_id$ is the sole key in each nontrivial functional dependency. To examine whether $order\_id, suborder\_id$ is a superkey for the relation, we determine $(order\_id, suborder\_id)^+$, the closure of $order\_id, suborder\_id$ under F:

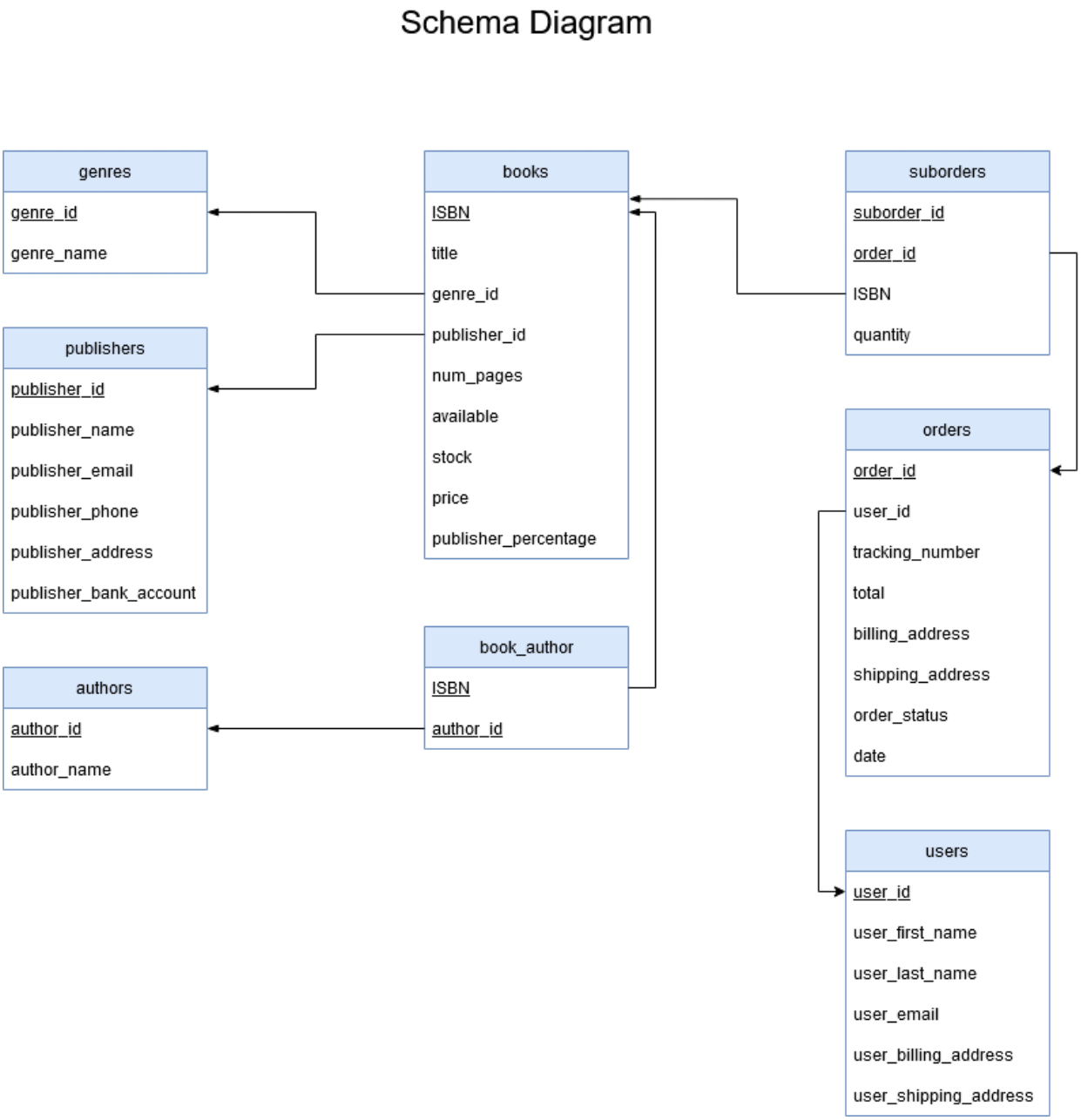
$result = order\_id, suborder\_id$
$order\_id, suborder\_id \rightarrow ISBN : result = order\_id, suborder\_id, ISBN$
$order\_id, suborder\_id \rightarrow quantity : result = order\_id, suborder\_id, ISBN, quantity$
$(order\_id, suborder\_id)^+ = (order\_id, suborder\_id, ISBN, quantity)$

Thus we find that $order\_id, suborder\_id$ is a superkey for the suborders relation. As for all nontrivial functional dependencies, $\alpha \rightarrow \beta, \alpha$ is a superkey for the relation, the suborders relation is thus in BCNF.

The *book_author* **relation** has no nontrivial functional dependencies. Thus as every functional dependency in this relation is trivial, it is by definition in BCNF.

We have therefore found that all our reduced relations are in BCNF and thus are in good normal form. In addition, as each relation was already originally in BCNF, we can assert that each relation is lossless and dependency preserving – these relations were never decomposed.

4 Database Schema Diagram



5 Implementation

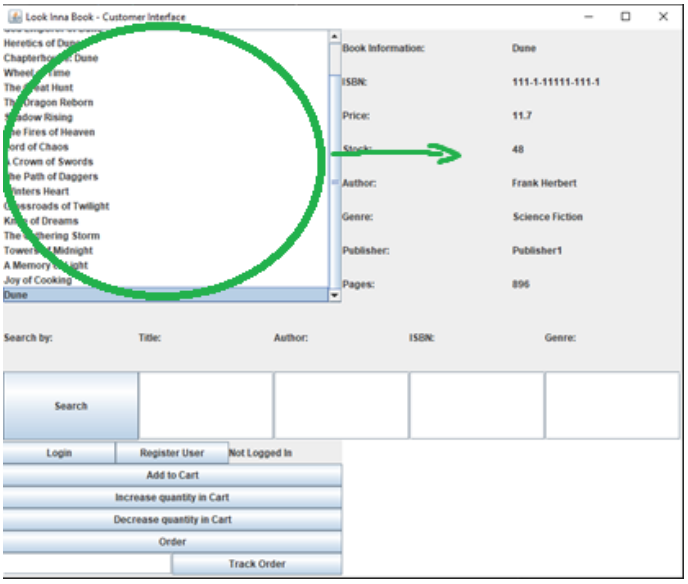
A Java Swing desktop application was made to integrate with the bookstore database. Three main components of the application exist – Application.java, CustomerBookstore.java, and OwnerBookstore.java. Application.java is a class that exists to launch either the customer bookstore application interface or the owner bookstore application interface based on the user's input.

CustomerBookstore roughly follows the MVC design pattern. Though the Controller and View aspects are combined into this single class, their methods are kept mostly separate. The database still represents the underlying model which supplies the bookstore information. Here the class displays information to a customer based on queries retrieved from the database. User input is handled by the class (e.g. selecting a book retrieves additional information about that book). The model may be updated by user input as well (e.g. when a user places an order), and thus the user input is transformed into a query to the database to update its information. Once this update has occurred, the information is retrieved and displayed for the user, thus fulfilling the MVC design pattern.

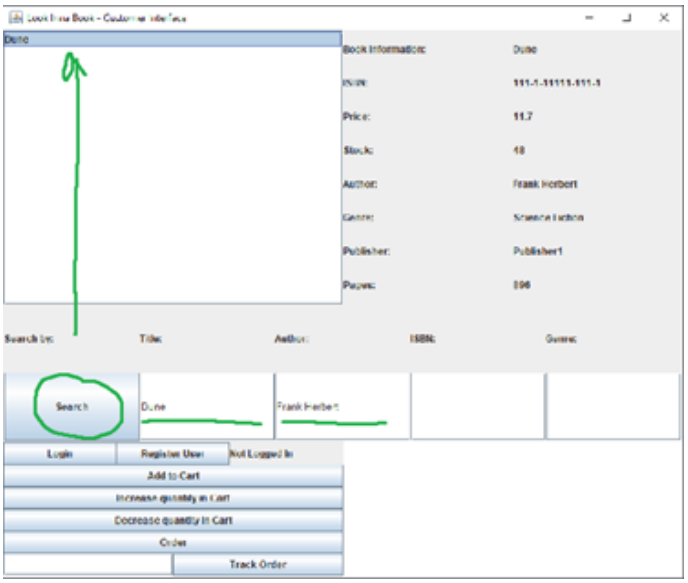
Likewise, OwnerBookstore follows the same MVC design pattern as above. The View and Controller components of the class are separated into dedicated methods to either handle displaying updated information retrieved from the database, or to handle translating user input into manipulating the database model, respectively. In both classes, all methods are commented to provide further insight and clarity regarding its functionality.

Various scenarios of using the Customer Bookstore Application are documented below:

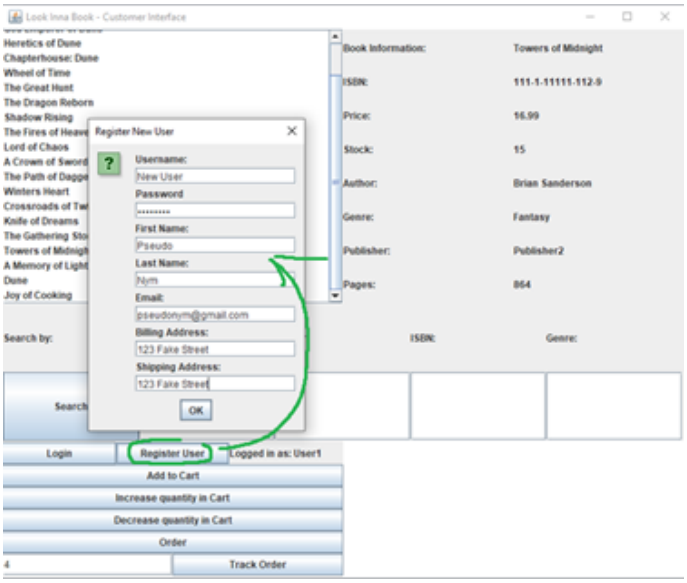
1. Browsing the bookstore:
   Once the user launches the application, the left pane displays all available books to the customer. Selecting a book from this pane will display further information about the book on the right pane.



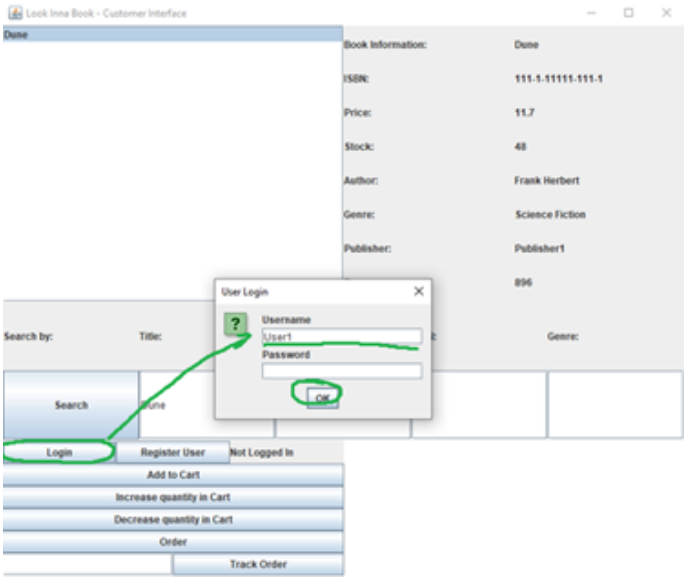
2. Searching the bookstore:
   The user can enter a title, author, ISBN, and/or genre into the fields to search the bookstore for all available books that match all of the input attributes (case-sensitive and exact matches). Blank fields are ignored in the search. Once the user has entered inputs into the desired fields, pressing the "Search" button will populate the left pane with the books that match those attributes.



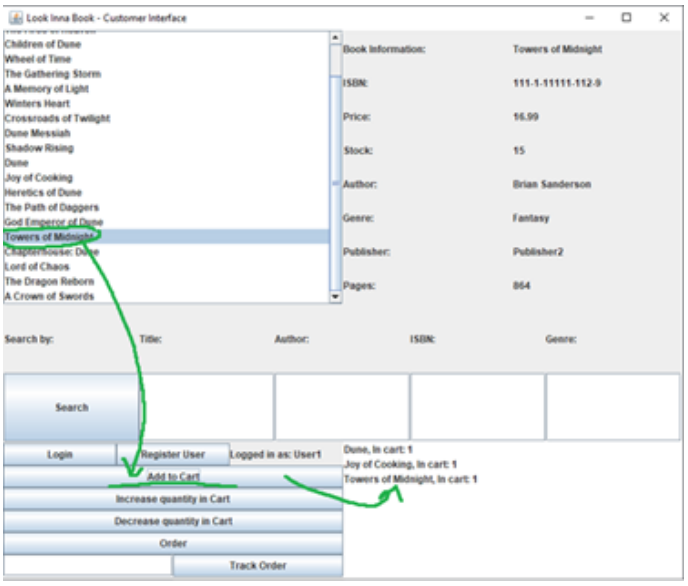
3. Registering new user:
   A new user can be registered with the bookstore by pressing the "Register User" button. A popup is displayed with fields for the user to fill out. Each field must be filled, no fields may be left blank. Once filled, pressing "OK" will register a new user with those details with the bookstore.



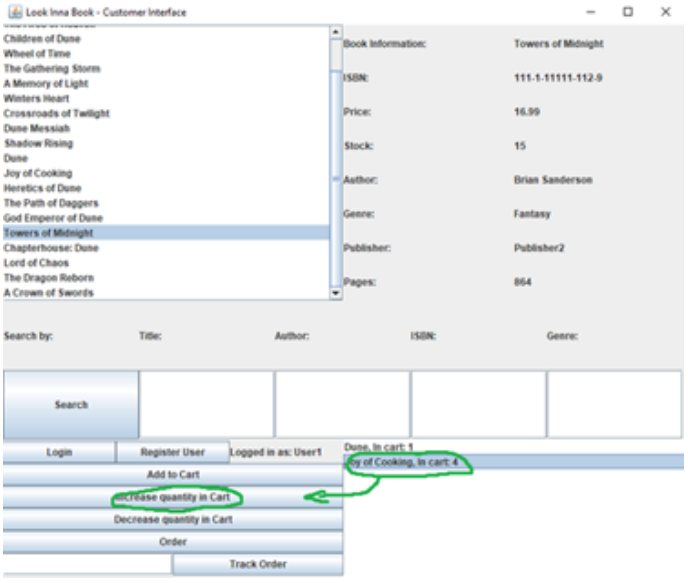
4. Adding books to cart:
   A user can add any number of books to their cart. The user needs to first log in to an account pressing the "Login" button and supplying a valid username (no password has been implemented at this time).



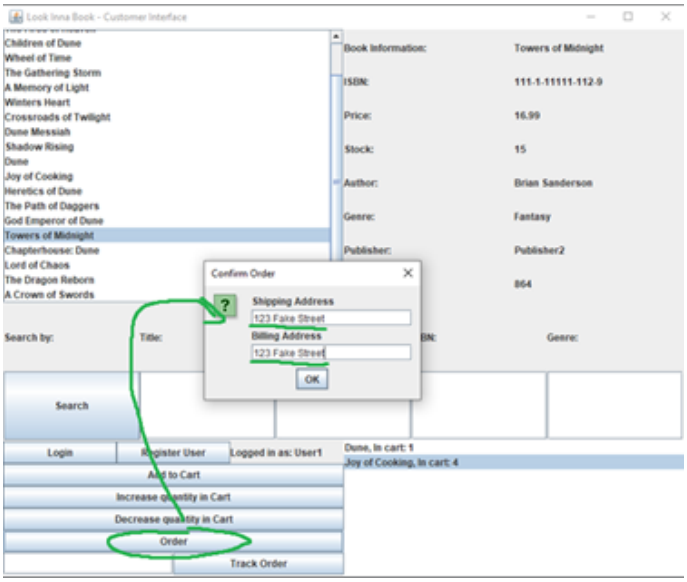
Once the user has logged in, books can be added by selecting a book from the left pane and pressing the "Add to Cart" button. The selected book will then be added to the cart, which is displayed at the bottom right pane. Any number of books from the bookstore can be added to the cart.



A book in a cart can have its quantity adjusted by first selecting it, then pressing the "Increase quantity in Cart" button, or "Decrease quantity in Cart" button. Note that the quantity of a book in a user's cart is limited by the stock available in the store. Decreasing the cart quantity of a book in the cart will remove it from the cart.

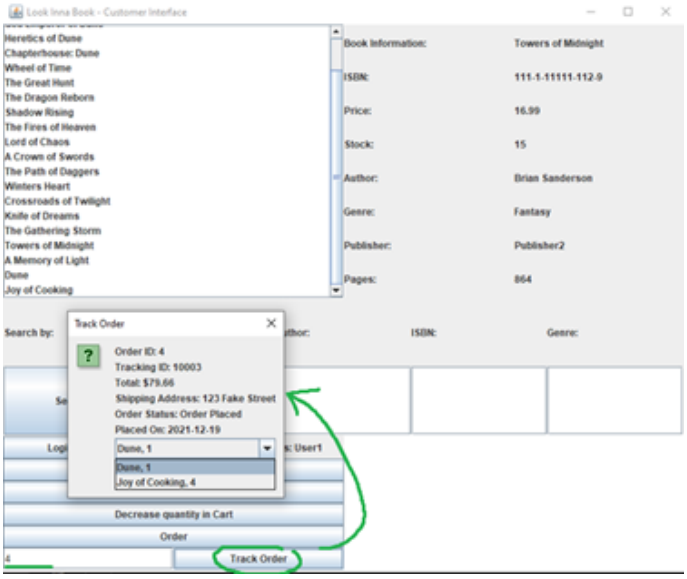


5. Confirming order:
   Once the user has added all the items they wish to add to their cart, they can check out by pressing the "Order" button. A popup will appear with fields for a shipping address and billing address for the user to fill in. These fields must both be filled for an order to be placed.



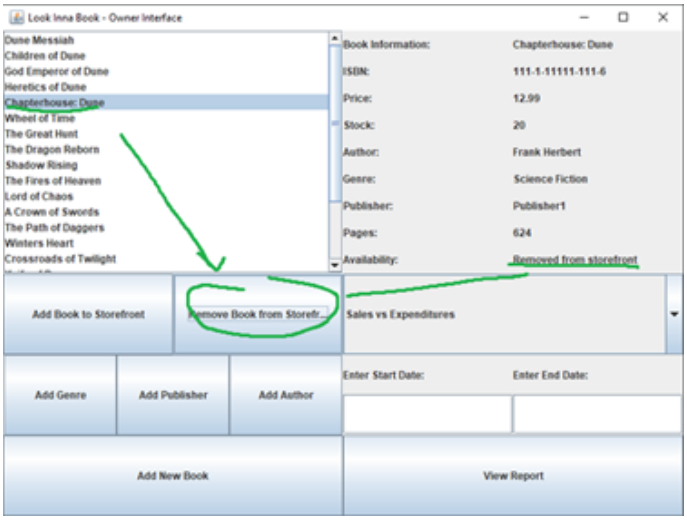
   Note that a user must be logged in to place an order. Once the "OK" button is pressed, the order is placed, and a popup providing confirmation and tracking IDs is displayed to the user.

6. Tracking order:
   Using the order ID number provided, a user can track their order by inputting that ID into the “Track Order” field. Once input, pressing the “Track Order” button will display a popup with that order’s status.



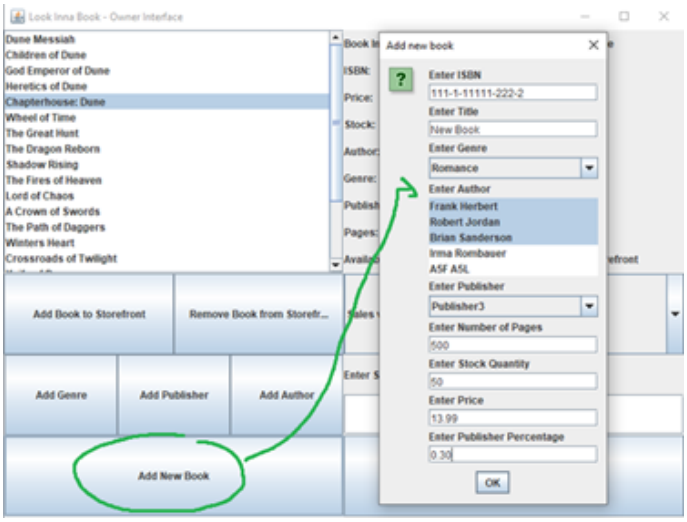
Various scenarios of using the Owner Bookstore Application are documented below:

1. Removing books from the storefront:
   An owner can remove a book from the storefront by selecting that book from the top left pane. Once selected, pressing "Remove Book from Storefront" will remove that book from the storefront, making it unavailable to customers.

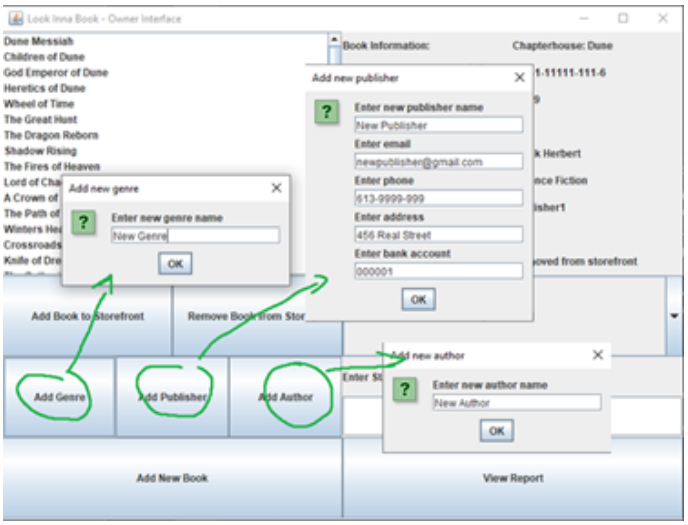


   Likewise, an unavailable book can be made available by following the same process, but pressing "Add Book to Storefront" instead.

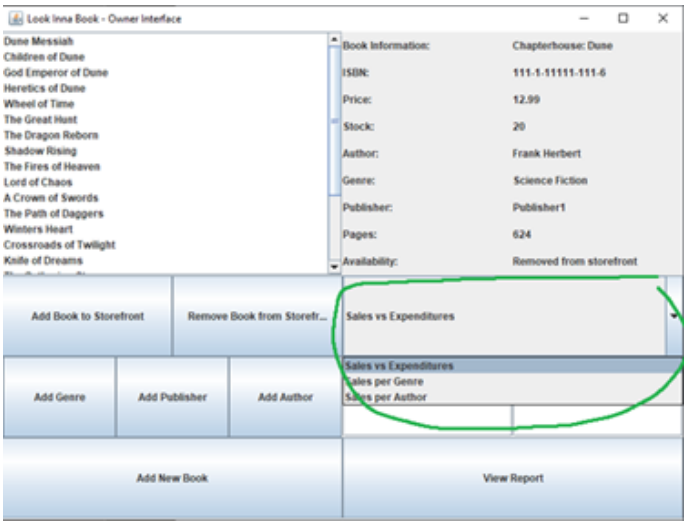
2. Adding new books to the bookstore:
   An owner can add a new book to the bookstore by pressing the "Add New Book" button. A popup will be displayed with fields for each attribute of the book. Note that each field must be filled in for the book to be added. Additionally, multiple authors may be selected for the book by Control + Clicking each author. Pressing "OK" will add this new book to the bookstore.



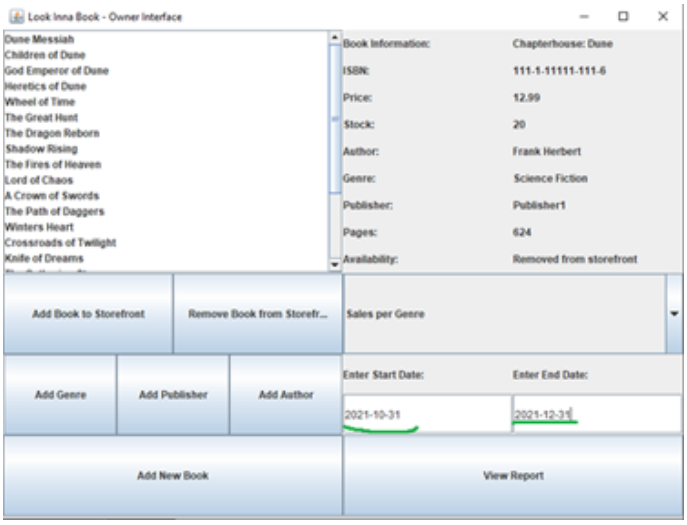
Please note that when adding a new book, existing genres, authors, and publishers can only be chosen from. Should a new book require a new genre, author, or publisher, these new entities can be created by pressing their respective "Add ..." buttons first. Then the new book can be added as described before.



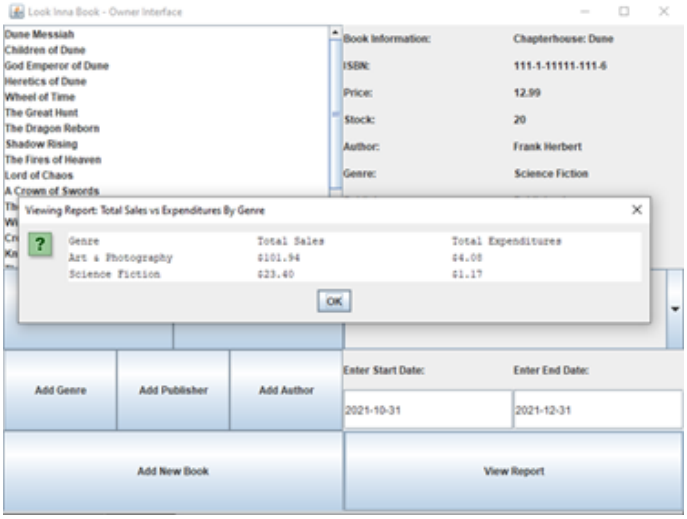
3. Viewing reports:
   An owner can view reports regarding Total Sales vs Expenditures, Sales vs Expenditures per Genre, and Sales vs Expenditures per Author, each over a user defined period. First the owner can select the report type from the dropdown menu on the right pane:



Once selected, the owner can define the starting and end dates by filling in the two fields below. Note that these inputs must be of the format YYYY-MM-DD.



After pressing the "View Report" button, a report of the selected type will be generated and displayed to the owner.



Please read the README.txt for additional information on how to configure and run the application and database.

6 Github Repository

The repository containing the database, and application can be accessed here:
https://github.com/michaeldquach/COMP-3005-Final-Project

A README.txt has been included for additional information on how to configure and run the application and database.

A directory named "SQL" has been included, consisting of DDL.sql, Functions and Triggers.sql, Populate Database.sql, and Queries.sql. All are commented for further information.

7 Appendix

Availability time slots:

1:00 - 1:20
1:20 - 1:40
1:40 - 2:00

2:00 - 3:20
2:20 - 3:40
2:40 - 3:00