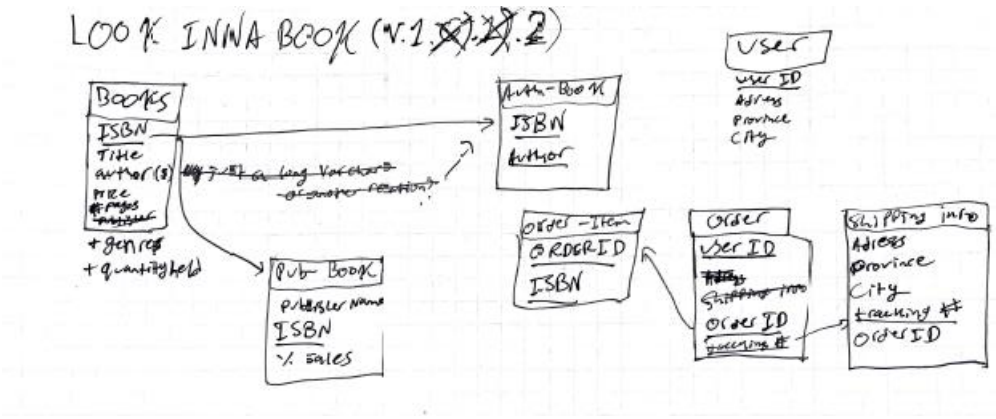
Look Inna Book Final Project

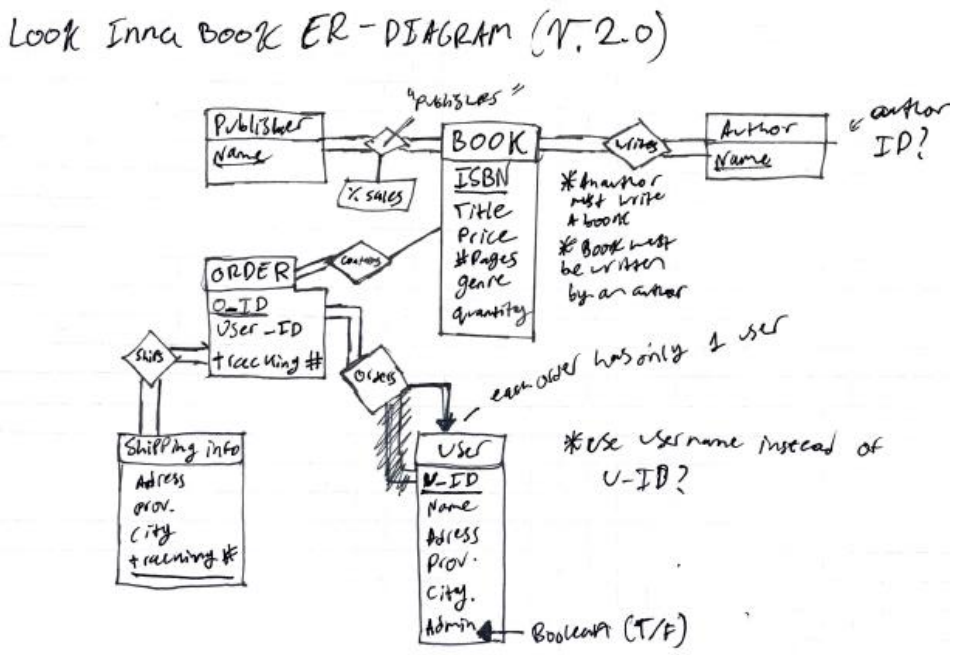
Ahmad Alkfri (101187272)

Conceptual Design

The conceptual design for this database went through 3 revisions to reach an acceptable final version. The first draft resembled a schema diagram more than an ER diagram, but it laid the groundwork down for future versions:

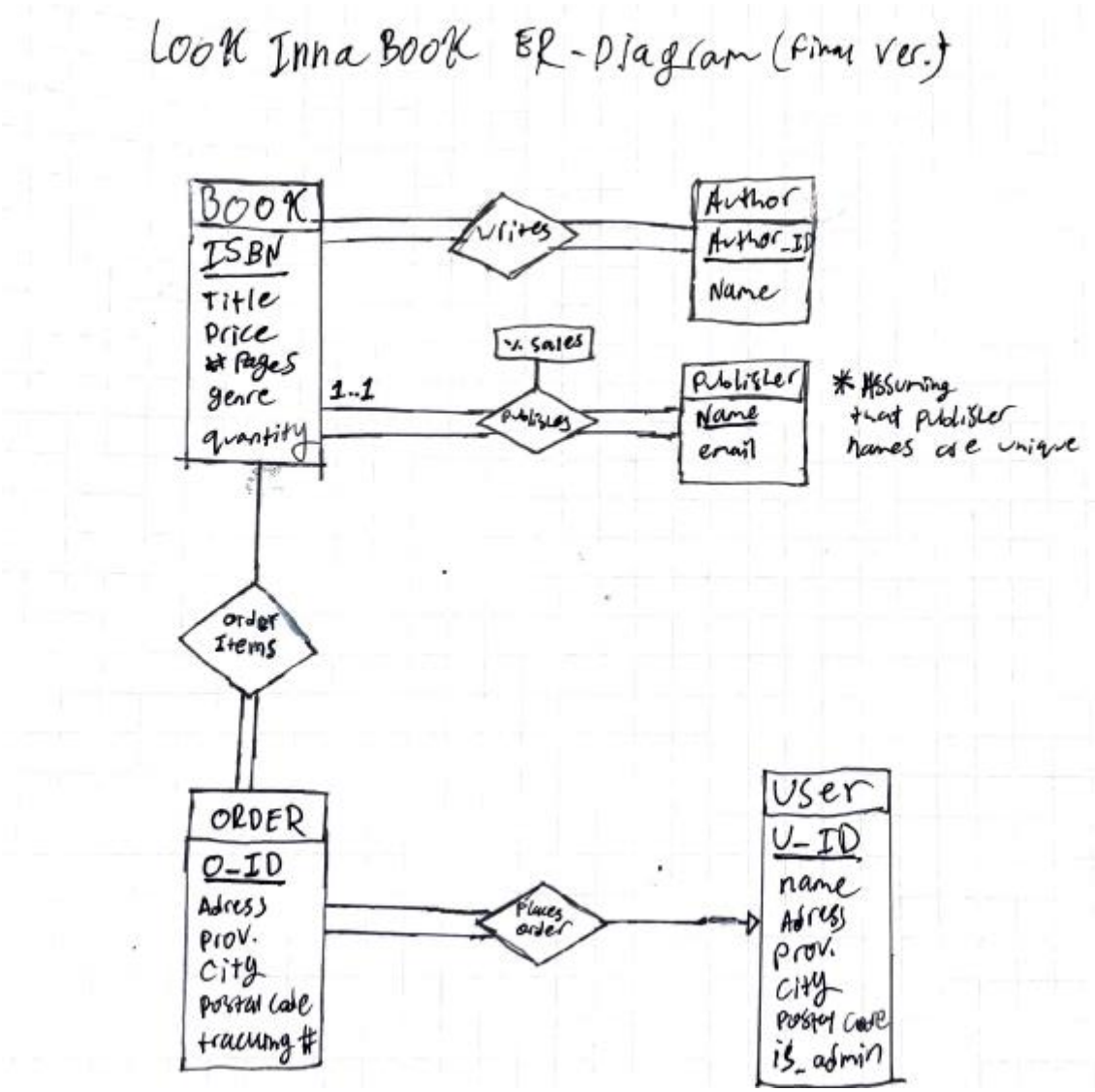


The general idea was to have a book entity set, an order entity set, and a user entity set. The rest of the entity sets would be created to store more specific information about the main entity sets. For example, the auth-book entity set stores the authors name and the ISBN of the book they wrote, to allow for multiple authors on one book. Other relations were just vague ideas, and at this point the ER diagram was not a proper one at all, so a second draft was made:



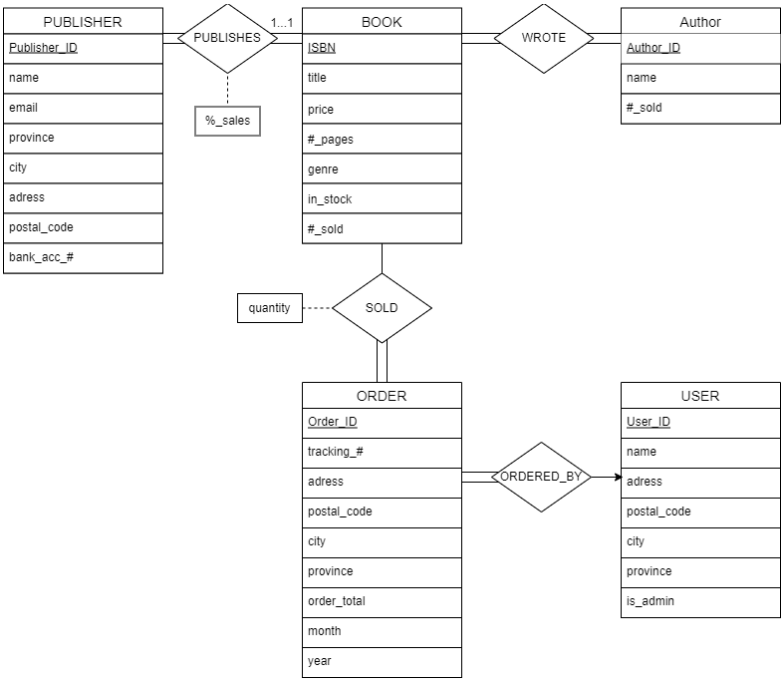
In this second draft, the relationship sets have been added to represent the interaction between the entity sets. Some distinction has also been made between total and partial participation in each

relationship set. “Book” has total participation in both “writes” and “publishes” because every book needs to have been both written by someone as well as published by a publisher. The participation of “order” in orders is total since every order needs to be ordered by a user. The design concerning the shipping information wasn’t quite satisfactory at this point, so one last rough draft was created:



In this revision of the ER diagram, the entity set “Book” has total participation in the “writes” and “publishes” relationships because each book must have been both written and published to be in the store. A book can have multiple authors, and an author can write may books so the relationship there is many to many. The “author_id” attribute has been added so that authors with the same name can exist in the database. The author entity set is needed because of the potential case where two authors with the same name wrote the same book. The publishes relationship is many to one: each book has at most one publisher, but one publisher can publish any number of books. The participation is total on both ends, since a company cannot be considered a publisher until it publishes a book. The publisher’s email has been added to the entity set so that an email can be sent to them to ask for more books when a given book’s quantity becomes too low. The “order items” relationship can be thought of as the “cart”. It contains the books that are being ordered, and the order they belong to. The order entity set has been

significantly changed. The shipping information is now stored with the order, as is the tracking number. The user ID is now stored in a relationship set rather than the entity set, and the participation of order in "places order" is total. The user entity is mostly the same, with a many to one relationship with "places order" since each order is placed by exactly one user, but a user can place multiple (or zero) orders. The property "is_admin" will be a boolean that will determine if a user is allowed to modify book details (adding/removing from the store). Using this final draft, the ER diagram for the Look Inna Book database was created:



This ER diagram is mostly the same as the previous version, but the entity sets now have all the required attributes. Order now has the year and moth the sale took place, so that monthly sales can be tracked. The publisher information is now far more detailed, and has their address, shipping information, and banking info. Book and Author keep track of their sales so that you can see the best-selling items by book, genre, and author.

Relation Schemas

Reducing the final ER diagram produces the following relations:

book(ISBN, title, price, #pages, genre, quantity, num_sold, Publisher_ID, percent_sales)

wrote(ISBN, Author_ID)

author(Author_ID, name, num_sold)

publisher(Publisher_ID, email, province, city, address, postal_code, bank_acc_num)

cart(ISBN, Order_ID, quantity)

order(Order_ID, Username, tracking_num, province, city, address, postal_code, month, year, order_total)

user(Username, password, province, city, address, postal_code, is_admin)

The schemas mostly match the ER diagram. The User_ID and name were replaced by username and password. The two other notable changes are the integration of the “ordered by” and “publishes” relations into the “order” and “book” relations. Since each book had to appear exactly once in the “publishes” relation, it was removed, and the information was appended to the “book” relation instead. That same logic applies to removing the “ordered_by” relation and just adding the username to each order instead.

Normalization

Book Schema

book(ISBN, title, price, #pages, genre, quantity, num_sold, Publisher_ID, percent_sales)

Set of functional dependencies, F:

$$F = \{ISBN \rightarrow ISBN, title, price, \#pages, genre, quantity, num\_sold, Publisher\_ID, percent\_sales\}$$

No other functional dependencies exist, since the only unique part of a book is the ISBN (duplicate titles *do* exist). This makes the book’s ISBN a super key, and the primary key for this relation. Since the ISBN is a super key, this dependency is in BCNF, so this relation is in good normal form.

Wrote Schema

wrote(ISBN, Author_ID)

Set of functional dependencies, F:

$$F = \{ISBN, Author\_ID \rightarrow ISBN, Author\_ID\}$$

The only functional dependency that exists is the trivial one shown above. A book can have more than one author, so ISBN does not determine author. An author can write multiple books so author cannot determine ISBN. The combination of ISBN and Author_ID creates the only super key for this schema. This means that this relation is in good normal form (BCNF).

Author Schema

author(Author_ID, name, num_sold)

Set of functional dependencies, F:

$$F = \{Author\_ID \rightarrow Author\_ID, name, num\_sold\}$$

Once again, no other functional dependencies exist because authors can have the same name, and they can also have the same number of books sold. Author_ID is the super key for this relation, so the functional dependency above does not violate BCNF, meaning this relation is in good normal form.

Publisher Schema

publisher(Publisher_ID, email, province, city, address, postal_code, bank_acc_num)

Set of functional dependencies, F:

$F = \{Publisher\_ID \rightarrow Publisher\_ID, email, province, city, address, posal\_code, bank\_acc\_num$
$bank\_acc\_num \rightarrow Publisher\_ID$ //Each publisher should have a unique bank account
$email \rightarrow Publisher\_ID$//Each publisher should have a different email to contact them through
$postal\_code \rightarrow province, city\}$//VIOLATES BCNF

Since the last dependency violates BCNF, publisher needs to be decomposed into 2 different relations:

$$R1 = postal\_code \cup province, city = postal(postal\_code, province, city)$$
$$R2 = \big(publisher - (province)\big)$$
$$= publisher(Publisher\_ID, email, adress, postal\_code, bank\_acc\_num)$$

This decomposition is dependency preserving and removes the violation in BCNF. However, this violation is quite minor, and the redundancy caused by it is very small, since there are thousands of postal codes in Canada. The space saved by not "redundantly" storing the province and city when you already have the postal code will be a few bytes of data per duplicate postal code. The users of the application will still have to enter their province and city every time, so the benefit of implementing this schema is next to nothing. As a design choice, this will be kept OUT of the database, and the publisher schema will store the province, city, and postal code (this applies for all schemas where an address is stored)

All other functional dependencies in F have super keys on the left-hand side (determining a super key makes that attribute a super key). Thus, none of these functional dependencies violate BCNF. No other dependencies exist, since there are cities with the same name in different provinces (Victoria in BC, Ontario, and NL).

Sold Schema

cart(ISBN, Order_ID, quantity)

Set of functional dependencies, F:

$F = \{ISBN, Order\_ID \rightarrow quantity\}$

This is the only functional dependency present in this relation, and it does not violate BCNF because ISBN and Order_ID make the super key for the relation.

Order Schema

order(Order_ID, username, tracking_num, province, city, address, postal_code, month, year, order_total)

Set of functional dependencies, F:

$F = \{Order\_ID \rightarrow Order\_ID, username, tracking\_num, province, city, address, postal\_code, month, year, order\_total$
$tracking\_num \rightarrow Order\_ID$//Each order has a unique tracking number
$postal\_code \rightarrow province, city\}$*//See publisher schema*

The first functional dependency has the relations primary key on the left-hand side, so it does not violate BCNF. The second dependency determines a super key, making it a super key as well. Thus, none of the functional dependencies violate BCNF.

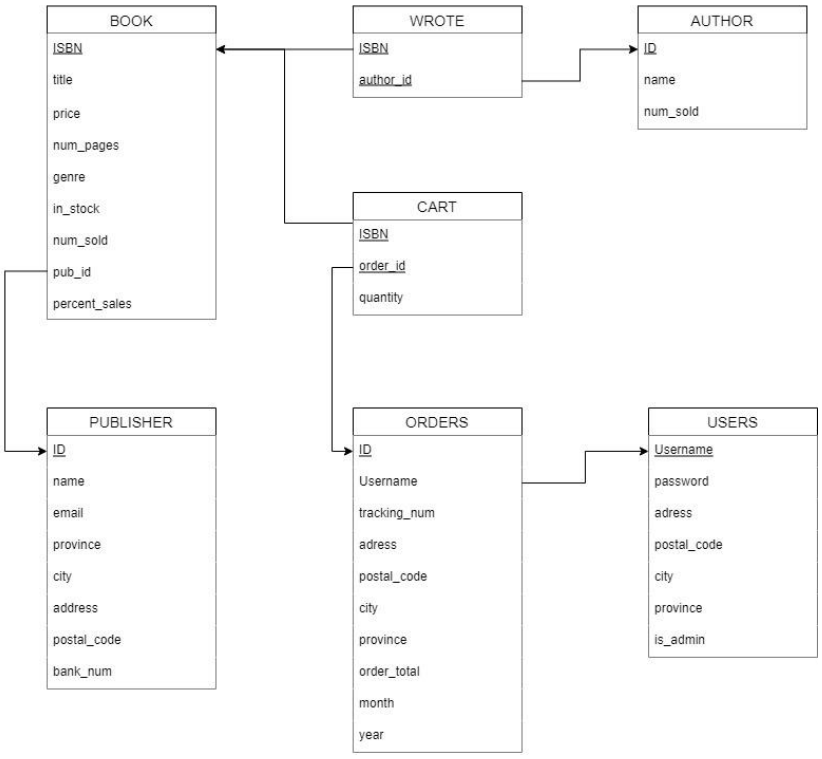
User Schema

user(Username, password , province, city, address, postal_code, is_admin)

Set of functional dependencies, F:

$F = \{Username \rightarrow username, password, province, city, address, postal\_code, is\_admin$
$postal\_code \rightarrow province, city\}$*//See publisher schema*
The only functional dependency present is the super key for this relation, meaning that this relation is already in BCNF.

Schema Diagram



Names of schemas and their attributes were slightly edited to make working with SQL easier.

Implementation

GitHub username: AhmadAlkfri

Available times to present: All day Monday after 10:00AM and before 7:30PM. I would also *like* to present my work (if I can volunteer).

The implementation of my database has many features, mostly on the user side. Without logging in, the user can:

- Search the database by book ISBN, title, author, or genre:

```
Welcome to LookInnaBook! Please select an option:
1. Search LookInnaBook
2. Log In
3. Register
1
1. Search by ISBN
2. Search by Title
3. Search by Author
4. Search by Genre
-1. Exit
```

When searching by title, the user can search broadly (matches any search term in any order) or narrowly (matches the search terms in order):

```
1. Search by ISBN
2. Search by Title
3. Search by Author
4. Search by Genre
-1. Exit
2
Please select: Broad Search (1) or Narrow Search (2):
1
Please enter the title of the book: Flies Of The
RESULTS:
----------
The Fellowship Of The Ring
Author(s):
    J. R. R. Tolkein
Price: $10.99
Genre: Fantasy
In Stock: 10

----------
Lord Of The Flies
Author(s):
    William Golding
Price: $14.50
Genre: Fiction
In Stock: 8

-------
```

- Log in:

```
Welcome to LookInnaBook! Please select an option:
1. Search LookInnaBook
2. Log In
3. Register
2
Please enter your Username:
user1
Please enter your Password:
user1
```

- Register:

```
Welcome to LookInnaBook! Please select an option:
1. Search LookInnaBook
2. Log In
3. Register
3
Please enter your Username (15 Char MAX):
newUser
Please enter your Password (15 Char MAX):
12345
Please enter your Province (2 Char format):
ON
Please enter your City:
Ottawa
Please enter your address:
111 Fake Rd
Please enter your postal code (no dashes or spaces):
K2J4S2
Success.
```

The inputs of any field are bulletproofed against invalid entries, and the code will not break if a invalid value is entered.

Once the user is signed in, they have access to ordering:

- Ordering starts by searching for a book. Searching is the same as before:

```
Welcome to LookInnaBook! Please select an option:
1. Search LookInnaBook
2. Log Out
4. Start an order
4
1. Search by ISBN
2. Search by Title
3. Search by Author
4. Search by Genre
-1. Exit
-2. Checkout
2
Please select: Broad Search (1) or Narrow Search (2):
2
Please enter the title of the book: Lord Of The
RESULTS:
0. Lord Of The Flies
Enter item # to add to cart (enter -1 to add nothing):
```

- After getting search results, the user picks one to add to cart, and choses the quantity (entering more books than in stock cancels the process). The user is immediately shown their cart as is again, and they can now look for a new book (or checkout):

```
RESULTS:
0. Lord Of The Flies
Enter item # to add to cart (enter -1 to add nothing):
0
Quantity to add to cart (8 in stock):
5
Your Cart:
5x - Lord Of The Flies
-----
1. Search by ISBN
2. Search by Title
3. Search by Author
4. Search by Genre
-1. Exit
-2. Checkout
```

- Checking out lets the user either use the data they have in their profile to order or enter their own:

```
Use user data to checkout (Y/N)?
Y
Your Cart:
5x - Lord Of The Flies
-----
Total Cost $14.5
Confirm this order? (Y)
```

- Confirming the order displays the order number (randomly generated and bulletproofed against duplicates) before returning to the main menu:

```
-----
Total Cost $14.5
Confirm this order? (Y)
Y
Order ID: 77928587
Welcome to LookInnaBook! Please select an option:
1. Search LookInnaBook
2. Log Out
4. Start an order
```

- Behind the scenes: User side

    - When the user makes a username, it must be unique otherwise the software will not let them use it. Province codes must also be valid. When searching by ISBN, the ISBN is validated before the user searches.
    - When the user checks out, the num_sold in the book and author schema is updated. The in_stock is decremented.

As for the admin side, there isn't much done there due to time constraints, however the admin can add authors, books and publisher's to the database. The admin is a user with admin privileges, so they can also use all user features. Future features would include viewing bestselling books/authors (the views for these are already created), removing books and publishers, viewing revenue, expenses, and profit.