Online Book Store

COMP421_Project Part1_Group 6
Yanhan Liu, 260797917
Hao Shu, 260776361
Yimeng Hu, 260795862
Zhiying Tan,260710889

1. Requirement Analysis

1.1. Introduction

1.1.1. Purpose

This application attempts to build a working online marketplace where all registered bookstores can sell books. The structure helps stores maintain inventory information and enables clients to add books to cart and saved lists, make a purchase and write comments for books.

1.1.2. Scope and special requirements

We limit the scope of our model to the online shopping platform itself strictly. The main focus of the application is the act of book purchase and maintenance of accurate inventory for each store. Thus, the modelling of some elements, including shipping & delivery and the physical management of inventory, are not taken into consideration.

1.1.3. Terminology

ISBN - the acronym for International Standard book number. This 10 or 13-digit number identifies a specific book, an edition of a book, or a book-like product (such as an audiobook). Since 1970 each published book has a unique ISBN. (The reason why we do not choose it as the primary key of the entity book will be indicated in the section of data requirement).

1.1.4. Resources

- *ISBN number definition*
  https://en.wikipedia.org/wiki/International_Standard_Book_Number

1.2. Data Requirements

1.2.1. Entities and Attributes

The following entities will be stored in the tables of a relational database.

CUSTOMER*:* Customers are identified by their unique email address. A customer is associated with several other elements including name, address, and phone number as their attributes.

MEMBER*:* Member is considered as a registered customer who possesses the attributes of Customer in addition to two more attributes including username and password. A member is also uniquely identified with email.

WAREHOUSE: Warehouse is where stores stock up their products. It is uniquely identified with a warehouse ID. Moreover, it has three more attributes including name, address and phone

Store: This entity set is to keep track of all the store information. Each store can be uniquely identified with a sid number. In addition the name, contact information and address are also noted.

BILL: Bill is to keep track of different orders made by the customer. Each customer can have multiple bills but one bill can only belong to one customer. They are uniquely defined by a unique payment_ID. It also stores the payment type of this order and the final amount paid.

CART: Each customer has their own cart identified by a Cart_id. In addition, it also stores the total cost of the items stored in a given cart.

SAVED LIST: Its relationship is the same as the one of the entity Cart. Its function is to store the books that a customer wishes to have but not yet decided to buy.

BOOKS: Books are the product of stores. It is a weak entity of Store. Though every book has its own unique ISBN, multiple stores can be selling the same book. In order to uniquely identify a book, each store gives its product a unique ID within the store called InStore_BookID which act as a partial key of the weak entity Books.

REVIEWS: A book can have many reviews and they are listed by the first-come-first-serve principle. It is a weak entity of Books. A review can only be uniquely identified with Book_id together with its partial key review_number.

1.2.2. Relationships

ISA: Member is a subclass of Customer. It is a non-covering ISA hierarchy relationship

Belongs: This is a one-to-one relationship. Each Customer can have at most one cart and each cart can belong to at most one customer.

Own: This is a one-to-one relationship. Each customer can have at most one saved list and each saved list can belong to at most one customer.

Move_to: An item saved in saved list can be moved cart if a customer decides to make the purchase. Vice versa, an item stored in cart can be moved to saved list if a customer changed their mind and decide to buy later. This is a one-to-one relationship since a customer can make changes to their unique saved list and unique cart.

Payment: This is a ternary relationship between Customer, Cart and the Bill. A customer make many payments but each time a cart can only issue one payment. Similarly with Bill, each payment made can only have one Bill.

Stock: This is a many to many relationships between the entities Store and Warehouse. There is no constraint on this relationship. A store can have or not have a warehouse to stock up their store or it can has many warehouses as their supplier, vice versa.

Contain: This is a many-to-many relationship between the saved list and book. A saved list can contain many books and a book can be saved in multiple saved lists.

Added: This is a many-to-many relationship between the cart and book. A cart can have many books and a book can be added in multiple cart.

Have: This is a one-to-many relationship since Book is a weak entity of Stores. Each book can only be uniquely identified with sid together with its partial key Instore_bookid.

Examine: This is a one-to-many relationship between Books and Review since Review is a weak entity of Books. Each review is assigned a review number which can be repetitive if its corresponding book is different. Hence it can only be uniquely identified together with the InStore_BookIn and Sid.

1.3. Application description

1.3.1. Overview

This application seeks to efficiently keep track of the books that a customer wants to add to cart or saved list, check if the book is in stock, and determine how much the customer needs

to pay for the order. Calculations are performed on the cart items and the payment method selected to determine the bill.

1.3.2. Preliminary calculations

Once the customer adds books to the cart, the algorithm detects the quantity of each book in the cart. The quantities are stored for determining the bill.

1.3.3. Algorithm description

By summing up the product of the price and the quantity of every book in the cart, the total cost of the books is calculated. Then the algorithm will detect the payment method selected and add the possible handling fee to the order. Finally, the tax is added to the order and the final bill will be displayed to the customer.

2. Relations

2.1. Entities

- *Customer(email, name, address, phone_number)*

- *Member(username, password, name,address,email,phone_number)*
  ---(email ref Customer)
- *SavedList(savedlist_id, email)*
  ---(email ref Customer)
- *Store(sid, name, location)*

- *cart(total_cost, cartID, savedListID)*
  ---(savedListID ref SavedList)

- *Bill(payment_id, payment type, final_amount_paid,email,cartID)*
  ---(email ref Customer)
  ---(cartID ref Cart)
- *Warehouse(warehouse_id, address, phone)*

2.2. Weak Entities

- *Books(sid, instore_bookID, ISBN, category, price, author, publisher, stock_status)*
  ---(sid ref Store)
- *Review(sid, instore_bookID, review, number, rate, comment)*
  ---(sid,instore_bookID ref Books)

2.3. Relationships

- *added(quantity, cartID, sid, inStore_BookID)*
  ---(cartID ref Cart)
  ---(sid,inStore_BookID ref Books)
- *contain(savedListID, inStore_BookID, sid)*
  ---(savedListID ref SavedList)
  ---(sid,inStore_BookID ref Books)

- *stock(sid, warehouse_id)*
  ---(sid ref Store)
  ---(warehouse_id ref Warehouse)

Other constraints:

Unable to stop the customer from purchasing when out of stock

Books amount reduction

Reference:

https://www.scribd.com/presentation/252870451/Entity-Relations-Model