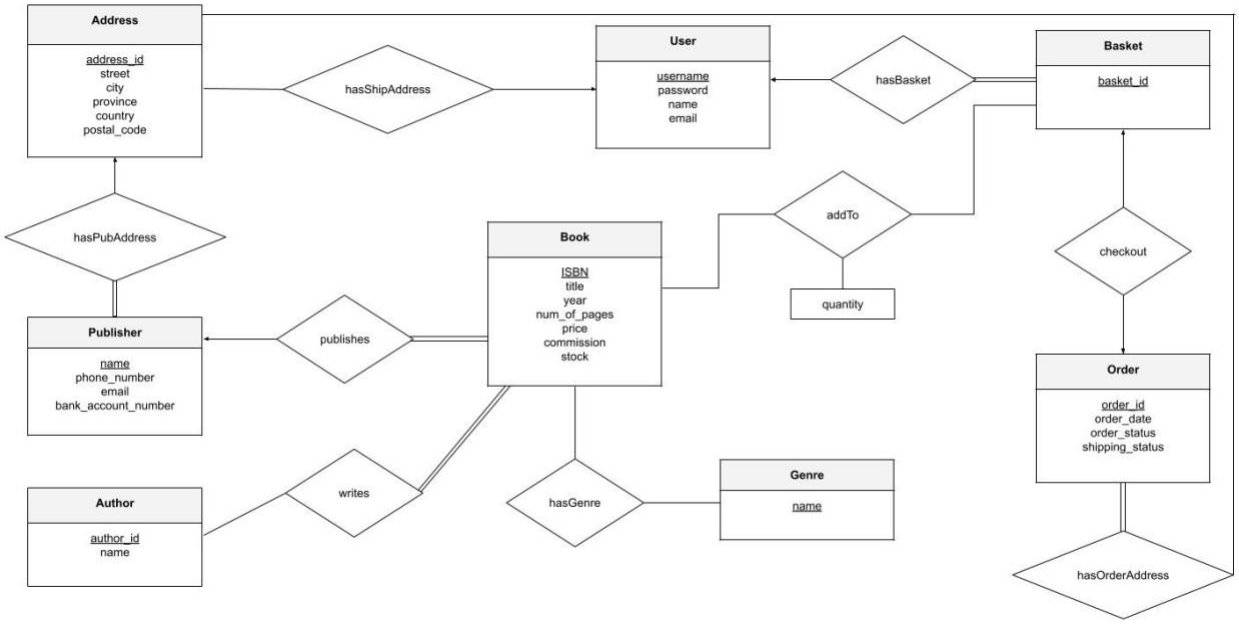
Look Inna Book: Final Project

Isabella Joao: #101149867
COMP3005
Ahmed El-Roby and Abdelghny Orogat
December 11, 2022

2.1 Conceptual Design



Book
The “Book” entity indicates there is a book in the bookstore. The primary key is the ISBN because no two books can have the same ISBN. The attributes are ISBN, title, year, number of pages, price, commission, and stock.

Genre
The “Genre” entity indicates the genre of each book. The primary key is the name because that is the only information about the genre that needs to be recorded. Name is also the only attribute.

User
The “User” entity indicates there is a user registered on the bookstore. The primary key is the username because no two users can have the same username. The attributes are username, password, name, and email.

Publisher
The “Publisher” entity indicates the publisher of a book. The primary key is the name because no two publishers can have the same name. The attributes are name, phone number, email, and bank account number.

Author
The “Author” entity indicates the authors of each book. The primary key is the author’s id because there can be multiple authors with the same name. So, designating an id for each author would be more optimal. The attributes are author id and name.

Address
The “Address” entity indicates an address. The primary key is the address’ id because multiple orders, for example, can share the same address. So, designating an address_id would be optimal for differentiating each occurrence. The attributes are address_id, street, city, province, country, and postal code.

Basket
The “Basket” entity indicates the user’s basket/cart of books. The primary key is the basket id because there will be a unique basket of books every time the user checks out. Basket id is also the only attribute.

Order
The “Order” entity indicates an order that has been placed. The primary key is the order number because no orders can have the same order number. The attributes are order id, order date, order status, and shipping status.

---

hasShipAddress
The “hasShipAddress” relation indicates an address with a user. The user’s address is supposed to be represented by the relation. The attributes for this relation are the address id and name. The primary key is the address id because each user can only have one address.

hasPubAddress
The “hasPubAddress” relation indicates a publisher with an address. The publisher's address is supposed to be represented by the relation. There can only be one address for each publisher. So, the attributes for this relation are address id and name. The primary key is name because there can be only one address associated with each publisher.

hasOrderAddress
The “hasOrderAddress” relation indicates an order to an address(s). The order’s address is supposed to be represented by the relation. The attributes for this relation are address id and order number. The primary keys are address id and order number because there can be more than address to an order and multiple orders can have the same address.

hasBasket
The “hasBasket” relation indicates a basket to a specific user. The user’s basket is supposed to be represented by the relation. The attributes for this relation are username and basket_id. The primary key is basket id because there a user can’t share a basket with another user, so it must be unique.

addTo
The “addTo” relation indicates a book to a specific basket. The basket’s book(s) are supposed to be represented by the relation. The attributes for this relation are ISBN, basket id and quantity.

The primary keys are ISBN and basket id because a basket can have a multitude of books and a multitude of baskets can have any books.

checkout
The “checkout” relation indicates an order from a basket has been ‘completed’. The attributes are order number and basket id. The primary keys are also order number and basket id because order number can only be associated to one basket and a basket can only be “checked out” once.

publishes
The “publishes” relation indicates a book to a specific publisher. How many of the publisher’s book(s) there is represents the relation. The attributes are IBSN and name. The primary keys are also IBSN and name because each book can only have one publisher.

writes
The “writes” relation indicates a book to a specific author. How many books the author writes represent the relation. The attributes are ISBN and name. The primary keys are also ISBN and name because each book can have more than one author and vise versa.

hasGenre
The “hasGenre” relation indicates a book to a specific genre. The book’s genre is supposed to be represented by this relation. The attributes are name and ISBN. The primary keys are also name and ISBN because there cannot be more than one genre for each book.

Publisher to Address
Publisher to Address has a many-to-one cardinality. The ‘one’ is indicated on the Address entity because the same address can be in relation to a multitude of publishers. However, only one address can be in relation to a publisher.

Book to Publisher
Book to Publisher has a many-to-one cardinality. The ‘one’ is indicated on the Publisher entity because a multitude of books can have the same publisher. However, a publisher can only publish a book once and another publisher cannot publish the same book.

Book to Author
Book to Author has a many-to-many cardinality. This is because an author can write a multitude of books and a book can have a multitude of authors.

Book to Genre
Book to Genre has a many-to-many cardinality. This is because a genre can apply to a multitude of books and a book can have a multitude of genres associate to it.

Basket to User

Basket to User has a many-to-one cardinality. The ‘one’ is indicated on the User entity because a basket can only be in relation to one user. However, a user can have a multitude of baskets.

Address to User
Address to User has a many-to-one cardinality. The ‘one’ is indicated on the Address entity because an address can’t have a multitude of users, but a user can have a multitude of addresses.

Book to Basket
Book to Basket has a many-to-many cardinality. This is because a basket can have a multitude of books and a book can exist in a multitude of baskets.

Order to Address
Order to Address has a many-to-many cardinality. This is because an order can be in relation to a multitude of address and an address can have a multitude of orders.

Order to Basket
Order to Basket has a one-to-one cardinality. This is because an order in a basket cannot be “checked out” more than once.

Total Participation
Book to writes needs every book to have a minimum of one author associated with it. Book to publishes needs every book to have one publisher associated with it. Basket to hasBasket needs every basket needs to have a user associated with it. Lastly, Publisher to hasPubAddress needs every publisher to have a shipping address.

Partial Participation
All other relations besides the relations above are partial relationships. This is because the other relations do not require their relations to participate.

2.2 Reduction to Relation Schemas

book (ISBN, title, year, num_of_pages, price, commission, stock)
user (username, password, name, email)
address (address_id, street, city, province, country, postal_code)
author (author_id, name)
publisher (name, phone_number, email, bank_account_number)
genre (name)
basket (basket_id)
order (order_id, order_date, order_status, shipping_status)
writes (ISBN, name)

publishes (ISBN, name)
hasGenre (ISBN, name)
hasOrderAddress (order_number, address_id)
hasShipAddress (address_id, name)
hasPubAddress (name, address_id)
addTo (basket_id, ISBN, quantity)
hasBasket (basket_id, username)
checkout (basket_id, order_number

2.3 Normalization of Relation Schemas

book (ISBN, title, year, num_of_pages, price, commission, stock)
F = { ISBN -> title, year, num_of_pages, price, commission, stock }
ISBN+ = {ISBN, title, year, num_of_pages, price, commission, stock}
ISBN is a superkey. BCNF is shown to be true.

user (username, password, name, email)
F = { username -> password, name, email }
username+ = {username password, name, email }
username is a superkey. BCNF is shown to be true.

address (address_id, street, city, province, country, postal_code)
F = { address_id -> street, city, province, country, postal_code }
address_id + = { address_id, street, city, province, country, postal_code }
address_id is a superkey. BCNF is shown to be true.

publisher (name, phone_number, email, bank_account_number)
F = { name -> email, bank_account_number
bank_account -> name
email -> name
phone_number -> name
}
name+ = { name, phone_number, email, bank_account_number }
phone_number, email, bank_account_number are candidate keys. BCNF is shown to be true.

author (author_id, name)
F = { author id -> name }
author_id += { author_id, name }
Author_id a superkey. BCNF is shown to be true.

publisher (name, phone_number, email, bank_account_number)
F =
{

name -> phone_number, email, bank_account_number
bank_account_number -> name
email -> name
phone_number -> name
}
name+ = {pub name, email, phone num, bank acc}
phone_number, email, bank_account_number are candidate keys. BCNF is shown to be true.

genre (name)
F = { name -> name }
name is a superkey. BCNF is shown to be true.

basket (basket id)
F = { basket_id -> basket_id }
basket_id is a superkey. BCNF is shown to be true.

order (order_id, order_date, order_status, shipping_status)
F =
{
order_id -> order_date, order_status, shipping_status
shipping_status -> order_id
}
order_id + = shipping_status + = { order_id, order_date, order_status, shipping_status }
order_id and shipping_status are candidate keys. BCNF is shown to be true.

writes (ISBN, name)
F =
{
ISBN -> ISBN
name -> name
}
(name) + = { name } and ISBN+ = { ISBN }
name and ISBN are not superkeys, so they must become a superkey (ISBN, name) and
BCNF will be true.

publishes (ISBN, name)
F =
{
ISBN -> ISBN
name -> name
}
(name) + = { name } and ISBN+ = { ISBN }
name and ISBN are not superkeys, so they must become a superkey (ISBN, name) and

BCNF will be true.

hasGenre (ISBN, name)
F =
{
ISBN -> ISBN
name -> name
}
(name) + = { name } and ISBN+ = { ISBN }
ISBN and name are superkeys. BCNF is shown to be true.

hasOrderAddress (order_number, address_id)
F =
{
order_number, -> order_number
address_id -> address_id
}
(order_number) + = { order_number } and address_id + = { address_id }
order_number and address_id are superkeys. BCNF is shown to be true.

hasShipAddress (address_id, name)
F = { address_id -> name }
address_id+ = { address_id, username }
address_id is a superkey. BCNF is shown to be true.

hasPubAddress (name, address_id)
F =
{
name -> address_id
address_id -> name
}
name + = address_id+ = { name, address_id}
name and address_id are candidate keys. BCNF is shown to be true.

addTo (basket_id, ISBN, quantity)
F = { basket_id, ISBN -> quantity }
(basket_id, ISBN) + = {basket_id, ISBN, quantity}
basket_id and ISBN are superkeys. BCNF is shown to be true.

hasBasket (basket_id, username)
F = { basket_id -> username }
basket_id+ = {basket_id, username}
basket_id is a superkey. BCNF is shown to be true.

checkout (basket_id, order_number)
F =
{
basket_id -> order_number
order_number -> basket_id
}
basket_id+ = order number + = {basket_id, order_number}
basket_id and order_number are candidate keys. BCNF is shown to be true.

2.4 Database Schema Diagram