COMP 3005 Project

Nicolas El-Khoury, Vinh Nguyen

April 14, 2020

1 Conceptual Design

1.1 Assumptions

Book:

- Each book's ISBN must be unique.
- Books may have the same title, author, and genre.
- The same book can be published by different publishers.

User:

- Each username must be unique.
- Users may have the same email, address, and phone number.
- Genders are restricted to $\{m, f, o\}$, representing 'Male', 'Female', and 'Other'.

Publisher:

- Publishers may have the same name, banking account, address, email, and phone number.

Order:

- Orders must have unique IDs and may only have one corresponding user.
- Orders may have the same date, tracking number, shipping company, billing and shipping address, and payment method.

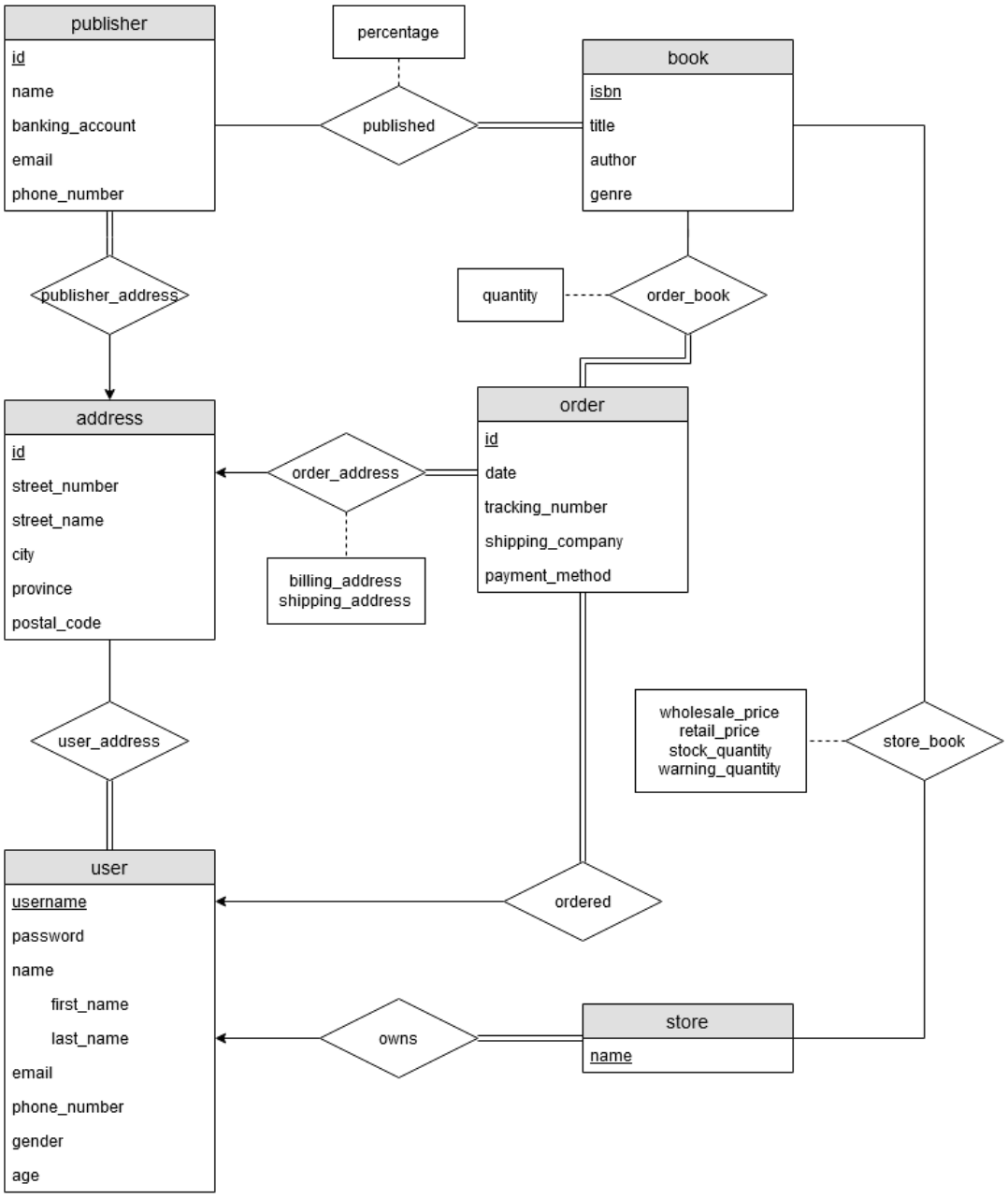
Address:

- Addresses may have the same street number, street name, city, province, and postal code.

Store:

- Each store only has one owner.
- Each store requires a corresponding user.
- Different stores may contain the same book(s).

1.2 ER Diagram



2 Reduction to Relation Schemas

- book(<u>isbn</u>, title, author, genre)
- publisher(<u>id</u>, publisher_name, banking_account, address_id, email, phone_number) *foreign key (address_id) references address(id)*
- published(<u>isbn</u>, <u>publisher_id</u>, pub_percentage)
- address(<u>id</u>, street_number, street_name, city, province, postal_code)

- order(<u>id</u>, date, tracking_number, shipping_company, billing_address, shipping_address, payment_method) *foreign key (billing_address) references address (id), foreign key (shipping_address) references address(id)*
- user(<u>username</u>, password, first_name, last_name, address_id, email, phone_number, gender, age) *foreign key (address_id) references address(id)*
- store(<u>store_name</u>, <u>username</u>) *foreign key (username) references user(username)*
- store_books(<u>store_name</u>, <u>isbn</u>, retail_price, wholesale_price, stock_quantity, warning_quantity) *foreign key (username) references user(username), foreign key (isbn) references book(isbn)*
- ordered(<u>order_id</u>, username) *foreign key (order_id) references order(id), foreign key (username) references user(username)*
- order_book(<u>order_id</u>, <u>isbn</u>, quantity) *foreign key (order_id) references order(id), foreign key (isbn) references book(isbn)*

3 Normalization of Relation Schemas

3.1 Book

isbn → title, author, genre, publisher_percentage
isbn → title, author, genre, publisher_percentage, isbn (Axiom of Augmentation)
$(\text{isbn})^+$ = {title, author, genre, publisher_percentagem, isbn}

Since the LHS of our functional dependency is our super key, and it is a non-trivial functional dependency, the book relation is currently in BCNF since there are no violations.

3.2 Publisher

publisher_id → publisher_name, banking_account, address, email, phone_number
publisher_id → publisher_name, banking_account, address, email, phone_number, publisher_id (Axiom of Augmentation)
$(\text{publisher\_id})^+$ = {publisher_name, banking_account, address, email, phone_number, publisher_id}
Since every LHS on the functional dependency is a candidate key, and all of the functional dependencies are non-trivial, the publisher relation is currently in BCNF since there are no violations.

3.3 Published

isbn, publisher_id → pub_percentage
isbn, publisher_id → pub_percentage, isbn, publisher_id (Axiom of Augmentation)
$(\text{isbn, publisher\_id})^+$ = {pub_percentage, isbn, publisher_id}

Since the LHS of our functional dependency is our super key, and it is a non-trivial functional dependency, the published relation is currently in BCNF since there are no violations.

3.4 Order

id → date, tracking_number, shipping_company, billing_address, shipping_address, payment_method
id → date, tracking_number, shipping_company, billing_address, shipping_address, payment_method, id (Axiom of Augmentation)
$(\text{id})^+$ = {date, tracking_number, shipping_company, billing_address, shipping_address, payment_method, id}

Since the LHS of our functional dependency is our super key, and it is a non-trivial functional dependency, the order relation is currently in BCNF since there are no violations.

3.5 User

username → password, first_name, last_name, address_id, email, phone_number, gender, age
username → password, first_name, last_name, address_id, email, phone_number, gender, age, username (Axiom of Augmentation)
$(username)^+$ = {password, first_name, last_name, address_id, email, phone_number, gender, age, username}
Since the LHS of our functional dependency is our super key, and it is a non-trivial functional dependency, the user relation is currently in BCNF since there are no violations.

3.6 Store

username → store_name
store_name → username
username → store_name, username (Axiom of Augmentation)
store_name → username, store_name (Axiom of Augmentation)
$(username)^+$ = {store_name, username}
$(store\_name)^+$ = {username, store_name}

Since the LHS of our functional dependencies are super keys, and they are non-trivial functional dependencies, the store relation is currently in BCNF since there are no violations.

3.7 Store_books

store_name, isbn → retail_price, wholesale_price, stock_quantity, warning_quantity
store_name, isbn → retail_price, wholesale_price, stock_quantity, warning_quantity, store_name, isbn (Axiom of Augmentation)
$(store\_name, isbn)^+$ = {retail_price, wholesale_price, stock_quantity, warning_quantity, store_name, isbn}

Since the LHS of our functional dependency is our super key, and it is a non-trivial functional dependency, the store_books relation is currently in BCNF since there are no violations.

3.8 Ordered

order_id → username
order_id → username, order_id (Axiom of Augmentation)
$(order\_id)^+$ = {username, order_id}

Since the LHS of our functional dependency is our super key, and it is a non-trivial functional dependency, the ordered relation is currently in BCNF since there are no violations.

3.9 Order_book

order_id, isbn → quantity
order_id, isbn → quantity, order_id, isbn (Axiom of Augmentation)
$(order\_id, isbn)^+$ = {order_id, isbn, quantity}

Since the LHS of our functional dependency is our super key, and it is a non-trivial functional dependency, the order_book relation is currently in BCNF since there are no violations.

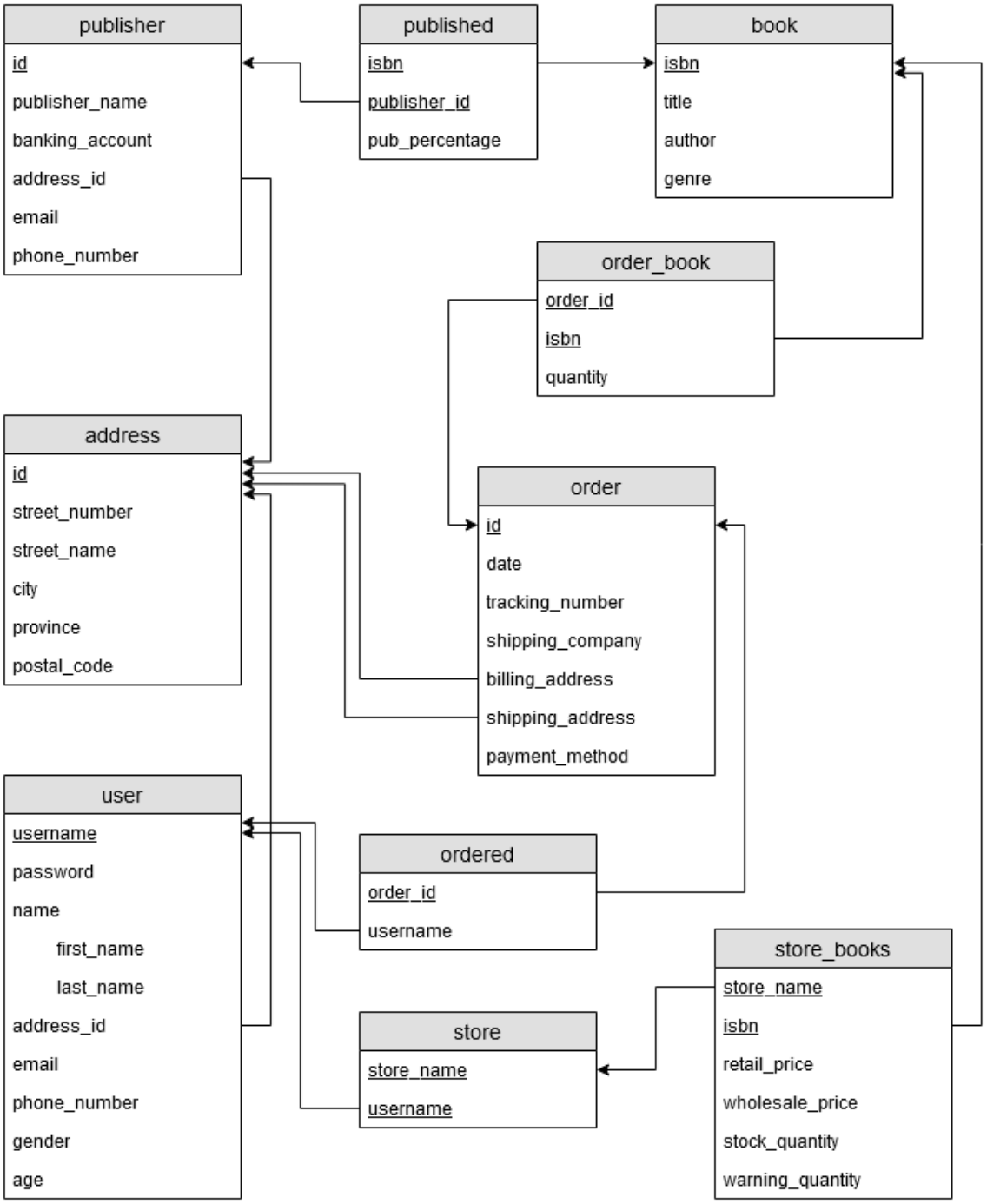
3.10 Address

id → street_number, street_name, city, province, postal_code
id → street_number, street_name, city, province, postal_code, id (Axiom of Augmentation)

postal_code → city, province (Axiom of Transitivity)
postal_code → city, province, postal_code (Axiom of Augmentation)
$(\text{id})^+$ = {street_number, street_name, city, province, postal_code, id} $(\text{postal\_code})^+$ = {city, province, postal_code}

As we can see, the postal_code functional dependency violates BCNF. If we decompose address into R_1 = (id, postal_code, street_number, street_name), R_2 = (postal_code, city, province), we can maintain BCNF in this decomposed relation.

4 Database Schema Diagram



5 Implementation

5.1 Backend by Nicolas

Implemented the back-end in NodeJS using the express, pg, and uuidv4 node modules. Creates queries dynamically in order to serve the user the correct information. Acts as a layer between the client and the SQL database that enables interaction in a safe and competent manner. Implements SQL injection protection to ensure users cannot manipulate SQL database in unwanted fashion. Authenticates users on a secure basis, utilizing uuidv4 to generate random authentication keys for each user. They are then verified through routes that utilize SQL queries to the database to ensure the client has the proper authentication levels to access the content they want.

5.2 Frontend by Vinh

The frontend is implemented using Node.js with Express, Pug, Axios, and Session modules. Sets up various routes for clients through Express. Serves templated pages to users by using the Pug template engine, based on criteria such as route, authentication, etc. Axios allows the frontend server to communicate with our backend by making XMLHttpRequests serverside to obtain information such as books, users, purchases, management, etc. The session module helps us to implement authentication with users as well as login and logout features for our website.

5.3 UI Screenshots

UI screenshots can be found within our github repository's README, found in section 7.

6 Bonus Features

- Separate backend and frontend servers, giving us modularity for additional projects and reuse in the future.
- Backend server is built as a stateless API, allowing other services and servers to make requests and interact with our database.
- Backend dynamically creates query based on given parameters, allowing for reuse with different databases, regardless of structure.
- Backend queries use both partial and exact matching for specific cases, allowing for flexibility in our search system.
- Frontend is built using Bootstrap for both web and mobile responsiveness, shown in UI screenshots.
- Frontend pages differ based on authentication level of client (ex: logged out, logged in, manager).
- Frontend has hidden account pages tailored to each user, using template engines and sessions.

7 Github Repository

Our project is hosted at the following public repository

`https://github.com/vinhvn/online-bookstore`