Database schema report

The database schema is outlined in Figure 1 below.

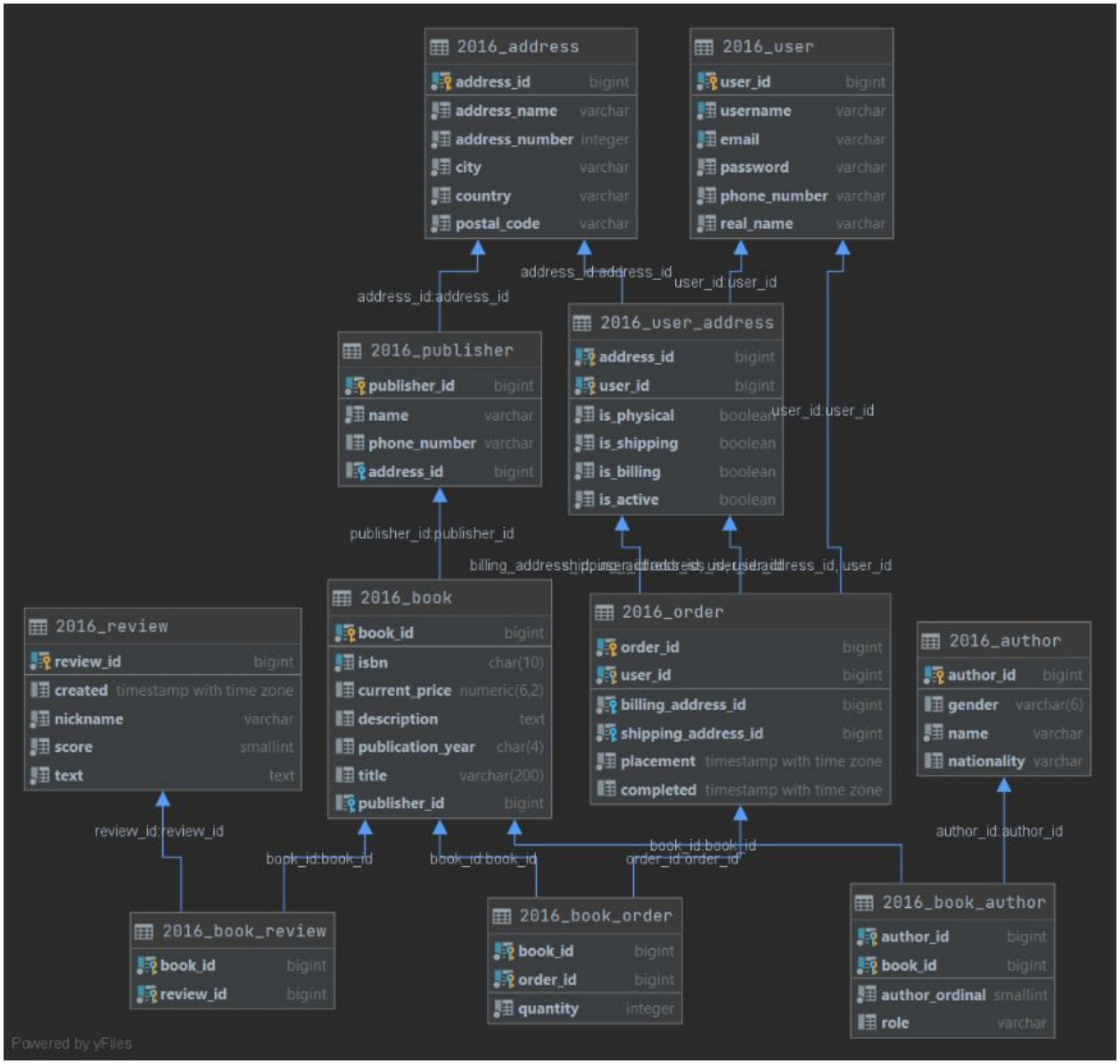

Figure 1: Database schema

Design practises and overview:

The schema consists of 11 tables. All of them are prefixed by 2016 (part of my academic ID), so I will omit it for the sake of simplicity.
For all the tables' primary keys, *big serial* data type was chosen in order to secure scalability, since the e shop could be successful in the future and acquire a lot of data.
Since all pks are auto incremented, for all the foreign keys, no update and delete strategy was followed, except for two keys.
For all date values timestamp with timezone (which defaults to utc) was used since we would want uniformity while storing data. The format should be handled by the code.
Postal codes can contain special characters in some countries so varchar is used, same applies to phone numbers.
Finally, if not restricted by the requirements, varchar fields may contain an arbitrary number of characters.
Many design choices were also affected by seeing the data.

Table book

attributes:
book_id : big serial -> primary key
isbn: char(10) -> unique not null
current_price: numeric(6,2) -> allow null
description: text -> allow null
publication_year: char(4) -> allow null
title: varchar(200) -> allow null
publisher_id: big int -> FK to table publisher(publisher_id) it can be null (on delete null)

Comments:
Since a comic book can be solely identified by its isbn, all other fields may take null values which can be later filled manually. Also eshop code could easily handle null values, by not displaying books for which data does not suffice.
String was chosen since isbns may contain check digits that are not numeric such as 'X'. The requirements specify that isbn should be of 10 digits, hence the 10 char restriction. Notice that since 2007 all new books come with an isbn consisting of 13 digits. So in reality, I would actually discuss with the customer to include an isbn and an isbn_13 in the database, since there are ways to transform isbn to isbn 13 and vice versa.
Price may be null, but the eshop admins will be responsible for adding the price when the eshop functions properly. Finally, a 6 precision number was chosen xxxx.xx, as it was assumed that a comic book cannot cost more than 9999.99 euros. It would be discussed with the customer
Text is used since description can be an arbitrary text of a couple of paragraphs.
The oldest comic book dates back to 1775, so char 4 was selected for the pub year.
Title is restricted to 200 characters by the requirements
If a publisher is deleted, we would not want to also remove the book from the database, hence on delete null strategy is specified.

Table publisher

attributes:
publisher_id : big serial -> primary key
name: varchar -> not null
phone_number: varchar -> allow null
address_id: big int -> FK to table address(address_id) it can be null (on delete null)

Comments:
Except for the name, all other values may be null and filled afterwards.
If an address is deleted, we would not want to also remove the publisher from the database, hence on delete null strategy is specified.

Table author

attributes:
author_id : big serial -> primary key
gender: varchar(6) -> allow null
name: varchar -> not null
nationality: varchar -> allow null

Comments:
Except for the name, all other values may be null and filled afterwards.
It is assumed that gender can take “female”, “male”, “other” choices, hence the 6 character restriction. Since data around gender are quite sensitive, no default value was chosen.

Table book_author

attributes:
author_id : big int -> FK to table author(author_id), primary key
book_id: big int -> FK to table book(book_id), primary key
role: varchar -> allow null
ordinal: smallint -> not null, default 0

Comments:
Represents the many to many relationship between books and authors.
Primary key is a composite key (author_id, book_id).
Author role was put in this table, since authors can have different roles for different books.
Ordinal indicates the order of the book author, so if a book has 5 authors it will take values from 1 to 5. Defaults to 0.

Table address

attributes:
address_id : big serial -> primary key
address_name: varchar -> not null
address_number: int -> not null
city: varchar -> not null
country: varchar -> not null
postal_code: varchar -> not null

Comments:
For addresses a separate entry was chosen, since many people can have the same address. Also the publisher address can be included here. Since we need all the fields above in order to specify an address, they cannot be null.

Table review

attributes:
review_id : big serial -> primary key
created: timestamp with time zone -> not null
nickname: varchar -> not null, defaults to 'anonymous'
score: smallint -> not null
text: text -> not null

Comments:
Since reviews can be anonymous, the 'anonymous' value was chosen as default for nickname. The score is restricted from 1 to 5 and should be checked by the code before inserted, in order to avoid db overhead.

Table book_review

attributes:
book_id: big int -> FK to table book(book_id), primary key
review_id: big int -> FK to table review(review_id), primary key

Comments:
Represents the many to many relationship between books and reviews.
Primary key is a composite key (book_id, review_id).

Table user

attributes:
user_id : big serial -> primary key
username: varchar -> not null, unique
email: varchar -> not null, unique
password: varchar -> not null
phone_number: varchar -> not null
real_code: varchar -> not null

Comments:
The email and password validation and encryption should be handled by the code. The database stores them as varchars.

Table user_address

attributes:
address_id: big int -> FK to table address(address_id), primary key
user_id: big int -> FK to table user(user_id), primary key
is_physical: bool -> not null defaults to true
is_shipping: bool -> not null defaults to true
is_billing: bool -> not null defaults to true
is_active: bool -> not null defaults to true

Comments:
Represents the many to many relationship between addresses and users.
Primary key is a composite key (address_id, user_id). Contains helping fields as a user can have many addresses, some which may be active or inactive. Moreover, we would need a shipping address and a billing address for orders.

Table order

attributes:
order_id: big serial -> primary key
user_id: big int -> FK to table user(user_id), primary key
physical_address_id: bool -> FK to table address(address_id) not null
shipping_address_id: bool -> FK to table address(address_id) not null
placement: timestamp with timezone -> not null
completed: timestamp with timezone -> allow null

Comments:
Primary key is a composite key (order_id, user_id)
Foreign keys are also composite keys (user_id, physical_address_id) and (user_id, shipping_address_id) since an order will typically require both a user's shipping and a billing address. However they can also be the same address.
Order placement is not allowed to be null, completed field should be filled once the order is completed

Table book order

attributes:
book_id: big int -> FK to table book(book_id), primary key
order_id: big int -> FK to table order(order_id), primary key
quantity: int -> not null, defaults to 1

Comments:
Represents the many to many relationship between books and orders.
Primary key is a composite key (book_id, order_id). Also there is a quantity field several copies of a book may be purchased by an order.

Using the schema:

The schema resides in the *sql* directory and provided that there is an active database, it can be created by the following command:

psql -f schema.sql <database_name>

The database name I used was comic_books.