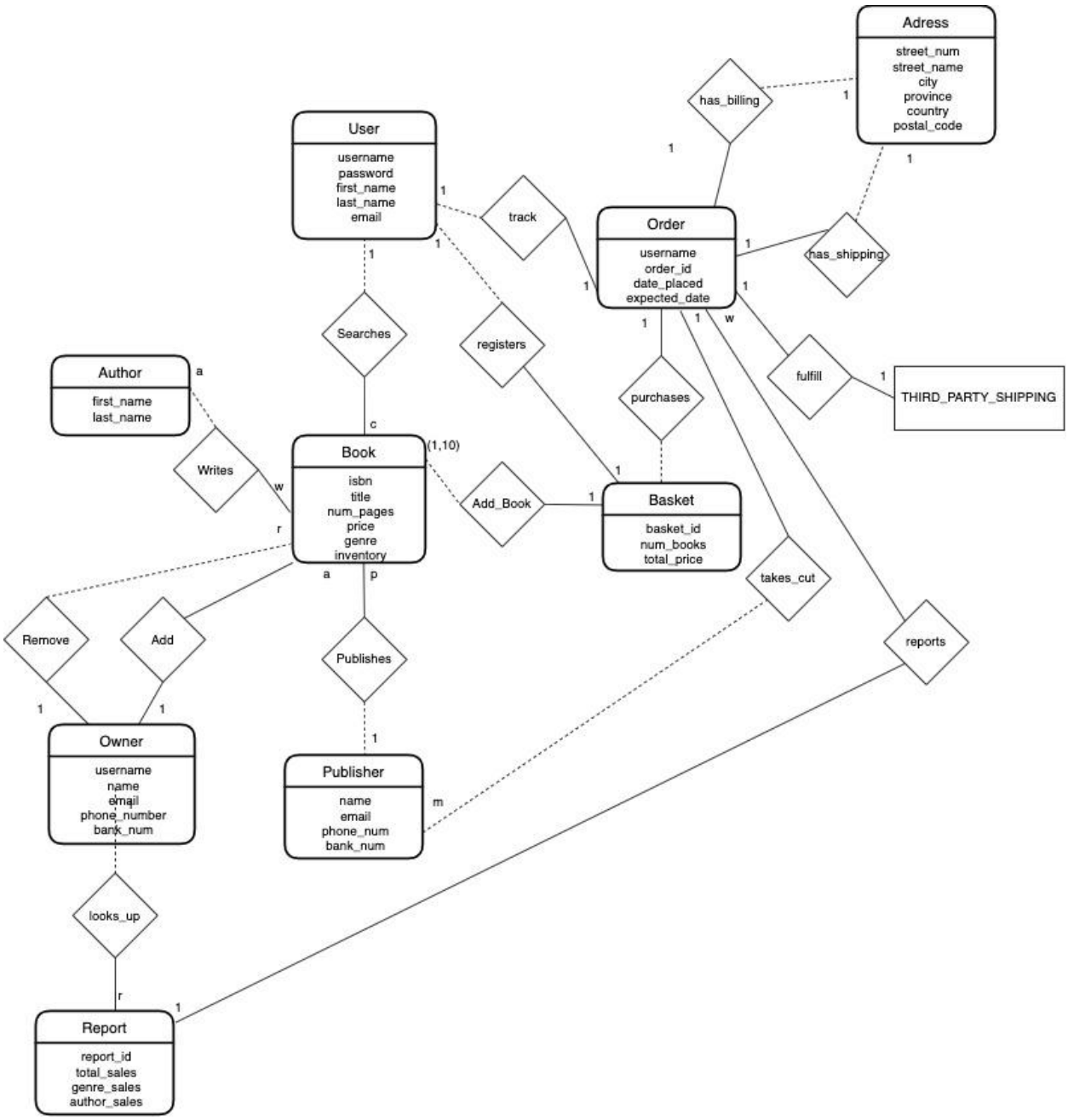
COMP 3005 COURSE PROJECT

Jacob Boruszkowski(101201450)
Ali Jameel (101185339)
Clint Garvez (101196386)

December 11, 2022

Link to GitHub: https://github.com/ClintGalvez/Bookstore

SECTION 2.1



Starting from the User entity, we have a relationship for searching books, for cardinalities we have 1 user, who can search for a c amount of books, as the user can add as many books as he or she likes. User and Searches has a partial participation as all users do not have to search, whereas Searches and Book has a full participation as every book must be searched through to find the proper book.

The User can also track a specific Order, there is a partial participation between User and track as not all users must take place in tracking an order, whereas there is a full participation between track and Order as all Orders must be tracked. User and Order share a one to one relationship as one User tracks one Order and an Order is tracked by one User.

User also shares a registers relationship with Basket as once the user gets to the basket, it will check if they are registered and that will determine if they can make an Order. User shares a partial participation with registers as not all users are registered, Basket shares a full participation with registers as all baskets will check if a user is registered. User and Basket share a one to one relationship and one user is checked if registered to that one users basket.

Next we move on to the Book entity, we have a relationship with basket through the relationship Add_Book, which adds a book to the basket. Book shares a partial participation with Add_Book as not every book has to be added to a basket, whereas basket has full participation to Add_Book as each book needs to correspond to a basket in order to be added. 1-10 books can be added to a basket, and 1 basket can have 1-10 books.

Book has a relationship with Author through the Writes relationship, Book and Writes share full participation as every book has to be written by an author, Author and Writes share a partial participation as not every book has to be written by a specific author. Author and Book share a many to many relationship as an author can write more than one book and books can be written by many authors.

Book has two relationships with Owner as an Owner can add or remove books. Book shares a partial participation with Remove as not every book has to be removed by an Owner, whereas Book shares a full participation with add as each book must be added by an Owner. Book shares a many to one relationship with Owner as many books can be added by one owner.

Book has a relationship with Publisher as a Publisher Publishes books. Book shares a full participation with Publishes as all books must be published. Publisher shares a partial participation with publishes because not every publisher has to publish a book. Book and Publisher share a many to one relationship as one publisher can publish more than one book.

Next, the Owner entity, the Owner entity has a looks_up relationship with Report as the owner must be able to look up the report for information. Owner shares a partial

participation with looks_up as not all owners look up report information, whereas looks_up and Report share a full participation as all reports are looked up by some owners. Owner and Report share a one to many relationship and one owner can look up many reports.

Next, the Report entity shares a reports relationship with Order as Report receives information from Order. Report shares a full participation with reports as each Report is reported to from Order. Order shares a full participation with reports as each Order reports back to Report. Report and Order share a one to many relationship as multiple Orders can be contained in a Report.

Next, the Publisher entity shares a takes_cut relationship with Order as the Publisher takes a cut (percentage of money) from the Order (based on which books are found in the order). Publisher and takes_cut shares a partial participation as not all publishers take a cut from a specific order. Order and takes_cut share a full participation as all orders have to allow a publisher to take a cut from them. Publisher and Order here form a many to one relationship as many Publishers can take a cuts from one Order.

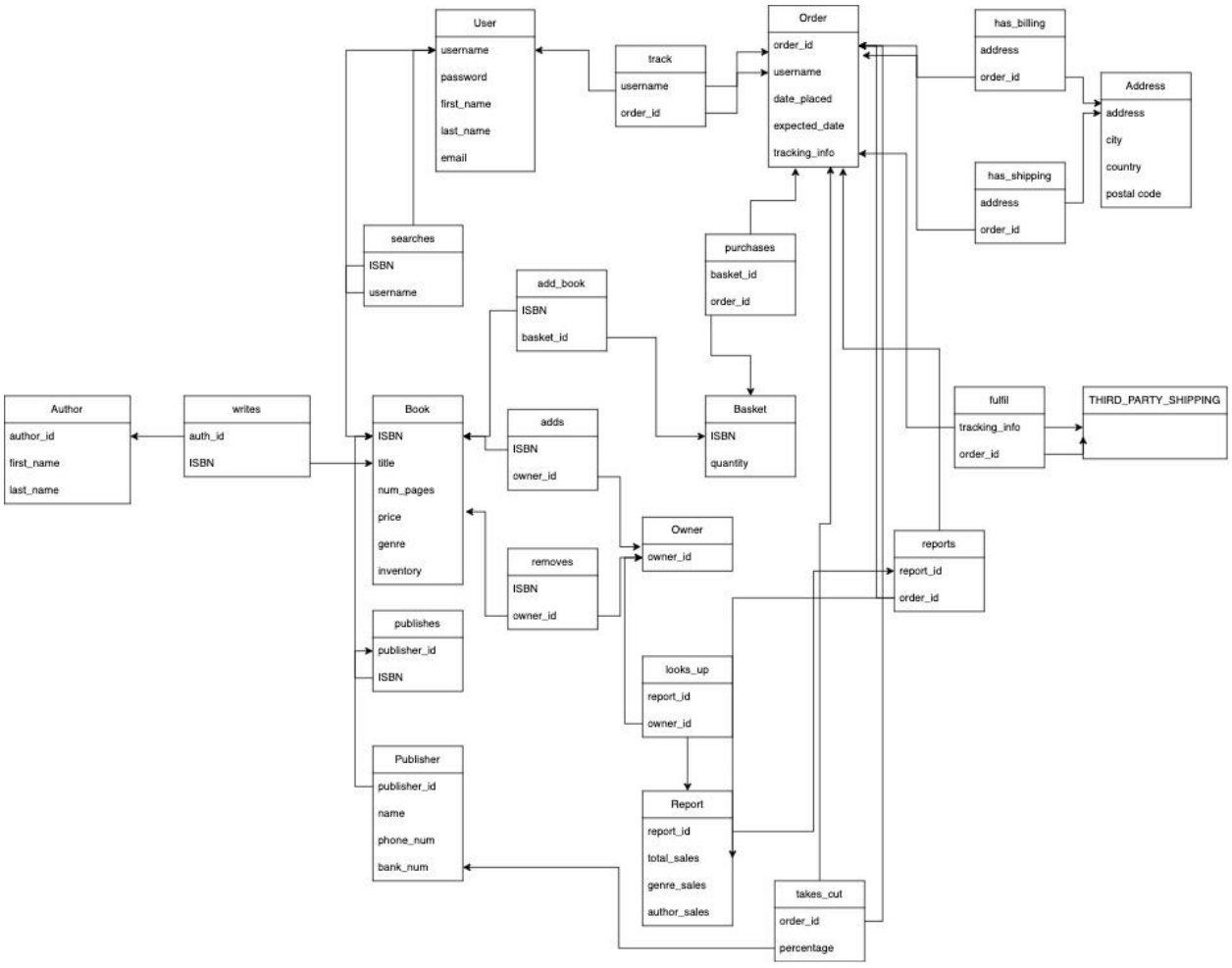
Next we have Basket, Basket shares a purchases relationship with Order as once a basket is made and completed (registration verified and all) then it must be purchased to officially create an Order. Basket and purchases shares a partial participation as not every Basket has to be purchased, whereas Order shares a full participation with purchases because each Order must be purchased. Order and Basket here share a one to one relationship as one Basket is purchased to create one order.

Next we move on to Order, Owner shares two relationships with address, has_billing and has_shipping, this is used to get the shipping and billing addresses to a specific Order. Order shares a full participation with has_billing and has_shipping as each order must have billing and shipping info. Address shares a partial participation with has_billing and has_shipping as not every address is part of an Order. Through has_billing and has_shipping Order shares a one to one relationship with Address as one Order has one shipping and billing Address (assuming an Order is only allowed to be sent to one Address not multiple, which is realistic to how orders work on any web store today).

Finally, Order has a fulfill relationship with THIRD_PARTY_SHIPPING as the order must be fulfilled by a third party shipping company. Order shares full participation with fulfill as each order must be fulfilled by a third party shipping company. THIRD_PARTY_SHIPPING shares a partial participation with fulfill as not all THIRD_PARTY_SHIPPING companies must partake in shipping a specific order. Here,

Order and THIRD_PARTY_SHIPPING share a one to one relationship as one order is fulfilled by one THIRD_PARTY_SHIPPING company.

SECTION 2.2



SECTION 2.3

Normalization

To check if a non-trivial dependency a->*b* causes a violation of BCNF:

Compute a^+

Verify that it includes all attributes of *R*, that is, it is a superkey of *R*

We must normalize all entities, to do so, ensure that they are all in BCNF (good form).

For the Book relation:

ISBN->title, num_pages, price, genre, inventory

We see that, for our only functional dependency, all attributes are dependent on ISBN, thus ISBN is a superkey of R, and none of the dependencies cause a failure of BCNF, therefore it is in BCNF (good form).

For the User relation:

username->password, first_name, last_name, email

We see that, for our only functional dependency, all attributes are dependent on username, this username is a superkey of R, and none of the dependencies cause a failure of BCNF, therefore it is in BCNF (good form).

For the Author relation:

author_id->first_name, last_name

We see that, for our only functional dependency, all attributes are dependent on a unique author_id, thus author_id is a superkey of R, and none of the dependencies cause a failure of BCNF, therefore it is in BCNF (good form).

For the Publisher relation:

publisher_id->name, email, bank_num

We see that, for our only functional dependency, all attributes are dependent on a unique publisher_id, thus publisher_id is a superkey of R, and none of the dependencies cause a failure of BCNF, therefore it is in BCNF (good form).

For the Owner relation:

owner _id->username, name, email, phone_number, bank_num

We see that, for our only functional dependency, all attributes are dependent on a unique owner_id, thus owner_id is a superkey of R, and none of the dependencies cause a failure of BCNF, therefore it is in BCNF (good form).

For the Basket relation:

basket_id->num_books, total_price

We see that, for our only functional dependency, all attributes are dependent on a unique basket_id, thus basket_id is a superkey of R, and none of the dependencies cause a failure of BCNF, therefore it is in BCNF (good form).

For the Order relation:

order_id->username, name, email, phone_number, bank_num

We see that, for our only functional dependency, all attributes are dependent on a unique order_id, thus order_id is a superkey of R, and none of the dependencies cause a failure of BCNF, therefore it is in BCNF (good form).

For the Address relation:

For this relation we added an address_id attribute which uniquely identifies addresses

address_id->street_num, street_name, city, province, country, postal_code

We see that, for our only functional dependency, all attributes are dependent on a unique address_id, thus address_id is a superkey of R, and none of the dependencies cause a failure of BCNF, therefore it is in BCNF (good form).

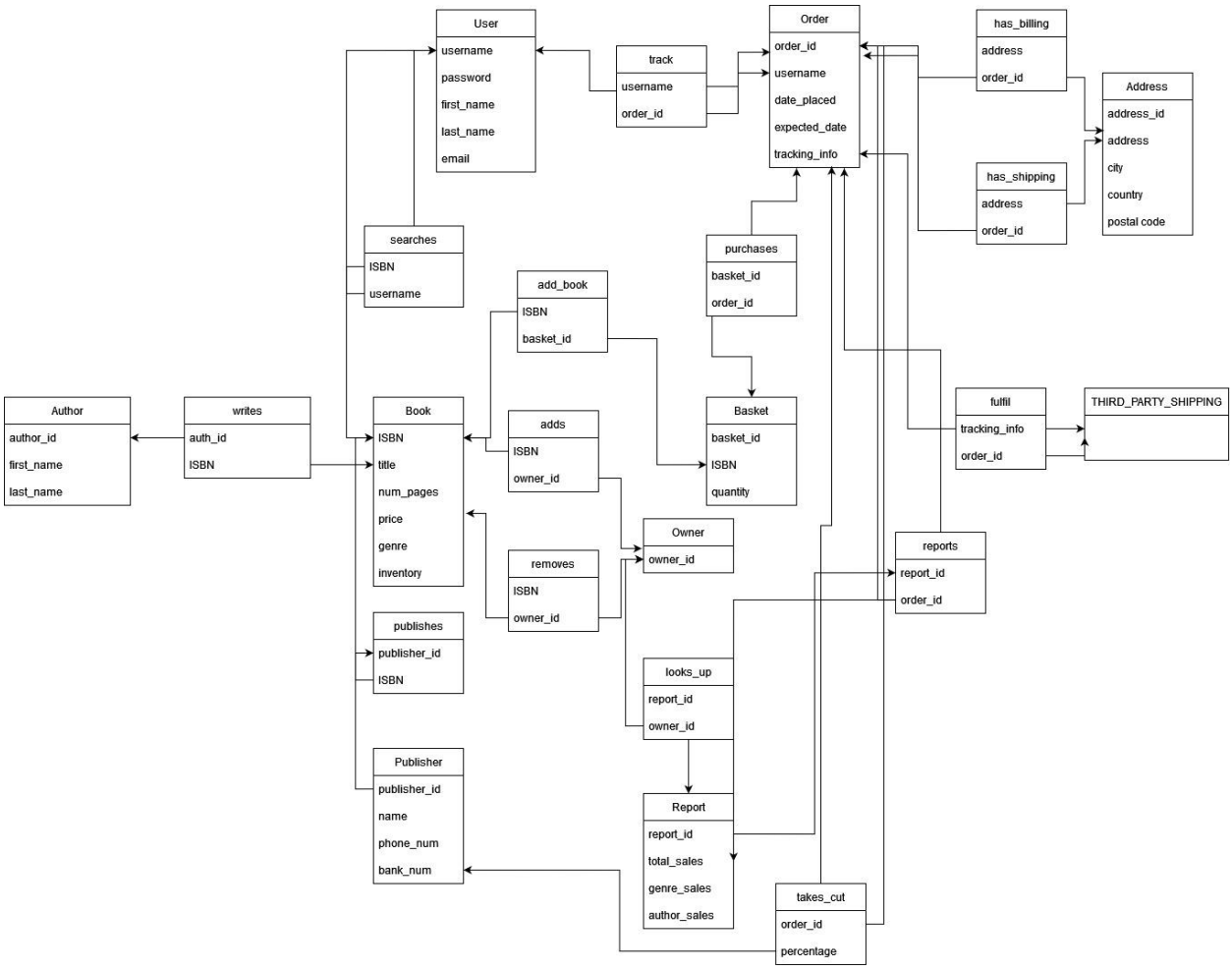
For the Report relation:

For this relation we added an report_id attribute which uniquely identifies report

report_id->total_sales, genre_sales, author_sales

We see that, for our only functional dependency, all attributes are dependent on a unique report_id, thus report_id is a superkey of R, and none of the dependencies cause a failure of BCNF, therefore it is in BCNF (good form).

SECTION 2.4



Not many changes for 2.4 as it was all in good form from 2.2, mainly minute attribute changes.

SECTION 2.5

Contents found in GitHub