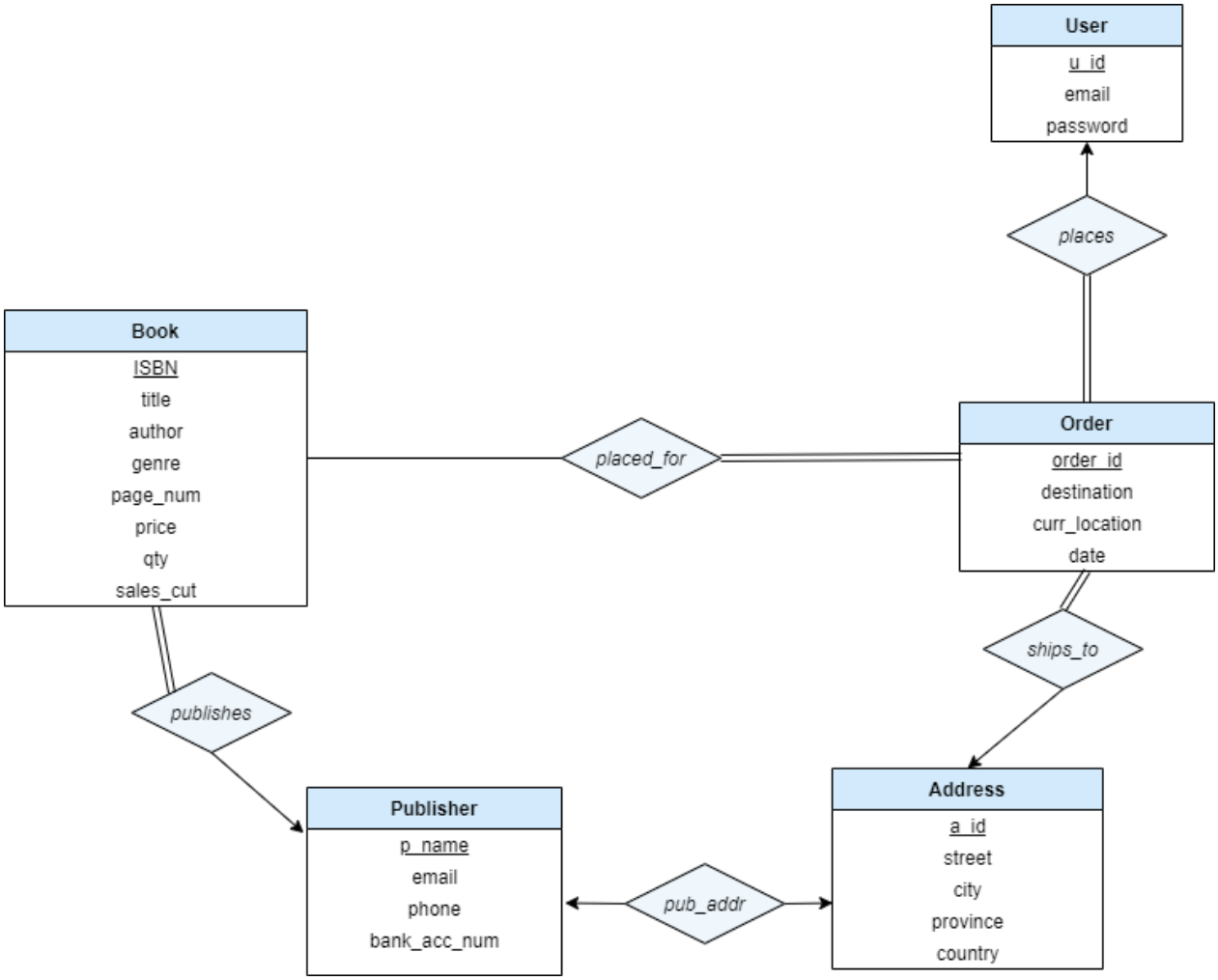
COMP 3005 Bookstore Database Design Project

Francis Bolton

April 14 2020

2. Project Report

2.1. Conceptual Design



Assumptions:

A specific order may only be placed by one user (i.e. an order id may not be associated with more than one user) and that an order must have been placed by a user for it to exist (total participation of *order* in *places*).

A publisher may only have one address, and likewise, a particular address may only be associated with one publisher.

For an order to exist it must contain books (total participation of *order* in $placed\_for$).

An order must ship to an address (total participation of *order* in *ships_to*). A particular order may only be sent to one address, but many orders may be shipped to a particular address.

An book may be published by only one company (i.e. publishers reserve exclusive publishing rights) and publishers may publish many books. A book must have a publisher (total participation of *book* in *publishes*).

It is also worth noting a few other design choices. I opted not to have a user register with an address, as the shipping address can be input on checkout. There is no need to store the user's shipping address. Users are uniquely identified by their user ID, which is interchangeable with username. A user need only register with an email address. Passwords are stored in plain text in the database (terrible practice, but hey).

I also assumed that in an actual implementation, sales reports would be generated by querying this database, and as such, no explicit information about sales is stored in the database itself. It is simply not needed. An example query to find monthly sales of a given book might be something like:

```
select ISBN, count(ISBN)
from placed_for
where date between '01-01-2019' and '01-31-2019'
```

Reports would be generated with these queries entirely.

2.2. Reduction to relation schemas

book(ISBN, title, author, genre, page_num, price, qty, sales_cut, p_name)
user(u_id, email, password)
order(order_id, destination, curr_location, date, u_id)
address(a_id, street, city, province, country)
publisher(p_name, a_id, email, phone, bank)
placed_for(order_id, destination, curr_location, date, ISBN, title, author, genre, page_num, price, qty, sales_cut)

Note: Because we have total participation in many-to-one relationships, we can add the primary key from the one-side to the many-side as an attribute. This has been done for the order and book relation schemas instead of having schemas for relationship sets involving those entities (i.e. publishes, placed_for, pub_addr, places, ships_to). The same has been done for the one-to-one relationship between publisher and address, with the publisher acting as the many-side.

2.3. Normalization

We will aim to have each relation schema in BCNF. We look at each schema and their functional dependencies to determine if one of them violates the rules for BCNF.

We can find F for each schema in section 2.2:
Book:
$ISBN \rightarrow title, author, genre, page\_num, price, qty, sales\_cut, p\_name$

User:
$u\_id \rightarrow password$

Order:
$order\_id \rightarrow destination, curr\_location, date, ISBN, u\_id, a\_id$

Address:
$a\_id \rightarrow street, city, province, country$

Publisher:
$p\_name \rightarrow a\_id, email, phone, bank\_acc\_num$

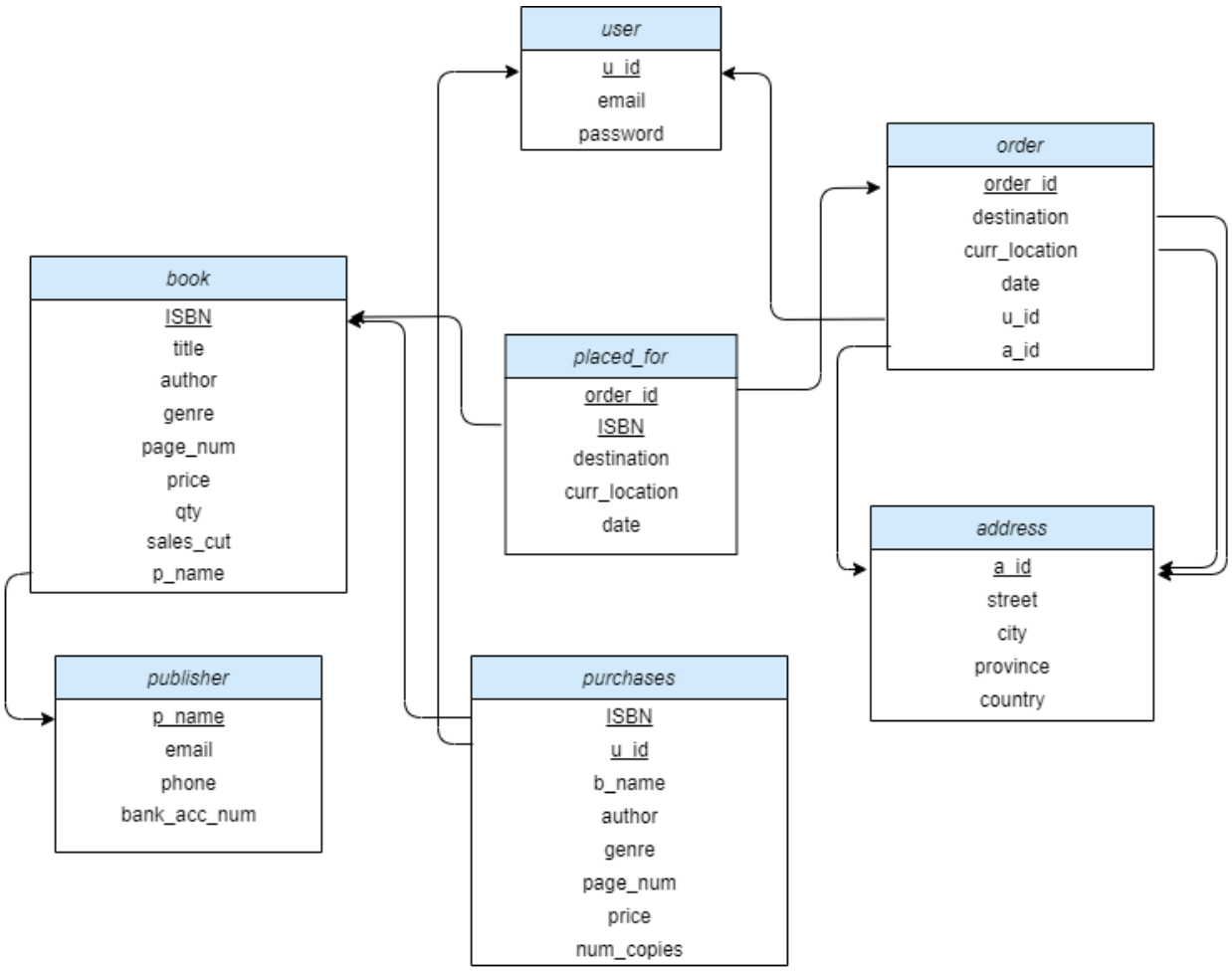
From above we can see that for each relation, F is composed of only one functional dependency representing the primary key constraint. This means that each relation is already in BCNF and no normalization is required for these.
Placed_for:
$order\_id \rightarrow destination, curr\_location, date$
$ISBN \rightarrow title, author, genre, page_num, price, qty, sales\_cut$

Since ISBN isn't a superkey, we must decompose.

$R_1(ISBN, title, author, genre, page\_num, price, qty, sales\_cut)$
$R_2(order\_id, destination, curr\_location, date, ISBN)$

We'll use R_2 in place of $placed\_for$, and discard R_1 because it is simply the *book* relation we already had. We will make the order_id and ISBN the primary key since an order may contain several books. One tuple should represent a particular book in an order.

2.4. Database Schema Diagram



2.5. Implementation

I unfortunately have opted to leave out the implementation.