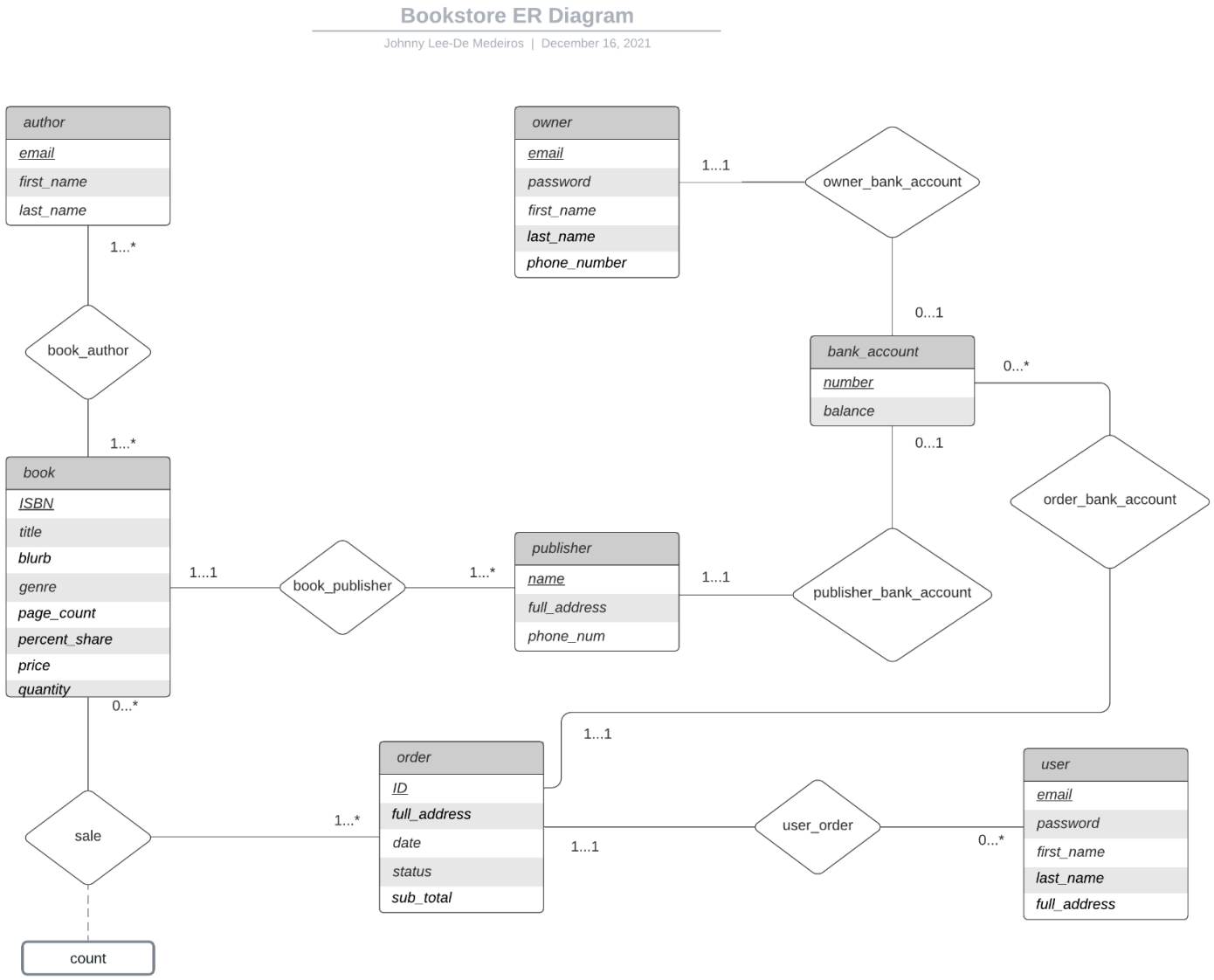
Conceptual Design



Assumptions Made:

All addresses are saved as “full_address”. These represent the street number, city, province/state, postal code in one string. Example: “123 Billsly ave, Bingbong, Kentucky, M4D 3UP”.

Author — An author is only an author in this database if they have written at least one book, therefore they have a many-to-many cardinality with book and full participation. They do not have a relationship set with publishers as it is unneeded in this application.

Book — A book must have at least one author, therefore it has a many-to-many relationship with author and full participation. A book must have exactly one publisher, therefore it has a

many-to-one relationship with publisher and full participation. A book may have never been ordered or multiple times, therefore it has a many-to-many relationship with order but partial participation.

Publisher — A publisher is only a publisher in this database if it has published at least one or more books, therefore it has a one-to-many relationship with book and full participation. A publisher must have exactly one bank account in order to receive the percent share of the revenue from a book sale, therefore it has a one-to-one relationship with bank_account and full participation.

Bank_Account — A bank account may have zero or one publisher associated with it since it can be associated with a publisher or an owner, therefore it has a one-to-one relationship with publisher but partial participation. A bank account may have zero or one owner associated with it since it can be associated with an owner or a publisher, therefore it has a one-to-one relationship with owner but partial participation. A bank account may have zero or multiple orders with which they are associated with, therefore it has a one-to-many relationship with order but partial participation. Bank accounts are more of a way to keep track of cash transactions more than represent that party's actual banking information. In hindsight, it's odd if a bookstore stored your entire bank account which should be handled by a third party; the bank.

Owner — There should only be a tuple in this table as the bookstore has one owner. An owner must have exactly one bank account in order to pay for new quantities of books when their supply goes below a threshold, therefore it has a one-to-one relationship with bank_account and full participation.

Order — An order must contain an associated ban account that it charged the sub_total to, therefore it has a many to one relationship with bank_account and full participation. An order must contain at least one book, therefore it has a many-to-many relationship with book and full participation. I assume that when an order is made and paid for, it is a sale which is the relationship set between book and order. Sale will have a count attribute for the amount of that particular book that was ordered for that unique order. An order must have exactly one user who put in the order, therefore it has a many-to-one relationship with user and full participation. I assume that all orders are instantly shipped to the user via an ebook copy sent to their email and as such will have a status of "Delivered to Email". I assume that if a bookstore is competent enough to be online, then it also sends an ebook copy to the user who orders any book. Note: this does not affect the ER diagram either way. Orders do not lookup the associated user's address as it does not need to match the one used in their registration, therefore it must have an attribute to keep track of this.

User — A user can order zero or more books, therefore it has a one-to-many relationship with order but partial participation. A user does not have the ability to save their billing information when registering as it is not needed and must input it on each order. An order

Reduction to Relation Schemas

Author(email, first_name, last_name)
book_author(email, ISBN)
book(ISBN, pub_name, title, blurb, genre, page_count, percent_share, price, quantity)
publisher(name, bank_number, full_address, phone_num)
bank_account(number, balance)
owner(email, bank_number, password, first_name, last_name, phone_number)
sale(ISBN, ID, count)
order(ID, user_email, bank_number, full_address, date, status, sub_total)
user(email, password, first_name, last_name, full_address)

Normalization of Relation Schemas

Author
Each tuple must have a unique email.

R = {email, first_name, last_name}
F = {
email → first_name, last_name }

Is it in BCNF?
Compute the closure of email:

result = email
email → first_name, last_name: result = email, first_name, last_name

email+ = {email, first_name, last_name}

Email determines R, therefore it is a Superkey. Because all functional dependencies' lefthand sides are Superkeys, author is already in BCNF.

book_author
Because both email and ISBN are primary keys, tuples are valid only if the combination of both values are not a duplicate. This means that the only functional dependencies are trivial (email → email, ISBN → ISBN), therefore book_author is already in BCNF.

book
Each tuple must have a unique ISBN. Each book should have a unique blurb paragraph summarizing the contents of the book.

R = {ISBN, title, blurb, genre, page_count, pub_name, percent_share, price, quantity}
F = {
ISBN → title, blurb, genre, page_count, pub_name, percent_share, price, quantity
blurb → ISBN}

Is it in BCNF?
Compute the closure of ISBN:

result = ISBN
ISBN → title, blurb, genre, page_count, pub_name, percent_share, price, quantity: result = ISBN, title, blurb, genre, page_count, pub_name, percent_share, price, quantity
ISBN+ = {ISBN, title, blurb, genre, page_count, pub_name, percent_share, price, quantity}

ISBN determines R, therefore it is a Superkey.

Compute the closure of blurb:

result = blurb
blurb → ISBN: result = blurb, ISBN
ISBN → title, blurb, genre, page_count, pub_name, percent_share, price, quantity: result = blurb, ISBN, title, genre, page_count, pub_name, percent_share, price, quantity
blurb+ = {blurb, ISBN, title, genre, page_count, pub_name, percent_share, price, quantity}

Blurb determines R, therefore it is a Superkey.

Because all functional dependencies' lefthand sides are Superkeys, book is already in BCNF.

publisher
Each tuple must have a unique publisher name. A publisher should have a unique address and phone number (not shared with other publishers).

R = {name, full_address, phone_number}
F = {
name → full_address, phone_number
address → name
phone_number → name }

Is it in BCNF?
Compute the closure of name:

result = name
name → address, phone_number: result = name, address, phone_number
name+ = {name, address, phone_number}

Name determines R, therefore name is a Superkey.

Compute the closure of address:

result = address
address → name: result = address, name
name → address, phone_number: result = address, name, phone_number
address+ = {address, name, phone_number}

Address determines R, therefore address is a Superkey.

Compute the closure of phone_number:

result = phone_number
phone_number → name: result = phone_number, name
name → address, phone_number: result = phone_number, name, address
phone_number+ = {phone_number, name, address}

Phone_number determines R, therefore it is a Superkey.

Because all functional dependencies lefthand sides are Superkeys, Publisher is already in BCNF.

bank_account
Each tuple must have a unique bank account number

R = {number, balance}
F = {
number → balance }

Is it in BCNF?
Compute the closure of number:

result = number
number → balance: result = number, balance
number+ = {number, balance}

Number determines R, therefore it is a Superkey.
Because all functional dependencies' lefthand sides are Superkeys, bank_account is already in BCNF.

owner
Each tuple must have a unique email. Because the bookstore should only have a single tuple for the owner, any functional dependencies will always be met.

R = {email, password, first_name, last_name, bank_number, phone_number}

F = {
email → password, first_name, last_name, bank_number, phone_number }

Is it in BCNF?
Compute the closure of email:

result = email
email → password, first_name, last_name, bank_number, phone_number: result = email, password, first_name, last_name, bank_number, phone_number
email+ = {email, password, first_name, last_name, bank_number, phone_number}

Email determines R, therefore it is a Superkey.
Because all functional dependencies' lefthand sides are Superkeys, owner is already in BCNF.

sale
Because ISBN and ID are both primary keys, each tuple is only valid if the combination of both values are not duplicates.

R = {ISBN, ID, count}
F = {
ISBN, ID → count }

Is it in BCNF?
Compute the closure of ISBN, ID:

result = ISBN, ID
ISBN, ID → count: result = ISBN, ID, count
(ISBN, ID)+ = {ISBN, ID, count}

ISBN, ID determines R, therefore it is a Superkey.
Because all functional dependencies' lefthand sides are Superkeys, sale is already in BCNF.

order
Each order must have a unique ID.

R = {ID, user_email, bank_number, full_address, date, status, sub_total}
F = {
ID → user_email, bank_number, full_address, date, status, sub_total }

Is it in BCNF?
Compute the closure of ID:

result = ID

ID → user_email, bank_number, full_address, date, status, sub_total: result = ID, user_email, bank_number, full_address, date, status, sub_total
result+ = {ID, user_email, bank_number, full_address, date, status, sub_total}

Result determines R, therefore it is a Superkey.
Because all functional dependencies' lefthand sides are Superkeys, order is already in BCNF.

user
Each tuple must have a unique email.

R = {email, password, first_name, last_name, full_address}
F = {
email → password, first_name, last_name, full_address }

Is it in BCNF?
Compute the closure of email:

result = email
email → password, first_name, last_name, full_address: result = email, password, first_name, last_name, full_address
email+ = {email, password, first_name, last_name, full_address}

Email determines R, therefore it is a Superkey.
Because all functional dependencies' lefthand sides are Superkeys, user is already in BCNF.

Database Schema Diagram