CSE 3241 Project Checkpoint 04 – Functional Dependencies and Normal Forms

In a **NEATLY TYPED** document, provide the following:

Names

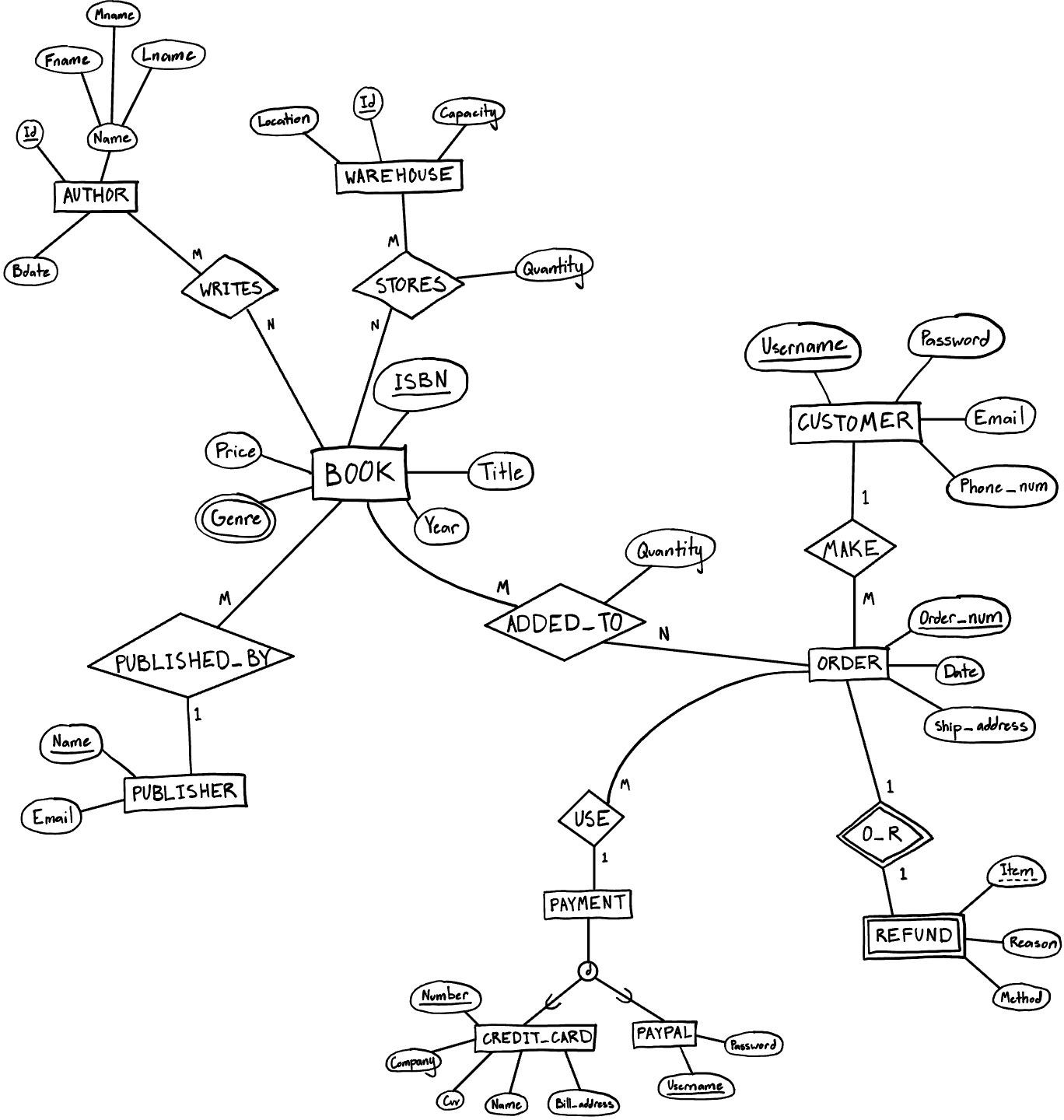
Date

Chaeun Hong, Defne Ceyhan, Maddison Chaffin, Yi-Ting

Nov.14

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 03. **If you were instructed to change the model for Project Checkpoint 03, make sure you use the revised versions of your models.**

**ER Diagram**



**Relational model**

- BOOK(Isbn, Price, Title, Year, P\_name (FK))

* GENRE (B\_isbn (FK), Genre)
* AUTHOR(Id, Fname, Lname, Mname, Bdate)
* WAREHOUSE(Id, Location, Capacity)
* CUSTOMER(Username, Password, Email, Phone\_num)
* PUBLISHER (Name, Email)
* ORDERS(Order\_num, Order\_date, Ship\_address, Customer\_username(FK), Creditcard\_number(FK), Paypal\_username(FK))
* CREDIT\_CARD(Number, Company, Cvv, Name, Bill\_address)
* PAYPAL(Username, Password)
* STORES (B\_isbn(FK), W\_id(FK), Quantity )
* WRITES (B\_isbn(FK), A\_Id(FK))
* ADDED\_TO (B\_isbn(FK), O\_order\_num(FK), Quantity)
* REFUND(Order\_num(FK), Item, Count, Reason, Method)

1. For each relation schema in your model, indicate the functional dependencies. Think carefully about what you are modeling here - make sure you consider all the possible dependencies in each relation and not just the ones from your primary keys. For example, a customer’s credit card number is unique, and so will uniquely identify a customer even if you have another key in the same table (in fact, if the customer can have multiple credit card numbers, the dependencies can get even more involved).

BOOK

Isbn -> {Price, Title, Year, P\_name} GENRE (no functional dependencies) AUTHOR

Id -> {Fname, Lname, Mname, Bdate} WAREHOUSE

Id -> {location, capacity} CUSTOMER

Username -> {Password, Email, Phone\_num} PUBLISHER

Name -> {email} ORDERS

Order\_num->{Order\_date, Ship\_address, Customer\_username, Creditcard\_numer, Paypal\_username} CREDIT\_CARD

Number -> {Company, Cvv, Name, Bill\_address} PAYPAL

Username -> Password STORES

B\_isbn, W\_id -> {Quantity}

WRITES (no functional dependencies) ADDED\_TO

B\_isbn, O\_order\_num -> {Quantity} REFUND

Order\_num, Item -> {Count, Reason, Method}

1. For each relation schema in your model, determine the highest normal form of the relation. If the relation is not in 3NF, rewrite your relation schema so that it is in at least 3NF.

BOOK: BCNF GENRE: BCNF AUTHOR: BCNF WAREHOUSE: BCNF CUSTOMER: BCNF PUBLISHER: BCNF ORDERS: BCNF

CREDIT\_CARD: BCNF PAYPAL: BCNF STORES: BCNF WRITES: BCNF ADDED\_TO: BCNF REFUND: BCNF

1. For each relation schema in your model that is in 3NF but not in BCNF, either rewrite the relation schema to BCNF or provide a short justification for why this relation should be an exception to the rule of putting relations into BCNF.

There is no relation that is in 3NF but not in BCNF.

1. For your database, propose at least two interesting views that can be built from your relations. These views must involve joining at least two tables together each and must include some kind of aggregation in the view. Each view must also be able to be described by a one or two sentence description in plain English. Provide the code for constructing your views along with the English language description of what the view is supposed to be providing.

-- retrieve the number of sold books for each genre CREATE VIEW VIEW1 AS

SELECT G.Genre, SUM(AT.Quantity) AS num\_of\_sold\_books FROM GENRE AS G, ADDED\_TO AS AT

WHERE G.B\_isbn=AT.B\_isbn GROUP BY G.Genre

ORDER BY num\_of\_sold\_books DESC;

-- retrieve a specific customer (eg Edwin5) for an order of preferred genres based on his/her purchasing record CREATE VIEW VIEW2 AS

SELECT G.Genre, COUNT(AT.B\_isbn) AS number\_of\_books FROM ORDERS AS O, ADDED\_TO AS AT, GENRE AS G

WHERE O.Order\_num=AT.O\_order\_num

AND AT.B\_isbn=G.B\_isbn

AND O.Customer\_username='Edwin5' GROUP BY G.Genre

ORDER BY number\_of\_books DESC;

-- retrieve the book's ISBN with less than 100 in stock in each warehouse in Columbus

SELECT AT.B\_isbn AS ISBN, W.Id AS warehouse\_id, (SUM(S.Quantity)-SUM(AT.Quantity)) AS remaining\_stock FROM STORES AS S, ADDED\_TO AS AT, BOOK AS B, WAREHOUSE AS W

WHERE B.Isbn=AT.B\_isbn AND S.B\_isbn=AT.B\_isbn AND W.Id=S.W\_id

AND W.Location = 'Columbus' GROUP BY S.B\_isbn

HAVING remaining\_stock < 100 ORDER BY W.Id;