Joseph Edradan
4/5/20
CIS-363-OLH-Enterprise Data Mgmt. w MySQL-CRN 42508

Phase 2 of your *Personal Collection Database Application* Mapping from E/R Model to Relational Design

Using MySQL Workbench rather than Lucid chart to create the E/R Diagram while also fixing and modifying the previous E/R table.

Relations:

User → Customer
User → Distributor
User → Manufacturer
User → Seller
The User table simplifies the accounts on the site with something they all have in common which is the standard information about the User. It is not possible to combine all the types of users into one because they all require different attributes to be fulfilled. This is meant for contact information used between the admin or between users.

Credit_Card → Seller
Credit_Card → Customer
Only Customers and Sellers can buy and sell things on the site and since both of those types of users need a credit card, they both get one from the Credit_Card table. It also allows for either the Seller or Customer to quickly call up their credit card information used in a transaction.

Prime_member → Customer
A Customer can have prime, but it's not required which is why Prime has its own table and a Customer_ID is assigned to a Prime_ID. It also allows for the admin to show how many customers are Prime members.

Prime_payment → Prime_member
A Prime member can have 0 or many payments and those payments need to be fulfilled in order to retain prime membership.

Distributor → Shipping_Information
Distributors can have Shipping_Information rows to indicate that an order needs to be fulfilled.

Customer → Order
A Customer can make 0 or many orders. An Order contains the customer who has made the order and the shipping information for that order.

Order → Shipping_Information
Order is in a Shipping_Information row to indicate that for Shipping_Information to exist, and Order needs to be made. Orders themselves are implicitly unique.

Order → Item_Order
An Order contains 1 or many items. It doesn't have many attributes to remove repetition on Shipping_Information. Orders themselves are implicitly unique.

Seller → Seller_product_listing
Sellers create Seller_product_listings for the products they want to sell. It can mean 1 Seller can have many items to sell as in different products or of some quantity of 1 product type.

Product → Seller_product_listing
A Seller_product_listing corresponds to 1 unique product.

Manufacturer → Product
Manufacturers create products. Products themselves are unique implicitly. If there are no more products in this table then nothing can be sold or bought.

Company → Manufacturer
Companies can have many manufacturers; this is to indicate that products can be made differently and in different places which allows for a trace back if a product has a problem. It also allows for Users and the admin to complain to them.

Product_site_ID → Seller_product_listing
Product_site_ID corresponds to the Seller_product_listing which essentially means a product from a seller can be bought at their listed price. Product_site_ID is just an ID because information about the product itself may not be necessary for an operation. An example is that an operation joining the Product_site_ID and Product_ID to show a product's comments. In this situation, information about the product is not necessary to display.

Product_site_ID → Product_information
Product_information has information about the product itself. It is used for the product page or for information about the product in meta data. It is separate from Product_site_ID because it contains a lot of data such as the image of the product and text about the product which is not always needed to be loaded.

Product_site_ID → Product_tag_Pair
Product_site_ID can have a list of tag IDs which correspond to what the tag is. It's an intermediate between a many to many relationship.

Tag_information → Product_tag_Pair
Tag_information has a tag title. Tag_information corresponds to a Product_tag_pair which then corresponds to Product_site_ID.

User → Product_comment
Users can make comments on a product's pages.

Product_site_ID → Product_comment
Product_site_ID can have Product_comment rows to indicate that it has many comments.

Customer → Shopping_cart_item
Customers can add items of a Product_site_ID because the product's themselves corresponds to actual items and a shopping cart item does not guarantee an actual product because bought/sold products are of actual value since they are apart of actual transactions.

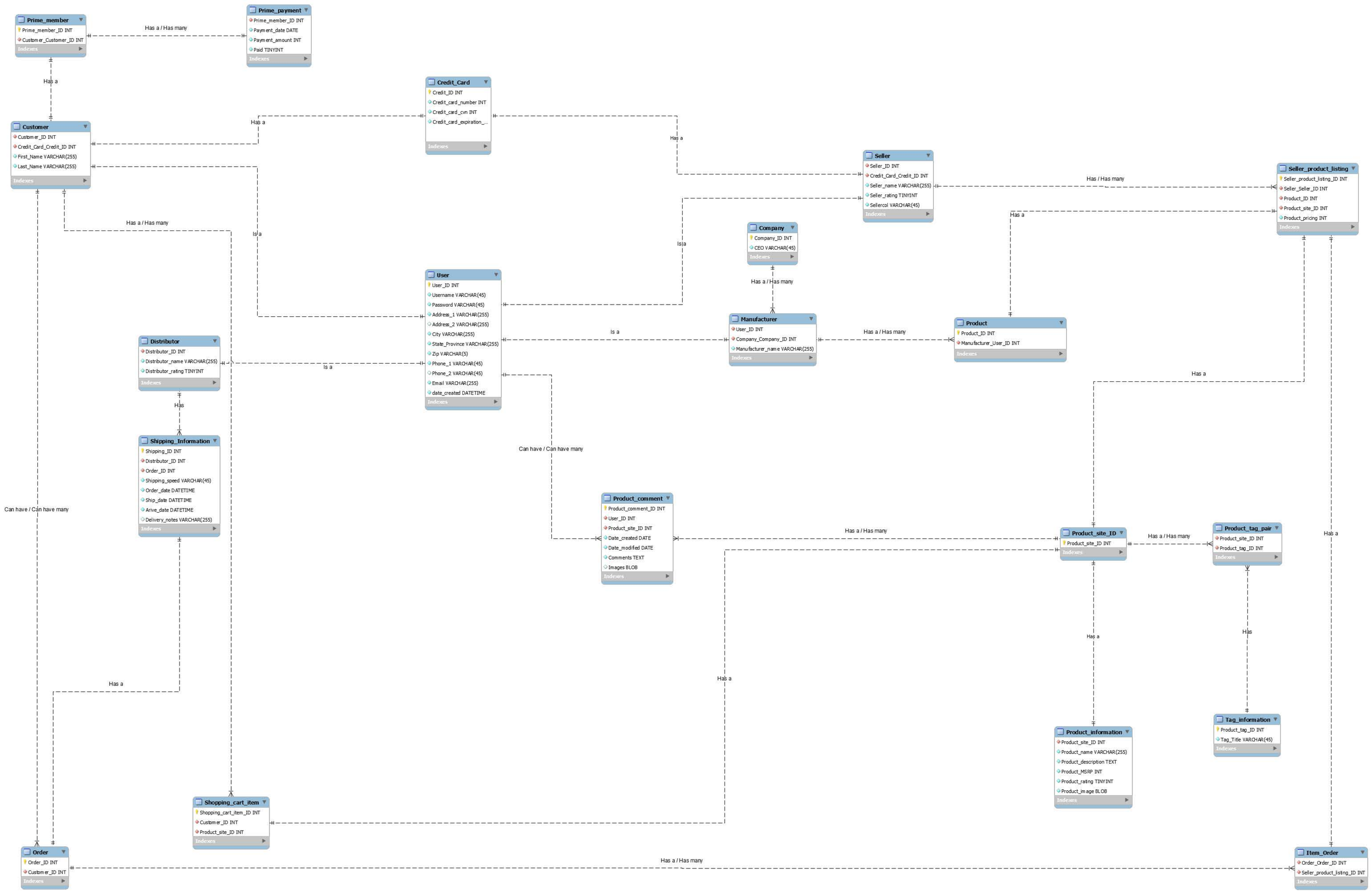
Normalization:

1. There are no redundancies, smaller tables with many relations such as Product_site_ID can be used for many different operations that may not need a lot of unnecessary information.
2. There are no relations that violate the normal forms such as the 4 types of Users and Seller_product_listing.
3. Combine or Decompose relations:
    a. **Users**: it is split up into many types because not every User is the same as a Customer is not a Company.
    b. **Shipping_information, Order, and Item_Order**: They are separate to remove repetition of Shipping_information for multiple items in an order. Order is alone because an Order can have many items bought which is put into the Item_Order table.
    c. **Seller_product_listing, Product, Seller, and Manufacture**: Sellers can sell multiple things or multiple things of 1 type of product. Products are unique and themselves are independent of the Seller_product_listing because different Sellers can sell that same type of product at a different price. For example, toilet paper sold today.
    d. **Product, Product_site_ID, Seller_product_listing, Product_tag_Pair, Tag_information, Product_information, Product_comment, and Shopping_cart_item**: Product_site_ID indicates a product sold by a Seller which is the Seller_product_listing. Product_tag_pair and Tag_information allows for Product_site_ID to have multiple tags. Product_information may not be needed for operations involving Product_site_ID, this reduces sever load. Shopping_cart_item goes to Product_site_ID because a shopping cart item does not involve the transaction of an actual Product. Product_comment refers to the Product_site_ID because you don't store comments about an actual Product or an arbitrary Product on the Product that was bought, but on the site where you bought it.
    e. **Customer, Prime_member, and Prime_payment**: Customer may or may not have Prime and if they do have Prime then they have payments of it.

Prime_payment is on it’s own because it may not be necessary to view Prime payments all the time.

f. **Seller, Customer, Credit_Card**: Not every user buys or sells things, they might be on the site to respond to customer or seller if they have problems with a Product.
g. **Company and Manufacturer**: A Company can have many manufacturers in different places.

4. Somethings I don’t like about the schema Is the way the Prime, Credit_Card and Shopping_cart_item tables are made because Prime and Credit_Card require an ID to be made first before the Customer and Customer/Seller can be made. If Product wasn’t a unique object tied to the existence of a Seller_product_listing, Shopping_cart_item could possibly be designed better.
5. The schema is ok.


Prime_member
Prime_member_ID INT
Customer_Customer_ID INT
Indexes
Has a / Has many
Prime_payment
Prime_member_ID INT
Payment_date DATE
Payment_amount INT
Paid TINYINT
Indexes
Has a
Credit_Card
Credit_ID INT
Credit_card_number INT
Credit_card_cvn INT
Credit_card_expiration_...
Indexes
Has a
Customer
Customer_ID INT
Credit_Card_Credit_ID INT
First_Name VARCHAR(255)
Last_Name VARCHAR(255)
Indexes
Has a
Seller
Seller_ID INT
Credit_Card_Credit_ID INT
Seller_name VARCHAR(255)
Seller_rating TINYINT
Sellercol VARCHAR(45)
Indexes
Has / Has many
Seller_product_listing
Seller_product_listing_ID INT
Seller_Seller_ID INT
Product_ID INT
Product_site_ID INT
Product_pricing INT
Indexes
Has a / Has many
Is a
Is a
Company
Company_ID INT
CEO VARCHAR(45)
Indexes
Has a
User
User_ID INT
Username VARCHAR(45)
Password VARCHAR(45)
Address_1 VARCHAR(255)
Address_2 VARCHAR(255)
City VARCHAR(255)
State_Province VARCHAR(255)
Zip VARCHAR(5)
Phone_1 VARCHAR(45)
Phone_2 VARCHAR(45)
Email VARCHAR(255)
date_created DATETIME
Indexes
Has a / Has many
Manufacturer
User_ID INT
Company_Company_ID INT
Manufacturer_name VARCHAR(255)
Indexes
Has a / Has many
Product
Product_ID INT
Manufacturer_User_ID INT
Indexes
Distributor
Distributor_ID INT
Distributor_name VARCHAR(255)
Distributor_rating TINYINT
Indexes
Is a
Has
Shipping_Information
Shipping_ID INT
Distributor_ID INT
Order_ID INT
Shipping_speed VARCHAR(45)
Order_date DATETIME
Ship_date DATETIME
Arive_date DATETIME
Delivery_notes VARCHAR(255)
Indexes
Can have / Can have many
Can have / Can have many
Product_comment
Product_comment_ID INT
User_ID INT
Product_site_ID INT
Date_created DATE
Date_modified DATE
Comments TEXT
Images BLOB
Indexes
Has a / Has many
Product_site_ID
Product_site_ID INT
Indexes
Has a / Has many
Product_tag_pair
Product_site_ID INT
Product_tag_ID INT
Indexes
Has a
Has
Has a
Has a
Product_information
Product_site_ID INT
Product_name VARCHAR(255)
Product_description TEXT
Product_MSRP INT
Product_rating TINYINT
Product_image BLOB
Indexes
Tag_information
Product_tag_ID INT
Tag_Title VARCHAR(45)
Indexes
Has a
Has a
Shopping_cart_item
Shopping_cart_item_ID INT
Customer_ID INT
Product_site_ID INT
Indexes
Order
Order_ID INT
Customer_ID INT
Indexes
Has a / Has many
Item_Order
Order_Order_ID INT
Seller_product_listing_ID INT
Indexes