Shanqing Gu

MSDS 7330 HW4 Due Date: 06/05/2018

Unit 4: HomeWork

Design Question:

> Building on the ecommerce database design we are working on, the retailer wants to add support for "Gift orders". The customer will now be able to order gifts for family and friends on a single order and have them shipped to different addresses.

- How would you change your database design to accommodate this requirement?
- Would you create new tables or change existing tables?
- Your ecommercedb now has close to 10k orders already. How will the requirement impact the existing orders in the tables? Are there special design considerations for handling the existing orders?

Turn in an updated ER model for the Ecommerce DB showing the design changes in addition to a word document addressing the questions above.

Answers for HW4:

Q1 (How would you change your database design to accommodate this requirement?) and Q2 (Would you create new tables or change existing tables?)

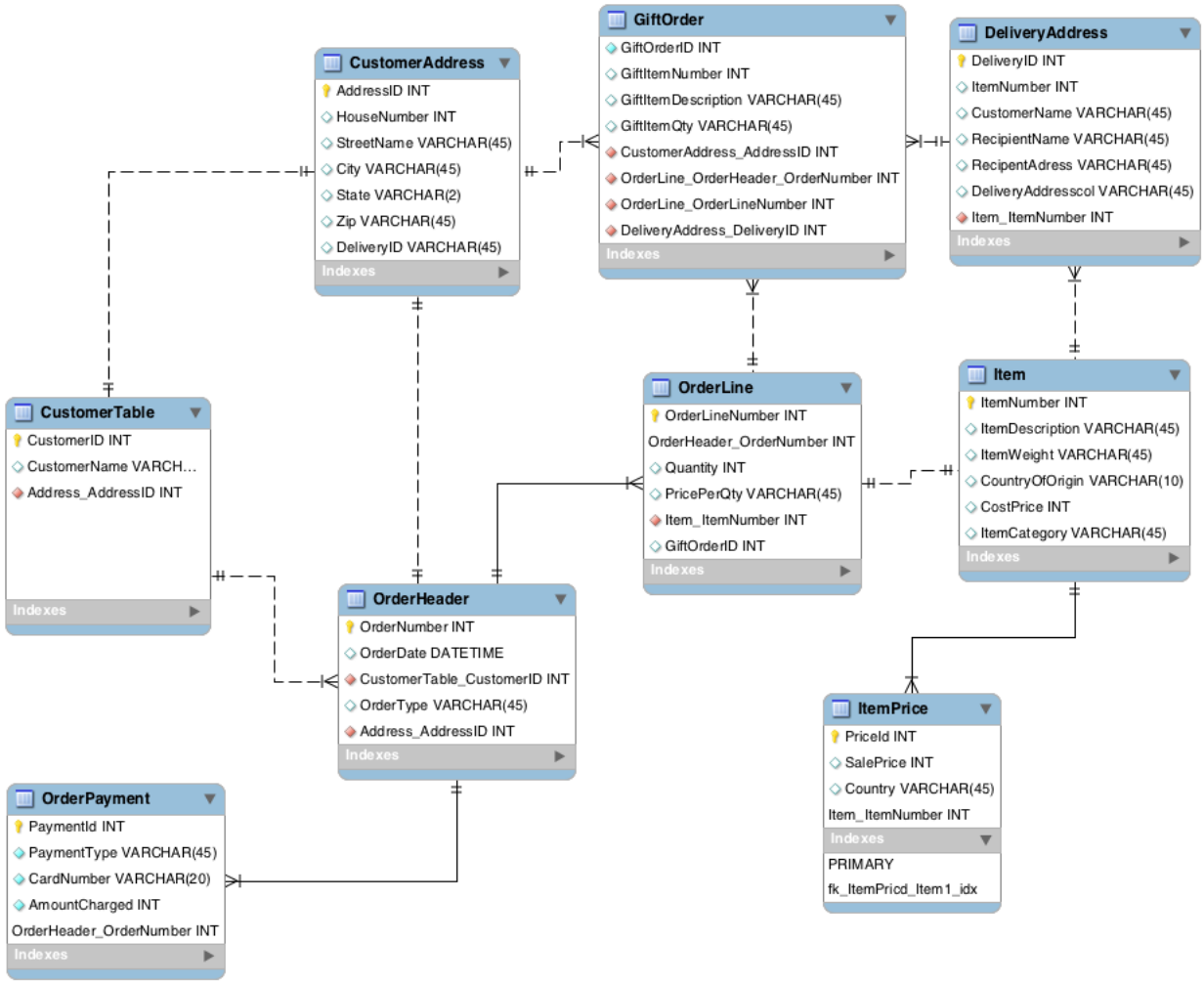
In the previously established Entity-Relationship (ER) Diagram, it only includes CustomerAddress table to deliver orders to customers who place orders. For the new "Gift orders" requirements, we can create new tables or change existing tables. However, creating new entity tables will be the easy and better way to address this question.

In order to meet the requirement to include gift order delivery, a new entity table DeliveryAddress should be set up for delivering gifts to family and friends.

For the requirement of ordering gifts for family and friends on a single order, another new entity table GiftOrder should be used to organize the single order in table OrderLine.

In addition, build the relations to hold the associations between table GiftOrder and OrderLine, DeliveryAddress, and CustomerAddress.

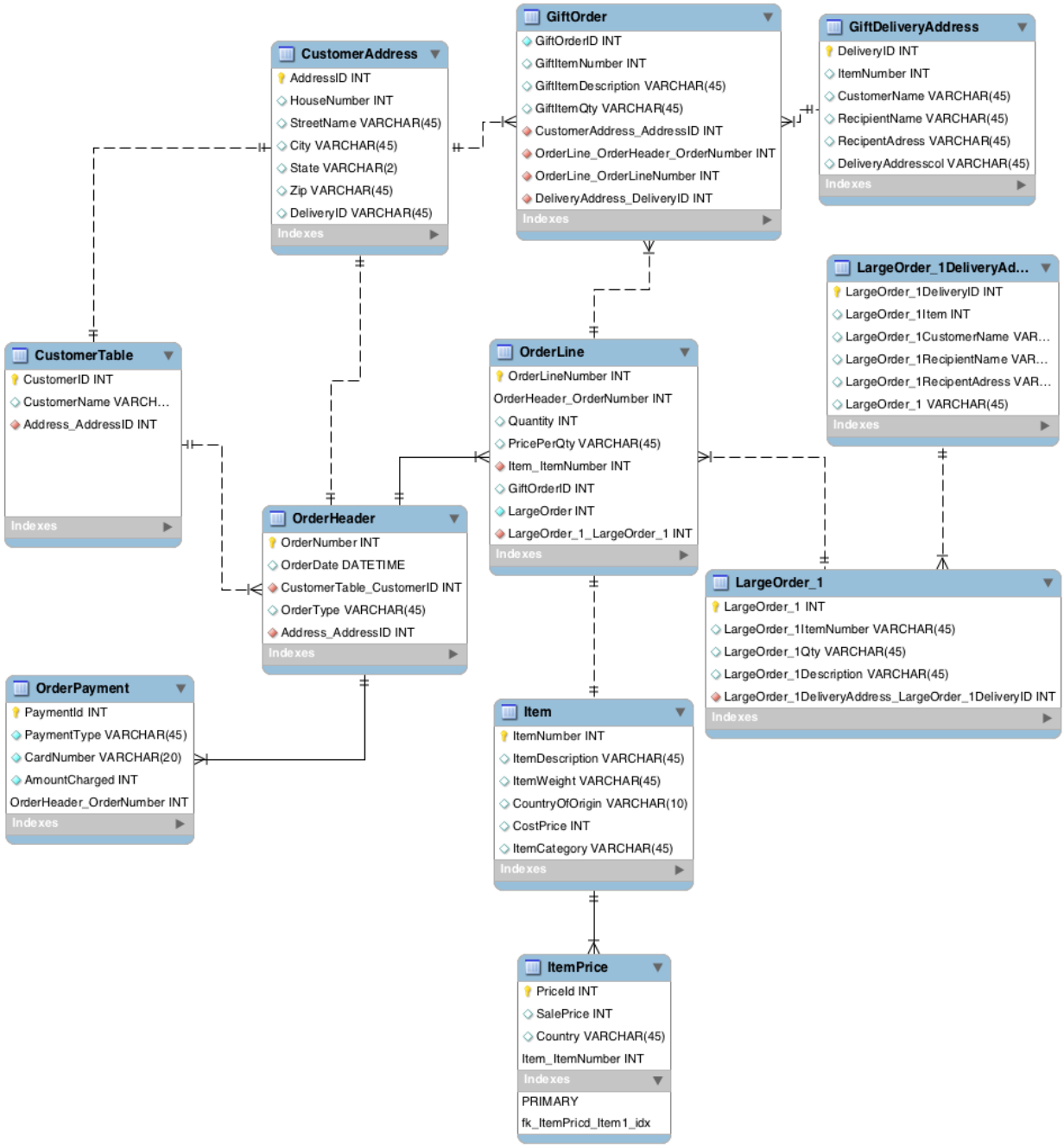
The Modified database design to meet "Gift orders" requirement is shown below:



Q3: Your ecommercedb now has close to 10k orders already. How will the requirement impact the existing orders in the tables? Are there special design considerations for handling the existing orders?

If the ecommercedb now has close to 10k orders already, the ER model will become less efficient. Therefore, we need to partition or index the database to enhance the performance, manageability and availability of data. For example, in the OrderLine table, we can set up different categories (such as GiftOrder, LargeOrder_1 … n, etc.). Each category associates with corresponding delivery address.

The modified model is shown below:


GiftOrder
GiftOrderID INT
GiftItemNumber INT
GiftItemDescription VARCHAR(45)
GiftItemQty VARCHAR(45)
CustomerAddress_AddressID INT
OrderLine_OrderHeader_OrderNumber INT
OrderLine_OrderLineNumber INT
DeliveryAddress_DeliveryID INT
Indexes
GiftDeliveryAddress
DeliveryID INT
ItemNumber INT
CustomerName VARCHAR(45)
RecipientName VARCHAR(45)
RecipentAdress VARCHAR(45)
DeliveryAddresscol VARCHAR(45)
Indexes
CustomerAddress
AddressID INT
HouseNumber INT
StreetName VARCHAR(45)
City VARCHAR(45)
State VARCHAR(2)
Zip VARCHAR(45)
DeliveryID VARCHAR(45)
Indexes
LargeOrder_1DeliveryAd...
LargeOrder_1DeliveryID INT
LargeOrder_1Item INT
LargeOrder_1CustomerName VAR...
LargeOrder_1RecipientName VAR...
LargeOrder_1RecipentAdress VAR...
LargeOrder_1 VARCHAR(45)
Indexes
CustomerTable
CustomerID INT
CustomerName VARCH...
Address_AddressID INT
Indexes
OrderLine
OrderLineNumber INT
OrderHeader_OrderNumber INT
Quantity INT
PricePerQty VARCHAR(45)
Item_ItemNumber INT
GiftOrderID INT
LargeOrder INT
LargeOrder_1_LargeOrder_1 INT
Indexes
OrderHeader
OrderNumber INT
OrderDate DATETIME
CustomerTable_CustomerID INT
OrderType VARCHAR(45)
Address_AddressID INT
Indexes
LargeOrder_1
LargeOrder_1 INT
LargeOrder_1ItemNumber VARCHAR(45)
LargeOrder_1Qty VARCHAR(45)
LargeOrder_1Description VARCHAR(45)
LargeOrder_1DeliveryAddress_LargeOrder_1DeliveryID INT
Indexes
OrderPayment
PaymentId INT
PaymentType VARCHAR(45)
CardNumber VARCHAR(20)
AmountCharged INT
OrderHeader_OrderNumber INT
Indexes
Item
ItemNumber INT
ItemDescription VARCHAR(45)
ItemWeight VARCHAR(45)
CountryOfOrigin VARCHAR(10)
CostPrice INT
ItemCategory VARCHAR(45)
Indexes
ItemPrice
PriceId INT
SalePrice INT
Country VARCHAR(45)
Item_ItemNumber INT
Indexes
PRIMARY
fk_ItemPricd_Item1_idx