Project Proposal and Outline - Final Revision CS 340 – Oregon State University
Kaitlyn Coyle and Chris Jacobs Winter 2022

Peer Feedback

“After reading over all the details I did notice as many others had already the varchar for the item_id when it might be more work to have it that was since they'd be sorting by the characters rather than sorting it more easily by integer. If the item ids are to also be automatically produced then having it be a randomized integer will be more beneficial. all the other datatypes are clearly depicted.”

“While I don't see a huge issue with some of the plural/non-plural use of naming conventions, I would say for the sake of programming or just the outline in general you need to keep things named fairly the same otherwise it could cause difficulty for fellow coders or anybody that is to observe your code.”

“Naming is not consistent between overview and outline entities. Entities should be plural ex: (change Order to Orders in outline). All attributes are singular. The Item_id attribute for the items entity is capitalized while the others aren't.”

“For the most part, this is done well. The Item PK however is a varchar when it should be an int if planning to use auto-increment.”

“The overview is not consistent with the entities at times. For example, "Customer" in the overview does not match the "Customers" entity and "Payments" in the overview does not match the "Payment" entity. Attributes are all singular, while some entities are singular and some are plural (it's suggested to make all the entities plural). Entities are correctly capitalized and attributes use snakecase and for the most part are not capitalized, one exception being the "Item_id" attribute for the "Item" entity.”

“The entities are explained and their attributes and relationships listed. I did notice some non ideal data types for certain attributes. For example, under the "Item" entity we have the "Item_id" attribute, which is listed as "varchar", or a string of characters. I'm assuming the item IDs are numerical, so it would be better to use an "int" data type. If the item ID is actually a combination of numbers and letters, then I'm not sure the database can "auto_increment" such a value.”

Revisions

1. (Peer) We fixed type errors that were pointed out in the feedback we received.

2. (Peer) We fixed the discrepancies in the naming of Attributes and Entities (plural vs singular) and fixed various discrepancies in capitalization. We also updated our naming schema for attributes to be more succinct - attribute names no longer prefixed with table names and foreign keys are not longer prefixed with “fk_”

3. We separated customer phone numbers and customer email addresses into separate entities (CustomerPhones and CustomerEmails)

4. We got rid of the Payments entity and replaced it with an entity named Transactions. We also added a PayMethods entity, which has a M:M relationship with Customers.

5. We added an Addresses entity and associated it with Customers, PayMethods, and Shipments.

6. We added the tables CustomerAddresses and CustomerPayMethods to allow for our many-to-many relationships between Addresses/PayMethods and Customers.

7. We associated Shipments with Products instead of just with Orders. This is because we wanted the ability for an Order to have separate Shipments, which means that a Shipment will need to be associated with specific Products that are part of that Shipment. Therefore, we also added a ShipmentItems table that tracks individual items within a Shipment.

8. We changed the name of the Items entity to “Products”.

9. We added a ProductPrices entity so that product prices could be tracked over time and not just changed and lost. Without ProductPrices, if a Product price were changed and a closed Order tried to reference the original sale price, we would have a reporting discrepancy because we would only know the new price of the Product, not the price at the time of the Order.

10. We got rid of the composite keys we were using because they were being used incorrectly (composite keys are no longer used). We also fixed how we were using foreign keys in the weak entities like ShipmentItems and OrderItems.

11. We added various attributes in Shipments to track weight, volume, and cost. We added attributes in Products to track name and color and changed the way we were tracking volume (switched to inches cubed).

12. The price of the order is no longer stored in Order, it is only stored within Transaction. Likewise, sub_total is no longer stored in OrderItems and total_weight and total_volume have also been removed from OrderItems. We did this because we now think we will be able to aggregate this type of data using SELECT and JOIN and store the totals in a Transaction and a Shipment.

13. Most of these revisions were done in an attempt to achieve third normal form. We are still validating whether 3NF has been achieved and may continue updating entity relationships in order to achieve 3NF and eventually BCNF.

Overview

CookNook, a small business that sells printed cookbooks, currently sells around 100 books per month. Up until now, low sales volume has allowed CookNook to use Etsy as their online storefront. However, due to an increase in sales volume, CookNook is interested in developing a self-hosted online storefront with a database backend that can handle a minimum of 1000 orders per month. Additionally, because of the increase in sales, CookNook wants to expand their product line into kitchen hardware.

For their database, CookNook needs to track **Customer** account information and online **Orders** that the customer makes, as well as the **Payments** used by Customers. CookNook also needs to track **Items** in their inventory as well as those that are part of **Orders** (**OrderItems**). CookNook ships products themselves, so they need to track **Shipments** related to orders.

Database Outline

---

primary key** ***foreign key* *****foreign key and member of candidate key

Customers
Represents an individual customer.

Attributes

***customer_id**	**INT**	**UNIQUE**	**AUTO_INC**	**NOT NULL**
first_name	VARCHAR			NOT NULL
last_name	VARCHAR,			NOT NULL
date_of_birth	DATE			NOT NULL

Relationships

- **CustomerPhones**: Customers may have many CustomerPhones, CustomerPhones have only one Customer (0:M)
- **CustomerEmails**: Customers must have at least one CustomerEmail, CustomerEmails have only one Customer (1:M)
- **PayMethods**: Customers may have many PayMethods, PayMethods may be used by many Customers (M:M)
- **Addresses**: Customers may have many Addresses, Addresses have many Customers (M:M)
- **Orders**: Customers may have many Orders, Orders must have only one Customer (0:M)

CustomerPhones
Represents a customer's phone number.

Attributes

***phone_id**	INT	UNIQUE	AUTO_INC	NOT NULL
***customer_id*	INT			NOT NULL
phone_number	VARCHAR			NOT NULL
text_alerts_on	TINYINT			NOT NULL

Relationships

- **Customers**: CustomerPhones have only one Customer, Customers may have many CustomerPhones (0:M)

CustomerEmails

Represents a customer's email address

Attributes

***email_id**	**INT**	**UNIQUE**	**AUTO_INC**	**NOT NULL**
****customer_id*	INT			NOT NULL
email_address	VARCHAR			NOT NULL
is_primary_email	TINYINT			NOT NULL

Relationships

- **Customers:** CustomerEmails have only Customer, Customers must have at least one CustomerEmails (1:M)

Addresses

Represents a customer's address that can be used for shipping and/or billing purposes.

Attributes

***address_id**	**INT**	**UNIQUE**	**AUTO_INC**	**NOT NULL**
first_name	VARCHAR			NOT NULL
last_name	VARCHAR			NOT NULL
address1	VARCHAR			NOT NULL
address2	VARCHAR			
city	VARCHAR			NOT NULL
state	VARCHAR			NOT NULL
zip_code	VARCHAR			NOT NULL
country	VARCHAR			NOT NULL

Relationships

- **PayMethods**: An Address may belong to a PayMethod, a PayMethod must have a single Address (0:M)

- **Shipments**: An Address may belong to a Shipment and a Shipment must have a single Address (0:M)

- **Customers**: An Address may have many Customers and a Customer may have many Addresses (M:M)

PayMethods

Represents payment methods used by customers for transactions.

Attributes

***payment_id**	**INT**	**UNIQUE**	**AUTO_INC**	**NOT NULL**
****address_id*	INT			NOT NULL
card_type	VARCHAR			NOT NULL
last_four_digits	INT			NOT NULL
expiration_date	DATE			NOT NULL

Relationships

- **Transactions**: PayMethods are used in at least one Transaction, a Transaction uses a single PayMethod (1:M)

- **Customers**: A PayMethod is used by at least one Customer, a Customer may have many PayMethods (M:M)

- **Addresses**: A PayMethod must have only one Address, an Address may be used with many PathMethod (0:M)

Products

Represents a product that a customer can purchase

Attributes

***product_id**	**INT**	**UNIQUE**	**AUTO_INC**	**NOT NULL**
name	VARCHAR			NOT NULL
color	VARCHAR			NOT NULL
weigh_lbs	DECIMAL			NOT NULL
volume_cubic_inch	DECIMAL			NOT NULL
in_stock_qty	INT			NOT NULL
reorder_at_qty	INT			
is_discontinued	TINYINT			NOT NULL

Relationships

- **Orders**: A Product may be associated with many Orders, an Order is associated with one or many Products (M:M)

- **Shipments**: A Product may be associated with many Shipments, a Shipment may be associated with many Products (M:M)

- **ProductPrices**: A Product is associated with at least one ProductPrice, a ProductPrice is associated with a single Product (1:M)

ProductPrices

Represents a history of prices for an individual product.

Attributes

***price_id**	**INT**	**UNIQUE**	**AUTO_INC**	**NOT NULL**
***product_id*	INT			NOT NULL
price	DECIMAL			NOT NULL
date_active	DATETIME			NOT NULL
date_inactive	DATETIME			

Relationships

- **Products:** A ProductPrice must be related to a single Product, a Product can have many ProductPrices (1:M)

Orders

Represents a collection of items that are part of a customer's order.

Attributes

***order_id**	**INT**	**UNIQUE**	**AUTO_INC**	**NOT NULL**
***customer_id*	INT			NOT NULL
num_items	INT			NOT NULL
datetime_created	DATETIME			NOT NULL
datetime_processed	DATETIME			
is_closed	TINYINT			NOT NULL
when_closed	DATETIME			

Relationships

- **Customers**: An Order must have a single Customer, a Customer may have many Orders (0:M)

- **Products**: An Order must have at least one Products, a Product can be part of many Orders (M:M)

- **Transactions**: An order must have a single Transaction, a Transaction is part of a single Order (1:1)

- **Shipments**: A shipment can belong to only one Order, an Order can be split into zero or many Shipments. We say 0:M here because Shipments are created *after* an Order has been made - therefore an order can have no Shipments.

<u>Shipments</u>

Represents one of many shipments that belong to an order, sent to a customer.

Attributes

***shipement_id**	**INT**	**UNIQUE**	**AUTO_INC**	**NOT NULL**
****order_id***	INT			NOT NULL
****address_id***	INT			NOT NULL
datetime_created	DATETIME			NOT NULL
tracking_number	INT			NOT NULL
total_lbs	DECIMAL			NOT NULL
total_cubic_in	DECIMAL			NOT NULL
shipping_cost	DECIMAL			NOT NULL
datetime_ready	DATETIME			
datetime_shipped	DATETIME			
datetime_arrived	DATETIME			

Relationships

- Orders: A Shipment must belong to a single Order, an Order may have many Shipments (0:M)

- Addresses: A Shipment must have a single Address, an Address may belong to many Shipments (0:M)

- Products: A Shipment has at least one Product, a Product may be part of many Shipments (M:M)

Transactions

Represents an individual transaction i.e. a purchase of an order made by a customer.

Attributes

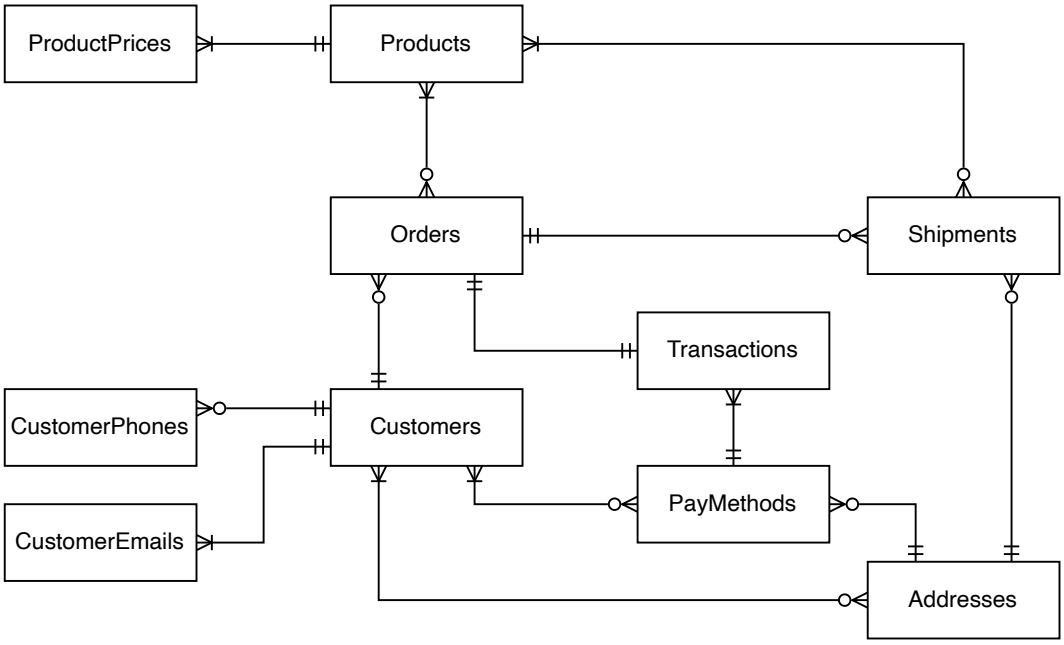
***transaction_id**	**INT**	**UNIQUE**	**AUTO_INC**	**NOT NULL**
****order_id**	**INT**	**UNIQUE**		**NOT NULL**
***payment_id*	INT			NOT NULL
transaction_amount	DECIMAL			NOT NULL
datetime_charged	DATETIME			NOT NULL
confirm_code	VARCHAR			NOT NULL
checksum	VARCHAR			NOT NULL
datetime_success	DATETIME			

Relationships

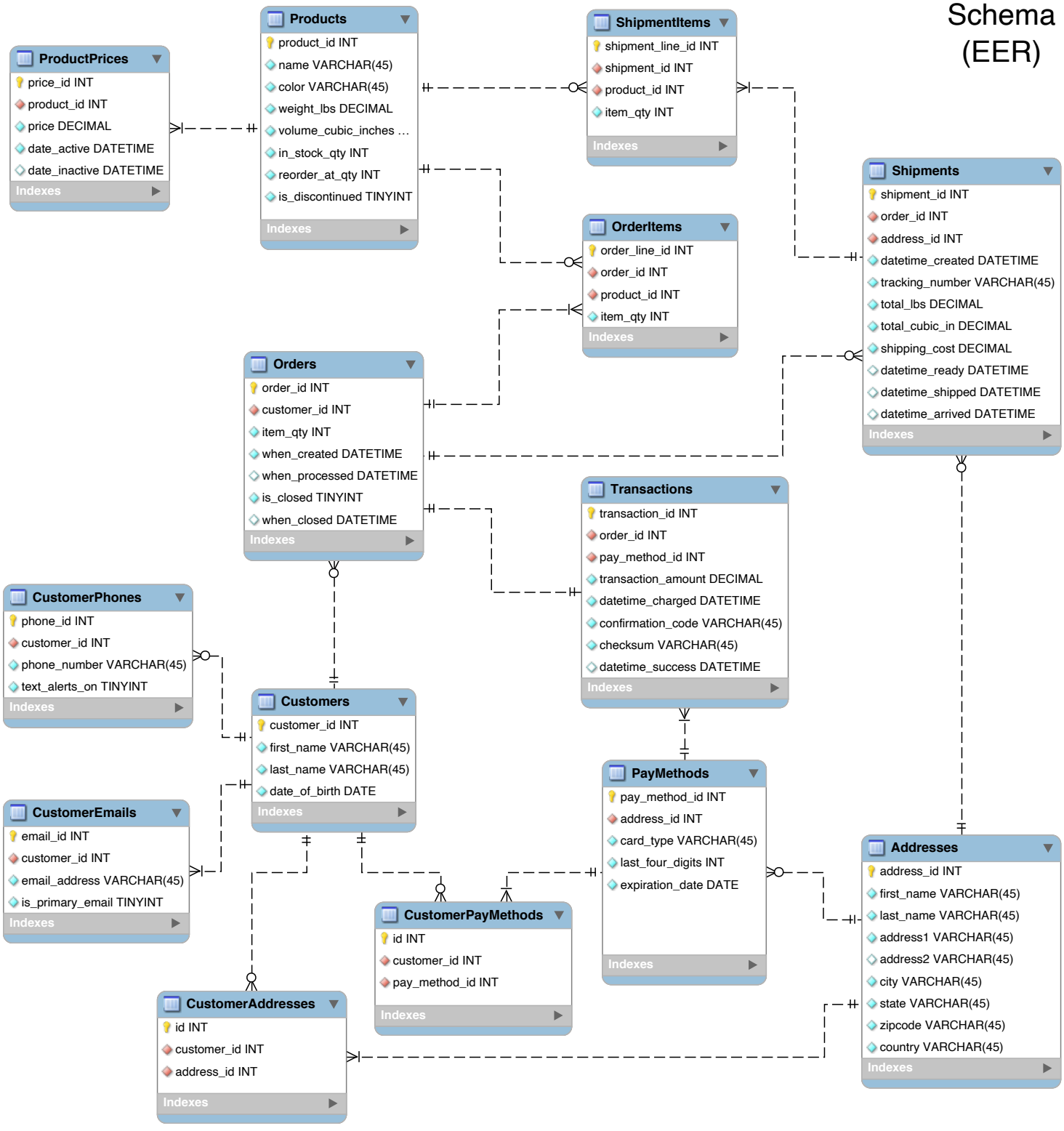
- **Orders**: A Transaction must be part of a single Order, an Order must have a single Transaction (1:1)

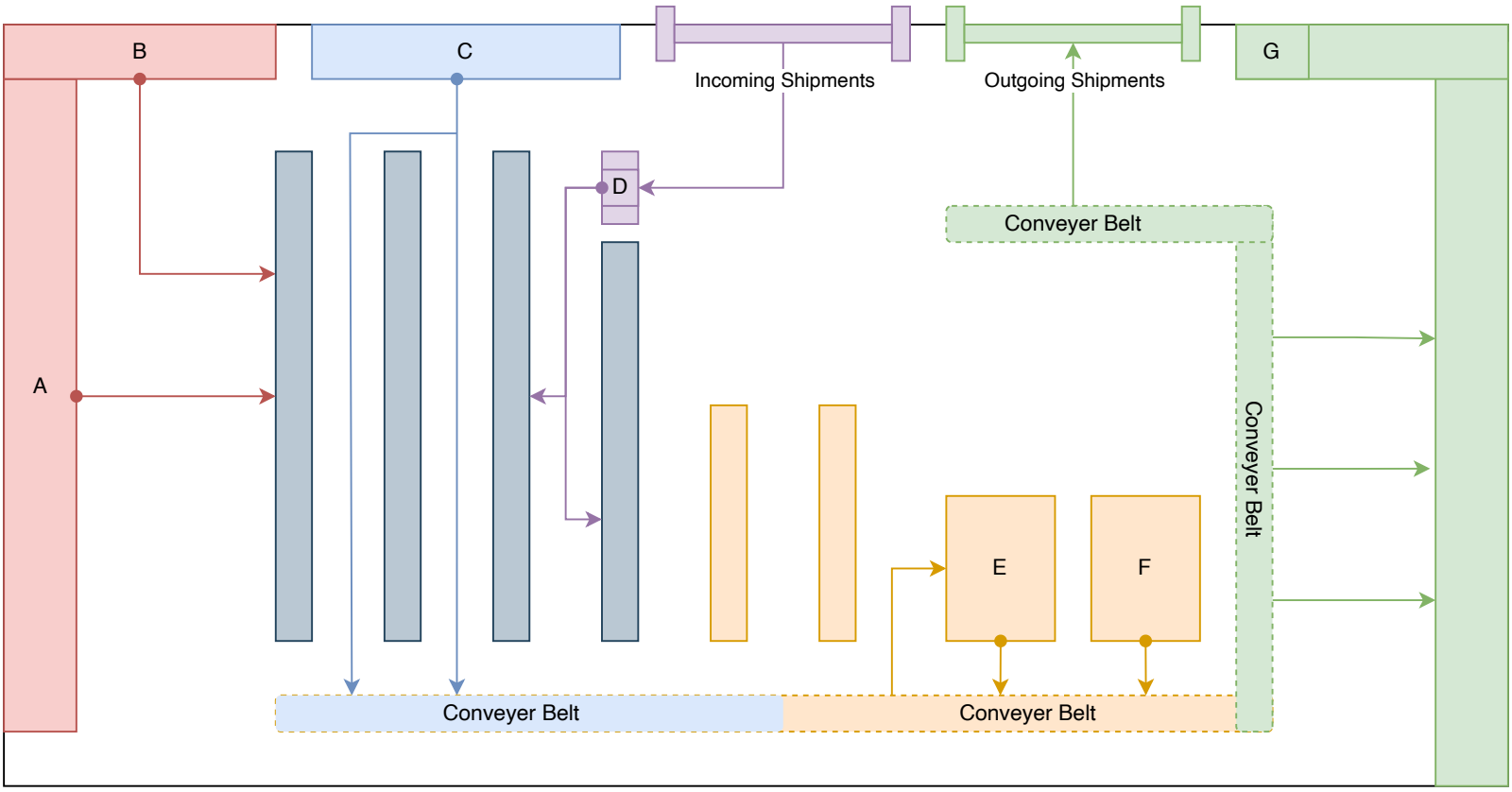
- **PayMethods**: A Transaction must have a single PayMethod, a PayMethod is used for at least one Transaction (1:M)

ERD


ProductPrices
Products
Orders
Shipments
Transactions
CustomerPhones
Customers
CustomerEmails
PayMethods
Addresses

Schema (EER)


Products
product_id INT
name VARCHAR(45)
color VARCHAR(45)
weight_lbs DECIMAL
volume_cubic_inches …
in_stock_qty INT
reorder_at_qty INT
is_discontinued TINYINT
Indexes
ProductPrices
price_id INT
product_id INT
price DECIMAL
date_active DATETIME
date_inactive DATETIME
Indexes
ShipmentItems
shipment_line_id INT
shipment_id INT
product_id INT
item_qty INT
Indexes
Shipments
shipment_id INT
order_id INT
address_id INT
datetime_created DATETIME
tracking_number VARCHAR(45)
total_lbs DECIMAL
total_cubic_in DECIMAL
shipping_cost DECIMAL
datetime_ready DATETIME
datetime_shipped DATETIME
datetime_arrived DATETIME
Indexes
OrderItems
order_line_id INT
order_id INT
product_id INT
item_qty INT
Indexes
Orders
order_id INT
customer_id INT
item_qty INT
when_created DATETIME
when_processed DATETIME
is_closed TINYINT
when_closed DATETIME
Indexes
Transactions
transaction_id INT
order_id INT
pay_method_id INT
transaction_amount DECIMAL
datetime_charged DATETIME
confirmation_code VARCHAR(45)
checksum VARCHAR(45)
datetime_success DATETIME
Indexes
CustomerPhones
phone_id INT
customer_id INT
phone_number VARCHAR(45)
text_alerts_on TINYINT
Indexes
Customers
customer_id INT
first_name VARCHAR(45)
last_name VARCHAR(45)
date_of_birth DATE
Indexes
PayMethods
pay_method_id INT
address_id INT
card_type VARCHAR(45)
last_four_digits INT
expiration_date DATE
Indexes
CustomerEmails
email_id INT
customer_id INT
email_address VARCHAR(45)
is_primary_email TINYINT
Indexes
CustomerPayMethods
id INT
customer_id INT
pay_method_id INT
Indexes
Addresses
address_id INT
first_name VARCHAR(45)
last_name VARCHAR(45)
address1 VARCHAR(45)
address2 VARCHAR(45)
city VARCHAR(45)
state VARCHAR(45)
zipcode VARCHAR(45)
country VARCHAR(45)
Indexes
CustomerAddresses
id INT
customer_id INT
address_id INT
Indexes



Business Logic with Employee Descriptions

A/B: Item/Book Assembly
They manufacture new items. So one or more interfaces that allow them to do the following:

(**1**) View all items that currently need to be restocked (i.e. manufactured by A/B)
(**2**) Create a new item that doesn't exist yet in inventory (e.g a new cookbook that has never been printed that they will now stock in the future).
(**3**) Update the instance of an item (e.g. because they've increased the item stock, no longer stocking an item, updating an item's price, etc)

C: Order Assembly
They are the employees that see a new order first. They find the items needed to complete the order and organize them into bins based on how the items will be shipped. Remember that an order can have many shipments (a shipment must still only be tied to one order).

My original idea was to find a way to do this automatically, so that when Employee C gets an order, shipments are made automatically based on volume/weight. At which point, Employee C just grabs the shipping labels associated with the order and those labels tell Employee C which items to place into which size bin, and then they send that bin down the line to Employees E/F. So groups of items will be assigned to shipments automatically based on the weight/volume of those items.

I don't know how to do this yet in an automated way - so for right now we can just say that that Employee C creates shipments manually. So they need a one or more interfaces that...

(**1**) Allows them to view an order and all of the items associated with that order.
(**2**) Allows them to create an empty shipment.
(**3**) Allows them to assign groups of items to that shipment.

D: Inventory Control (Skip)
We don't need to work on this right now, and doubt we will have time to implement it. Originally I was thinking we needed entities that track manufacturing supplies, order returns, etc.

E/F: Shipment Packaging
These employees package up the item. Since Employee C is doing all of the order and shipment processing and organizing, I don't think E/F will need an interface with the database (although they might eventually).

G: Shipment (Outgoing)

This employee scans a shipment after it has been packaged, then they move the package to a staging area where it waits (this scan would place a datetime value inside the Shipment's "datetime_ready" attribute. When the shipment is offloaded into the truck, the package is scanned again, this time putting a datetime value inside the Shipment's "datetime_shipped" attribute.

We have no way to replicate "scanners", so we will Employee G will need an interface that...
(1) Allows them to find a Shipment based on the shipment_id printed on the shipping label, then be able to press a button that assigns (or removes) a datetime to "datetime_ready", as well as assigns/removes a datetime to "datetime_shipped".