Databases Project
Canadian Tire

COMP 3413
Mohannad Al-Mousa
November 30, 2017

Danny Kivi
Nick Catanzaro
Peter Marinis
Francis Mullins

Original E-R Diagram

See end of PDF. The PNG is also zipped up with the code (/schema.png)

Additional Constraints

Most attributes are set to be not null. All foreign keys are restricted from deletion, and are set to cascade when updated. It is worth noting that the status field on the shipments table, along with roles on the users table could/should be constrained to a list of pending, approved, denied and user, manager, vendor respectively, along with constraining many numbers to only be positive integers.

Relational Schema

users (id, email, password, role)

addresses (id, address, address2, city, state_prov, country, postal_zip)

brands (id, name)

products (upc, product_name, size, msrp, brand_id)
brand_id FK (foreign key) of brand

categories (id, category_name)

category_product (category_id, product_upc)
category_id FK of categories, product_upc FK of products

vendors(id, name)

brand_vendor (brand_id,vendor_id)
brand_id FK of brands, vendor_id FK of vendors

stores (id, name, address_id)
address_id is a foreign key of addresses

stock (product_upc, vendor_id, store_id, quantity, price)
product_upc FK of products, vendor_id FK of vendors, store_id FK of stores

shipments (id, vendor_id, store_id, status, sent_date, recieve_date)
vendor_id FK of vendors, store_id FK of stores

product_shipment (<u>shipment_id</u>, <u>product_upc</u>, quantity, unit_price)
shipment_id FK of shipments, product_upc FK of products

customers (<u>id</u>, user_id, address_id)
user_id FK of users, address_id FK of addresses

orders (<u>id</u>, customer_id, store_id)
customer_id FK of customers, store_id FK of stores

order_product (<u>order_id</u>, <u>product_upc</u>, quantity, unit_price)
order_id FK of orders, product_upc FK of products

Sample Data Generation

Many of the id fields are set to auto-increment, so did not need to be entered. The Users, Brands, Vendors, and Products data was entered by hand. The Brand_vendor relation data was created to ensure not all products were provided by all vendors.

The stocks data was created using the online random data generator at http://mockaroo.com. As the stocks table has a foreign key constraint requiring on the vendor_id, and vendor has a foreign key constraint on the brand in the brand_vendor relation, each insert for stocks had to fulfill the correct brand and vendor. The random data was thus made in batches, with UPCs that matched brands to vendors, and with random stores, prices, and quantities.

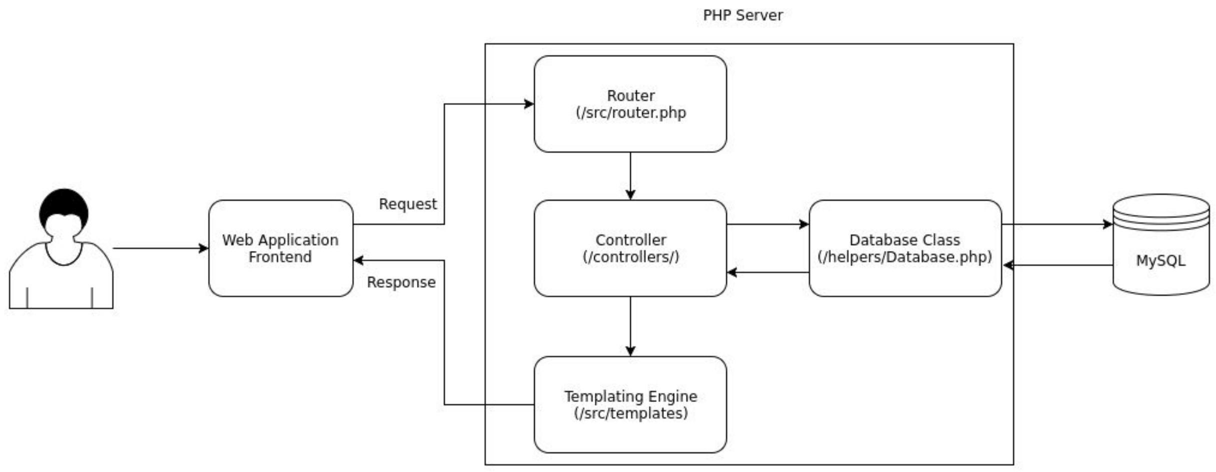
The addresses data was also created with the same random data generator.

Application Assumptions

We don’t have any billing system implemented, and we do not verify any information when a user submits an order, so we assume an order will succeed. Since the cart is only available within session information, we also assume a user will not want to access the application from different systems.

Our database supports many places to store prices, including: the MSRP (manufacturer’s suggested retail price), the price listed in the store, the price the store paid the vendor per unit on a shipment, and the price a user paid the store on an order. However; we assume that the MSRP (stored on the product) is used in all cases inside our application, which means no one will make any money, but it did allow for easier development.

Application Architecture Diagram


PHP Server
Router (/src/router.php
Web Application Frontend
Request
Response
Controller (/controllers/)
Database Class (/helpers/Database.php)
MySQL
Templating Engine (/src/templates)

Application Features

There are three main applications, one each role of Customer, Manager, and Vendor, as well as a login page.

The Login page contains fields for email and password. The email is used to verify the role that the user is authorized for. The password is masked when entered on the page. The user is automatically routed to the appropriate application for their role.

The customer page allows the user to select a store to shop from, and presents a list of products which are in stock at the selected store, which is generated by running query on the Stock table. The user can select items to place in their shopping cart. Different items are able to be added, and multiples of the same item can be added by clicking the same button.

The customer can enter the Cart page at any time, return to the same store and select more items, or select items from a different store, which will be placed in a different order on the cart page.

Entries are made in the Orders and Order_product tables at the time that the user places their order. The Stock table is also updated.

There is also a page for the customer to see the contents of their previous orders. This is generated through a query on the Orders, Order_product, Products, and Customers tables. Orders are linked to customer_id, so a customer will see their past orders but not those of other customers.

The manager role has access to an application which runs a query on the Stock, Brand, Vendor and Products tables, and generates a report which lists all products with 5 or less items in stock at that store. The user is presented the option to reorder items in any quantity. This action creates entries in the Product_shipment and Shipments tables, which are accessible by the Vendor role. The status field of Shipments is set to ‘pending’ until action is taken by the vendor role.

Vendors have access to a page where they can confirm or reject an order which has been placed. Confirming an order updates the Stocks for that store. If the order is accepted or denied, it will update the status field in the Shipments table to ‘approved’ or ‘denied’.

Future Improvements

In an operational public facing database, passwords would be encrypted for transmission, and would not be stored in the clear in the database.

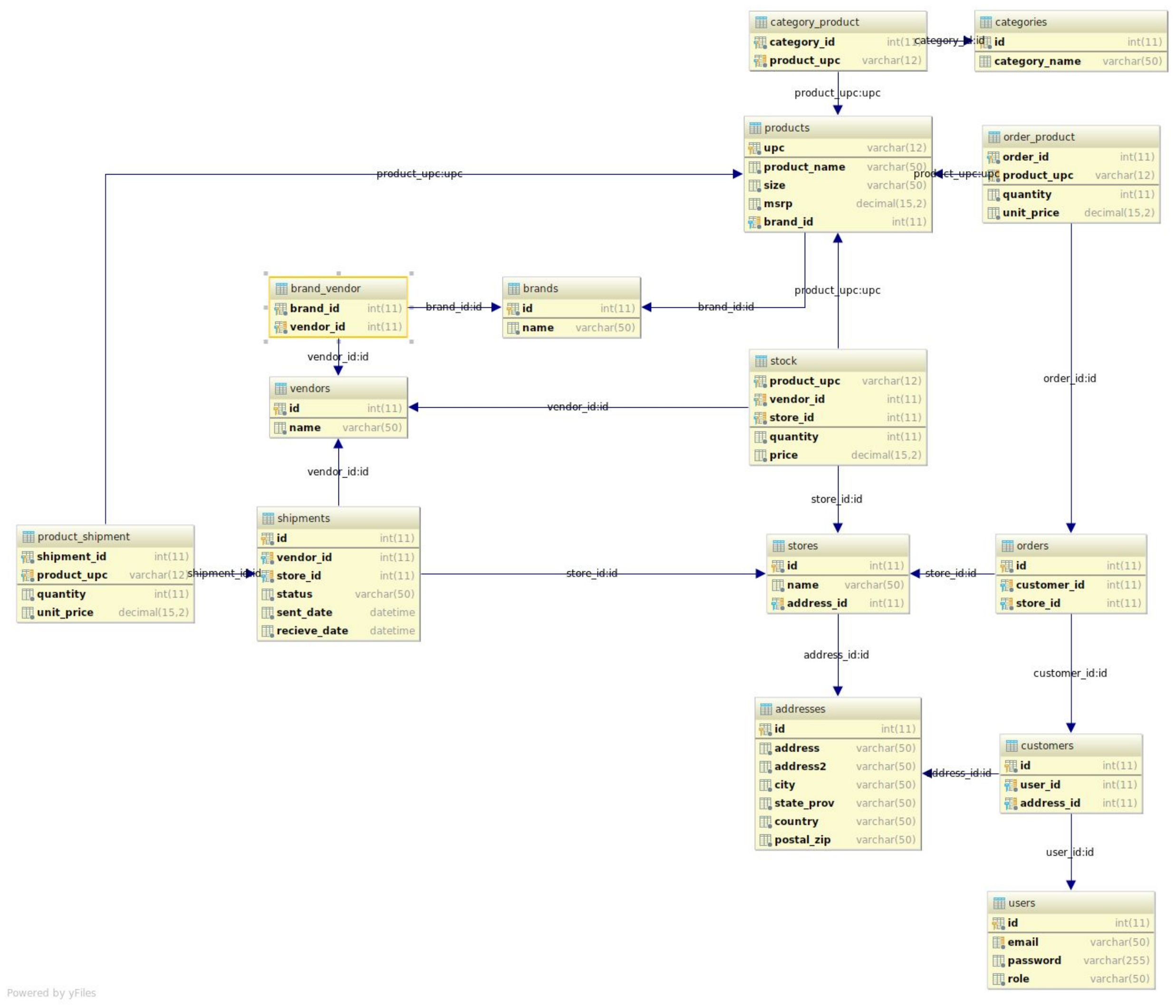
If an item is available when the customer first selects a store to shop at, they are able to add as many of that item as they wish regardless of the available stock at the store. The stock will then show as negative for on the Manager page.

A fully implemented database would also include separate manager and vendor accounts, for individual store orders, orders from an individual vendor, etc. When shipments are created by the manager or approved/denied by the vendor, the actual time would also be properly logged in the shipments table.

While the ER diagram shows a Categories entity and a Categories_product relation, and some entries were made to fill in this table, it was not used in the application. The intended use was to provide a central spot for product descriptions such as colour, country of manufacture, etc. that are not specific to one product. The categories table is easily expandable without requiring any change to the schema. It could be used on the customer shopping page to provide additional product information.

There are two prices available for each product, the MSRP which is included in the product table, and the store price. All prices listed in the application are the MSRP rather than the store price, though the store price was generated for the entries in the Stocks table.

E-R Diagram



category_product
category_id int(11)
product_upc varchar(12)
category_id:id
categories
id int(11)
category_name varchar(50)
product_upc:upc
products
upc varchar(12)
product_name varchar(50)
size varchar(50)
msrp decimal(15,2)
brand_id int(11)
product_upc:upc
order_product
order_id int(11)
product_upc varchar(12)
quantity int(11)
unit_price decimal(15,2)
product_upc:upc
brand_vendor
brand_id int(11)
vendor_id int(11)
brand_id:id
brands
id int(11)
name varchar(50)
brand_id:id
product_upc:upc
vendor_id:id
vendors
id int(11)
name varchar(50)
vendor_id:id
stock
product_upc varchar(12)
vendor_id int(11)
store_id int(11)
quantity int(11)
price decimal(15,2)
order_id:id
vendor_id:id
store_id:id
product_shipment
shipment_id int(11)
product_upc varchar(12)
quantity int(11)
unit_price decimal(15,2)
shipment_id:id
shipments
id int(11)
vendor_id int(11)
store_id int(11)
status varchar(50)
sent_date datetime
recieve_date datetime
store_id:id
stores
id int(11)
name varchar(50)
address_id int(11)
store_id:id
orders
id int(11)
customer_id int(11)
store_id int(11)
address_id:id
customer_id:id
addresses
id int(11)
address varchar(50)
address2 varchar(50)
city varchar(50)
state_prov varchar(50)
country varchar(50)
postal_zip varchar(50)
address_id:id
customers
id int(11)
user_id int(11)
address_id int(11)
user_id:id
users
id int(11)
email varchar(50)
password varchar(255)
role varchar(50)
Powered by yFiles