CST 363 Final Project

Moises Bernal, Aaron Gordon, Debajyoti Banerjee

Project Scope

This is a database schema for an online retailer, such as Amazon.com, that sells books, DVDs, CDs, games, furniture, and garden products. The database will be responsible for tracking customers (including the billing of customers upgraded to a premium account), customer payment methods, vendors, products, categories, and customer orders, including shipping address and total cost for the order.

Entities and Views

The database will track the following entities:

- **Customer**
  Every customer who uses the online store will be modeled as a Customer entity.

- **Premium Customer**
  Customers may subscribe to a premium account for a monthly fee. Premium Customers will be modeled as a separate Premium Customer entity with a foreign key back to the Customer table.

- **Address**
  Addresses will be used for both payment methods and shipping invoices. As such, to achieve normalization, address information will be modelled as an Address entity with a foreign key linking it to a specific Customer entity.

- **Payment**
  Payment methods (currently we only accept credit cards) will be modeled as a Payment entity with foreign keys linking it to both a Customer entity and an Address entity, used as a billing address.

- **Vendor**
  Vendors that distribute the products for sale will be modeled as a Vendor entity.

- **Product**
  Each product available for sale will be modeled as a Product entity with a foreign key back to the Vendor table.

- **Category**

Each category available sorted by subcategories (e.g., DVD is found in MEDIA which is in ROOT) with a foreign key ParentCategoryID back to the same table.

- **Invoice**
  An order that has been placed and ready to be shipped will be modeled as an Invoice entity with foreign keys linking it to a Customer entity, a Payment entity and an Address entity, used as a shipping address.

- **Line Item**
  Each invoice may be composed of multiple products, each of which will be modeled as a Line Item entity with a foreign key linking it to a specific Invoice entity.

In addition to the above entities, the following views will be exposed to assist with potentially common queries:

- **Available Books View**
  This view will list data for all books in stock (i.e., quantity in stock greater than 0).

- **Overdue Customer View**
  This view will list Premium Customers with a past-due balance.

Stakeholders

There are three primary stakeholders who will benefit from this database:

- **Inventory Managers**
  Employees responsible for managing the store's inventory will be able to add vendors, add products and update the quantity of products when vendor shipments arrive.

- **Customers**
  Customers will be able to browse available products, place orders, and have their previous orders - including payment method and shipping address - saved to the database.

- **Billing**
  Employees responsible for billing customers with premium accounts will be able to identify customers with past due payments and update customer credit when payment is received.

Use Cases

We now present five common use cases that the database supports.

1. **Customer Searches Products by Vendor**
   A customer wants to view all products supplied by a particular vendor (e.g., Pearson) sorted by category and product name both in ascending order.

2. **Inventory Manager Searches Low Stock Products**
   An inventory manager wants to view which products are low on stock (i.e., where quantity in stock is ten or less) sorted by quantity in stock in ascending order.

3. **Customer Searches for Available Books**
   A customer uses a storefront client to view records from the Available Books View sorted by price in ascending order.

4. **Customer Views Order Details**
   A customer wants to view details of an order, including product information for each line item ordered by line item in ascending order.

5. **Billing Sends out Overdue Payment Notices**
   An employee in billing uses the Overdue Customer view, joined with the Customer table, to retrieve the email addresses of all customers with an overdue balance

Triggers

The database will have the following triggers:

- **store.update_line_item**
  This trigger will adjust the invoice subtotal, tax amount, and total amount if the total price of a line item is updated.

- **store.delete_line_item**
  This trigger will adjust the invoice subtotal, tax amount, and total amount if a line item is deleted.

Interaction with Users

In addition to the use cases, we also provided a catalog website for customers:

- **index.html**
  This page has a link to the browse by vendor page (i.e., findProduct.php).

- **findProduct.php**
  This page displays all the distinct vendors from the vendor table.

- **vendorProducts.php**

This page displays all the products by the selected vendor.

ER Diagram