CIS 451/551 Final Project Fall 2021

name(s): Kelemen Szimonisz

project title: STP (Seize the Prints) Store

Connection information
port number: 3068
hostname: ix.cs.uoregon.edu
guest account username: guest
Guest account password: guest
database name: stp

project URL: http://ix.cs.uoregon.edu/~kelemens/finalproject/

Highlights:

Products and Materials

- Inventory Management: view all products and materials
- Add new products, remove existing products
- Convert materials into products
- Check what products can be produced with the current material inventory
- Check if any products or materials are out of stock
- Determine what materials are required to produce a specific product

Materials and Purchase Orders

- View Purchase Orders
- Create a new Purchase Order
- Receive inventory from a Purchase Order
- Remove Materials from the database

Profit Analysis

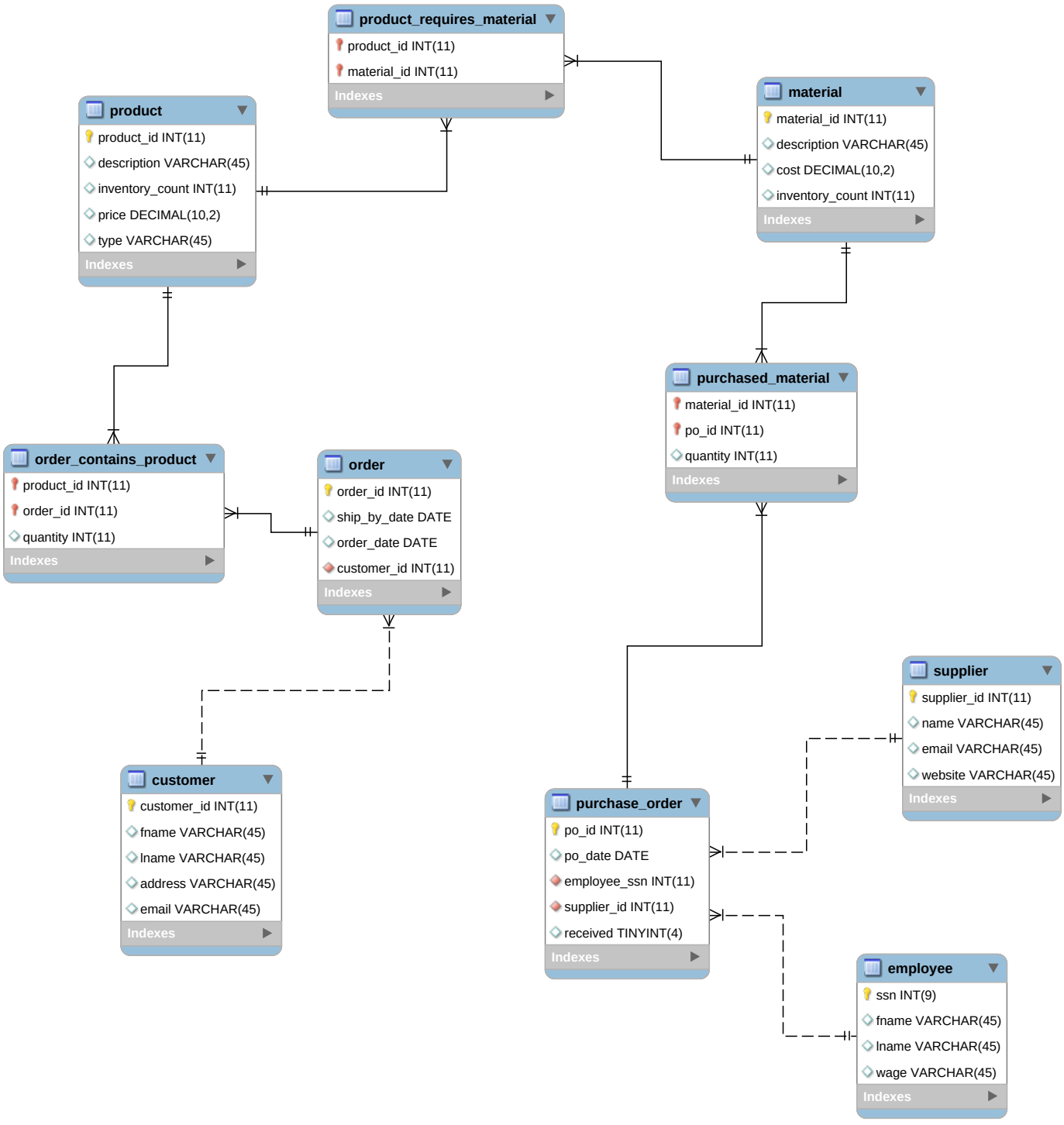
- Profit Per Product
- Profit Margin

Summary:

My friend, Stacy, operates a small business where she produces and sells arts and crafts. Stacy designs cards, stickers, t-shirts, and magnets, among other things. Each product requires different types and amounts of raw materials: vinyl paper, printer ink, cardstock, envelopes, blank t-shirts, etc. Stacy purchases these materials to produce her crafts and sells them for a profit. However, she has trouble keeping track of how much money she has spent on materials, how many products she has sold, and what her profit margins are for each product. I plan to design and implement a database that stores this type of data, as well as producing several applications that will help Stacy organize her business.

The main pieces of data that I plan to store are: the products that Stacy produces, the materials that each product is composed of, the orders that are requested by customers, and the purchase orders that are produced when materials are purchased by the business. Some of the application programs that I plan to create include: profit margin per product, total profit margin for all products, products that are out of stock, products that cannot be produced due to their required materials being out of stock, and converting materials stock into product stock.

Logical Design (Next Page)


product_requires_material
product_id INT(11)
material_id INT(11)
Indexes
product
product_id INT(11)
description VARCHAR(45)
inventory_count INT(11)
price DECIMAL(10,2)
type VARCHAR(45)
Indexes
material
material_id INT(11)
description VARCHAR(45)
cost DECIMAL(10,2)
inventory_count INT(11)
Indexes
purchased_material
material_id INT(11)
po_id INT(11)
quantity INT(11)
Indexes
order_contains_product
product_id INT(11)
order_id INT(11)
quantity INT(11)
Indexes
order
order_id INT(11)
ship_by_date DATE
order_date DATE
customer_id INT(11)
Indexes
customer
customer_id INT(11)
fname VARCHAR(45)
lname VARCHAR(45)
address VARCHAR(45)
email VARCHAR(45)
Indexes
supplier
supplier_id INT(11)
name VARCHAR(45)
email VARCHAR(45)
website VARCHAR(45)
Indexes
purchase_order
po_id INT(11)
po_date DATE
employee_ssn INT(11)
supplier_id INT(11)
received TINYINT(4)
Indexes
employee
ssn INT(9)
fname VARCHAR(45)
lname VARCHAR(45)
wage VARCHAR(45)
Indexes

Physical Design:

Table: employee

The employee table represents a worker at the STP store. The employee table uses a 9 digit (INT) SSN as a primary key. The fname and lname attributes (VARCHAR(45)) store the employee's first and last name respectively. Lastly, the wage attribute (INT) stores an employee's hourly wage. Employees can create purchase orders, which I will explain below.

Table: supplier

The supplier table represents an entity that STP purchases materials from. Each supplier is uniquely identified by a supplier_id (INT). The name, email, and website for the supplier are stored as well (VARCHAR(45)). Supplier's are referenced when an employee produces a purchase order.

Table: purchase order

The purchase order table represents a purchase order document. A PO is uniquely identified by a PO_id (INT). The date of the PO's creation is stored in the attribute PO_date (DATE). Two foreign keys are required for this table: employee_ssn and supplier_id. Employee_ssn refers to the employee that created the PO. Supplier_ssn refers to the supplier the employee purchased materials from. The received (TINYINT(4)) attribute represents whether or not the materials contained within that PO were received into inventory or not.

Table: material

The material table represents a raw material item that is required to produce one of STP's saleable products. Each material is uniquely identified by a material_id (INT). The description attribute (VARCHAR(45)) describes the material. The cost attribute (DECIMAL) represents the cost of that material. The inventory_count (INT) attribute represents how many instances of this material exist in stock.

Table: purchased material

The purchased material table is a bridge table between the **material** table and the **purchase_order** table. It represents a many-to-many relationship. A purchase order can contain many purchased materials. A material can be purchased many times. The primary key of this table consists of two foreign keys: material_id (from **material**) and po_id (from **purchase_order**). There is one non-primary attribute: quantity (INT); this attribute represents how much of this material was purchased.

Table: product

The product table represents a final product that is sold by STP to customers. A product is uniquely identified by the primary key product_id (INT). It has attributes description (VARCHAR(45)) to describe the product, inventory_count (INT) to reflect how much of this product is in stock, price (DECIMAL) to represent how much the product is sold for, and type (VARCHAR(45)) to organize what type of product it is (Card, Sticker, Magnet, or T-Shirt). Products are produced by combining required materials.

Table: product_requires_material

Products require materials to be constructed. The bridge table product_requires_material represents this many-to-many relationship. The table is uniquely identified by two foreign keys: product_id (from **product)** and material_id (from **material**). A product can require many materials. One material can be used to produce many different products.

Table: customer

The customer table represents a client of the STP store. Each customer is uniquely identified by a customer_id (INT). The first name and last name of a customer is stored in the attributes fname (VARCHAR(45)) and lname (VARCHAR(45)). The address and email may also be stored for each customer (both VARCHAR(45)). Customers can place **orders.**

Table: order

The order table represents an order that was placed by a **customer.** An order is uniquely identified by its order_id (INT). There are two (DATE) attributes that are stored: ship_by_date and order_date. Ship_by_date represents when the client is expecting to have their package in the mail. Order_date represents when the order was placed. The foreign key customer_id exists as an attribute, referring to the **customer** that placed the order. Each order contains one or more products.

Table: order_contains_product

An order may contain many products. One product may be ordered many times. This many-to-many relationship is represented by the bridge table order_contains_product. This entity is uniquely identified by the two foreign keys: order_id (from **order**) and product_id (from **product**). The attribute quantity (INT) also exists in this table and represents how much of the product was ordered.

List of Applications:

1. View all products and materials

- This application allows the user to view all products and materials that exist in the database. Data from the **product** and **material** tables are queried but not modified.
- Each query is printed on page load, and no interaction is required by the user.

2. Add new products, remove existing products

- This application allows the user to define a new product (or remove an existing one).
- Defining a new product: The product_id (primary key) is set to auto-increment, therefore the user does not need to input one themselves. Instead, the user is required to enter a product description, price, product type (from a selection of radio bubbles), and a comma-separated list of the material_id's required to create the product. This inserts a record into the **product** table. Also, for each material_id entered, a record is inserted into the **product_requires_material** table.
- Removing a product: The user can remove a record from the **product** table by entering a product_id. All records from **product_requires_material** that are dependent upon that product_id are also deleted.

3. Convert materials into products

- This application allows the user to convert **material** stock into **product** stock.
- The user must enter a product_id and quantity.
- If the product's required materials are in stock, then each required **material**'s inventory_count is decremented and the **product**'s inventory_count is incremented accordingly.
- If there is not enough materials in stock to create the desired product, the user is notified and there is no change to the database.

4. Check what products can be produced with the current material inventory

- This application allows the user to identify what products they can create given the current material stock.
- The query is printed as a table on each page load and no interaction is required by the user.

5. Check if any products or materials are out of stock

- This application allows the user to identify which products or materials are out of stock.
- Two tables are printed on each page load, one for OOS products and one for OOS materials. No interaction is required by the user.

6. Determine what materials are required to produce a specific product

- This application allows the user to enter a **product**'s product_id and receive a printed table of what **material**, and in what quantity, are required to make that product. The **product, product_requires_material,** and **material** tables are joined in this query. The database is not modified.

7. View Purchase Orders

- This application allows the user to view all purchase orders that exist in the database. The result of the query is printed on the page as a table upon each load. No interaction is required by the user.
- For some security, the employee_ssn is not used as a column, instead the employee's last name is used. The **purchase_order, employee, supplier, purchased_material, and material** tablets are joined in this query.

8. Create a new Purchase Order

- This application allows an employee to create a new purchase order.
- First, the employee must enter their SSN as an employee ID, the date the PO is created, and the name of the supplier.
- Second, the employee must decide whether or not they are purchasing a **material** that already exists in the database, or a **material** that has not been purchased before. The employee can select this using the respective radio buttons. Due to time constraints, the current implementation requires that an employee only purchase one material per PO.
- Existing material: the material_id and quantity of the purchased material are entered. A record is added to the **purchase_order** table and **purchased_material** table.
- New material: the material's description, cost, and quantity must be inputted by the employee. Upon submission, this inserts a new record into the **purchase_order** table, **material** table, and **purchased_material** table.

9. Receive inventory from a Purchase Order

- When a **purchase_order** is created, the received attribute is false, and the quantity of the purchased material is not yet added to inventory.
- This application allows an employee to enter a PO_id and change the received attribute from False to True. Once this is done, the quantity of the **purchased_material** belonging to that PO is added to the inventory_count attribute of the respective **material.**
- A record for both the **purchase_order** and **material** tables are modified to reflect these changes.

10. Remove Materials from the database

- This allows an employee to remove a record from the **material** table by entering a material_id.
- However, if a record in the **product_requires_material** table is dependent upon that **material's** material_id, then the material will not be removed and the user will receive a message.
- The employee would have to first remove all dependent products and then the material.

11. Profit Per Product

- This application joins the **product, product_requires_material,** and **material** tables to query the cost of materials for each product and profit for each product.
- The result of this query is printed on the page as a table on each page load and no interaction is required from the user.

12. Profit Margin

- This application joins the **product, product_requires_material,** and **material** tables to determine the profit margin for all of the products in the store (inventory count is not considered).
- The result of the query is always one column containing the Total Potential Revenue, Total Expenses, and Profit Margin %. It is printed on the page as a table on each page load and no interaction is required from the user.

User Guide:

Website Layout:

The homepage provides links to three main sections (pages) of the website:

- Products and Materials
- Materials and Purchase Orders
- Profit Analysis

Each of these pages contains multiple applications.

I described how to use each application in the previous section, so I will keep the user's guide fairly brief.

1. **Products and Materials Page**

An employee user would use this page for four main services: Adding new products to the database, removing existing products from the database, producing products from existing materials, and determining what materials are required to make a specific product.

Adding new products to the database:

Using the input form under the header "*Add a new product to the database:*", the user must input a description string (*Red T-shirt)***,** price (*4.99*), select a type (click on a radio button), and provide a string of comma-separated material_id's (*42,39,36*). After clicking submit, the tables will be updated and the page will refresh to reflect those changes.

Removing a product from the database:

Using the input form under the header "*Remove a product from the database:*", the user must enter a product_id (*205)*. After clicking submit, the page will refresh to reflect those changes.

Produce a product from existing materials:

To convert material inventory to product inventory (representing a production event), the user must enter a product_id (*205)* and quantity (*3*) and click submit. If the materials that are required to produce that product are in stock, then the page will refresh with the inventory count belonging to the entered product_id will be incremented by the entered quantity. The inventory count of each material used to produce the product will each be decremented by the entered quantity.

If the desired quantity for the entered product_id cannot be produced due to a lack of required materials, the user will be displayed a message and the tables will not be changed. To

see what materials are required to produce a product, the user can use the "what materials are required to make a specific product?" tool.

What materials are required to make a specific product:

The user may enter a product_id into the input form under the header "*what materials are required to make a specific product?"* and click submit. This will print a table displaying the materials needed to produce that product and the current inventory count for each material.

The results of this application allows the user to determine what products may need to be purchased and re-stocked. The user can purchase more materials by clicking the 'Homepage' link and travelling to the "Materials and Purchase Orders" page.

2. Materials and Purchase Orders Page

An employee user would use this page for four main services: creating a new purchase order containing existing material, creating a new purchase order containing new material, receiving inventory from existing purchase orders, and removing a material from the database.

Creating a new purchase order (existing material):

The user may create a purchase order by using the input field under the header "*New Purchase Order"* and entering their ssn (*123456789)*, the PO Date (*2021-12-06*), the supplier name (*Amazon)*, and selecting the "*Existing Material*" bubble.

The user must then enter the ID and quantity of the material they are purchasing. They can determine the material ID by looking at the table printed under the header "*Materials in Database"*. After clicking submit, the Purchase Orders table is updated.

Creating a new purchase order (new material):

The user may also create a purchase order for materials that do not exist in the materials table. After entering their ssn, PO Date, Supplier name, they must select the "*New Material"* bubble. Lastly, the user must enter a material description, cost, and quantity. After clicking submit, the page is refreshed and the user can see the newly created Material and Purchase Order printed in their respective tables.

Receive Inventory:

Creating a purchase order for a certain quantity of material does not automatically update the inventory count of the material. Instead, the user must receive inventory for that PO. The user can enter a PO_id in the input form located under the header "*Receive Inventory"*. The page will refresh and the item under the column "Received" for that PO is updated from "False" to "True". The inventory count for the material purchased in that PO is incremented by the quantity purchased and is reflected in the printed materials table.

3. **Profit Analysis Page**

An employee user would use this page for two main services: determining how profitable each individual product is and seeing the total profit margin for all products. No user interaction is required. The page is accessed by clicking on the "*Profit Analysis"* link found on the Homepage.

Contents of Tables:

The contents of each table is accessible via the Homepage by clicking on the link "Contents of Each Table".

You can also access it directly with this link:
http://ix.cs.uoregon.edu/~kelemens/finalproject/tableContents.php

Implementation Code:

All implementation code via this link: http://ix.cs.uoregon.edu/~kelemens/finalproject/txt_files/

Individual links to implementation code for a given page are located on the page itself.

Conclusion:

Overall, I have developed the basis of an inventory management system for a small business. If I had more time I would have implemented applications involving orders from customers. For example, if a customer were to place an order for a certain product that was out of stock it would be useful to see what products need to be restocked. Tracking the most popular products would also be a helpful application for the Profit Analysis page. Also, due to time constraints I only allow for employees to purchase one material per PO.

My mini-world is based off of a real-life small business that primarily uses Etsy to manage orders. Integrating some kind of connection between the database and Etsy's API could be really interesting and practical.