CSC 565 Final Project

Team 1

Charles J. Howard, Daniel de las Heras

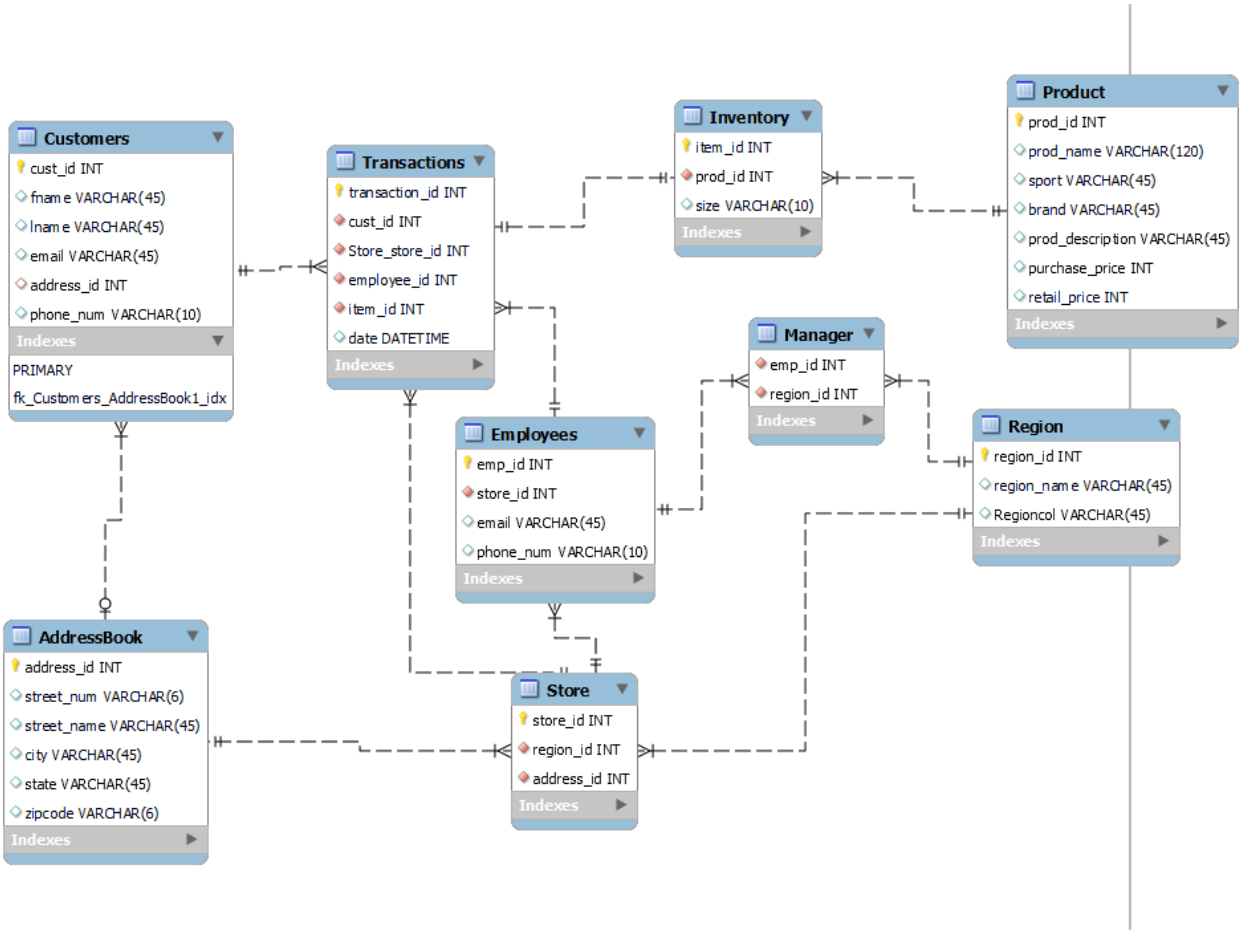
CSC-565 Database Management Systems

May 10, 2022

Our company is an athletic equipment company that helps fulfill the needs of athletes, students, and all kinds of different markets and sports. We offer equipment for a wide variety of sports, such as Baseball, Soccer, Hockey, to other activities like fishing, swimming, and hunting. Our retail stores are often recommended by notable athletes across the spectrum of competition in helping people find what they need to reach their highest potential in their sport.

In designing our database, we placed a focus on addresses, transactions, customer information, and the products they choose. Sports equipment comes in a wide range of sizes and brands for each sport. Each person has a preference on feel, color, brand, and sport, and so it's important to track which customers prefer which items and brands in which regions. Since we also serve businesses, the address book is normalized away from the customer and store entities, making it easier to update and change addresses. This allows the same address to be referenced by multiple different customers, like students, parents, and even small businesses run out of the homes of customers.

The Inventory entity tracks all items in our inventory, ranging from safety equipment such as helmets, for sports such as football, baseball, and hockey, to equipment for playing, such as balls, bats, gloves, and hockey sticks. We also sell gear for winter sports, and non-contact sports, such as weight lifting, swimming, and other such sports.



The Entity-Relationship diagram of our design is shown above. It features the following entities, a Customer, an Inventory, Products, Transactions, Employees, Manager, Stores, and Regions. The Customer entity contains an id, their first and last name, an email address and phone number, and their address. It represents the customer making the transaction. A transaction is a complete purchase by a customer. A transaction has an id, a seller id that references the Employee that made the sale, a store id referencing the location the transaction was made, and a customer id for referencing the customer. It also has a store id, which is where the transaction was made, and a sell date for when. The product entity references the items involved in the purchase. It has an id, a name, its purchase price, and the retail price for selling. It also has a sport attribute for categorizing the sport it belongs to.

The Inventory entity references the items in the transaction, chosen by the customer for purchase. This entity also has a foreign key to the product entity. For every transaction, there is a salesperson that processes it for the customer. An Employee entity has attributes for their name, address, email, and their store location. The store id signifies which store they are assigned to work at. The store entity has a store id, an address id, and a region id. The address id represents the store's address, in reference to the address book entity. The address book entity contains the street name, number, city, state, and zip code.

The Relational Schema for each entity is shown below. Primary Keys are underlined, Foreign keys are italicized.

Customer(cust_id, fname, name, email, *address_id*, phone_num)

Transaction(transaction_id, *cust_id, store_id, employee_id, item_id*)

AddressBook(address_id, street_num, street_name, city, state, zipcode)

Employees(emp_id, *store_id*, email, phone_num)

Store(store_id, *region_id, address_id*)

Inventory(item_id, *prod_id*, size)

Manager(emp_id, region_id)

Product(prod_id, prod_name, sport, brand, prod_description, purchase_price, retail_price)

Region(region_id, region_name, regioncol)

DDL Statements normalize the database by removing repeated values, and including cascading deletion and updates. There weren't many repeated values, as we took normalization into account when designing most of the database. One area where we did normalize was in the relationship between the product and inventory. Using foreign keys to reference the inventory table from the product relation in the transaction entity helped to prevent items from being lost in reference when removed. Originally we only had the product table, but realized that when items would be removed, there was no table to hold a general reference to them. We separated out the references to products in the transaction, to the inventory table. This way there is always a reference to those items.

The DDL statements are included in the following pastebin: https://pastebin.com/1VY4BGb8. The database is currently implemented in Daniel's database on the Compsci server.

We implemented several useful queries for the use of our database. One such procedure is "sports_profit". This procedure takes in a sports department, and shows the difference in costs and billing, showing the profits for that specific sport. Passing in Tennis shows Tennis sales profits, and can help determine which sports are becoming more or less popular. Another query is "search_by_brand", which takes in a brand such as Nike, and will show all products, their sport, their description, and their pricing. This will help customers and sales representatives show and find the products offered by specific brands through our store.

Another query we implemented is the "sports_sales" procedure. It takes in a sports department, and shows all transactions that were made featuring items of a specific sport department. Using Soccer as a parameter will show all transactions in which the items involved

have “Soccer” as their associated sport. One other query is the procedure “search_by_sport”, which allows searching the catalog by the name of the sport that a customer or sales rep is looking for. This shows its id, brand, description and pricing. Another useful query is the “list_of_purchases” procedure. This procedure shows all the transactions made by a customer at our stores, listing their items purchased, the pricing, and customer information. A very important query is the “find_profits” procedure, which returns the profits of the company. This is an important query as it shows the total profits of all departments, and how the company is doing overall. An equally important query is the “profits_by_sport”, saved as a view. This query shows the total profits gained by the sales made by each sports department. This query is important because it specifically shows how well each department is doing in relation to each other, and shows the top earners. This can show which sports could benefit from more advertising, and which to improve overall sales in.

Three SQL manipulation statements that were created to support the database’s management and use are the following procedures, “create_transaction”, “order_item”, and “change_manager”. The “create_transaction” procedure uses a transaction ID, employee ID, item ID, a customer ID, and a date to insert a transaction into the database, marking a purchase a customer made. “Order_item” inserts a new product into the database, to make available for customers to order. This item can then be referenced in a new transaction, or by customers that are searching the database for products. The “change_manager” procedure updates the region with a new employee id, replacing the old manager with a new employee. This can be useful for when managers retire or salespersons are promoted.

One issue we had was conflicting schedules. Other than that, we didn’t have many issues, as we had the same idea on what kind of retailer we wanted to run as a company. One limitation

of the system is that it does not differentiate between items that are useful in many sports. Sweatbands are used across basketball, tennis, soccer, and other activities, including general exercise. The limitation is that every item has to be associated with one specific sport. A sweatband can have tennis, or basketball as its sport, but not both. It might be more accurate to classify items in that category for the specific use, like “sweat removal/absorption.”. This would also improve customer experience, and help make searching even more useful to customers.