Dan Waters (danwaters@my.unt.edu)
CSCE 5350.002 | Fall 2022
Professor Dinayadura

Project: Online Musical Instrument Retailer

Project Report - "The Music Sea"

Dan Waters (danwaters@my.unt.edu)
CSCE 5350.002 | Fall 2022
Professor Dinayadura

Abstract

The Music Sea ("TMS") is a fictitious online musical instrument retailer whose only sales channel is their website. TMS offers guitars, basses, drums, pro audio equipment, keyboards, and other related gear across various departments. TMS manages their own distribution and shipping via a warehouse complex located in Nashville, Tennessee.

TMS maintains customer relationships via their staff of knowledgeable sales engineers, who help guide customers in their purchasing decisions and personally handle order fulfillment for every single order. Every order has a sales engineer attached, whether they spoke to the customer or not. Sales engineers advise a large number of customers.

Customers can also view & track orders via the web portal. Sales engineers periodically send advertisements and promotions to their customers directly, in order to drive demand for TMS's products.

This project aims to model a core subset of these relationships and features.

Scope and Requirements

Modeling and implementing an entire e-commerce retail business is well beyond the scope of this project. Thus, a subset of this domain has been chosen to model and implement for the project.

In scope:

- Items
- Departments
- Sales Engineers
- Customers
- Orders
- Order tracking

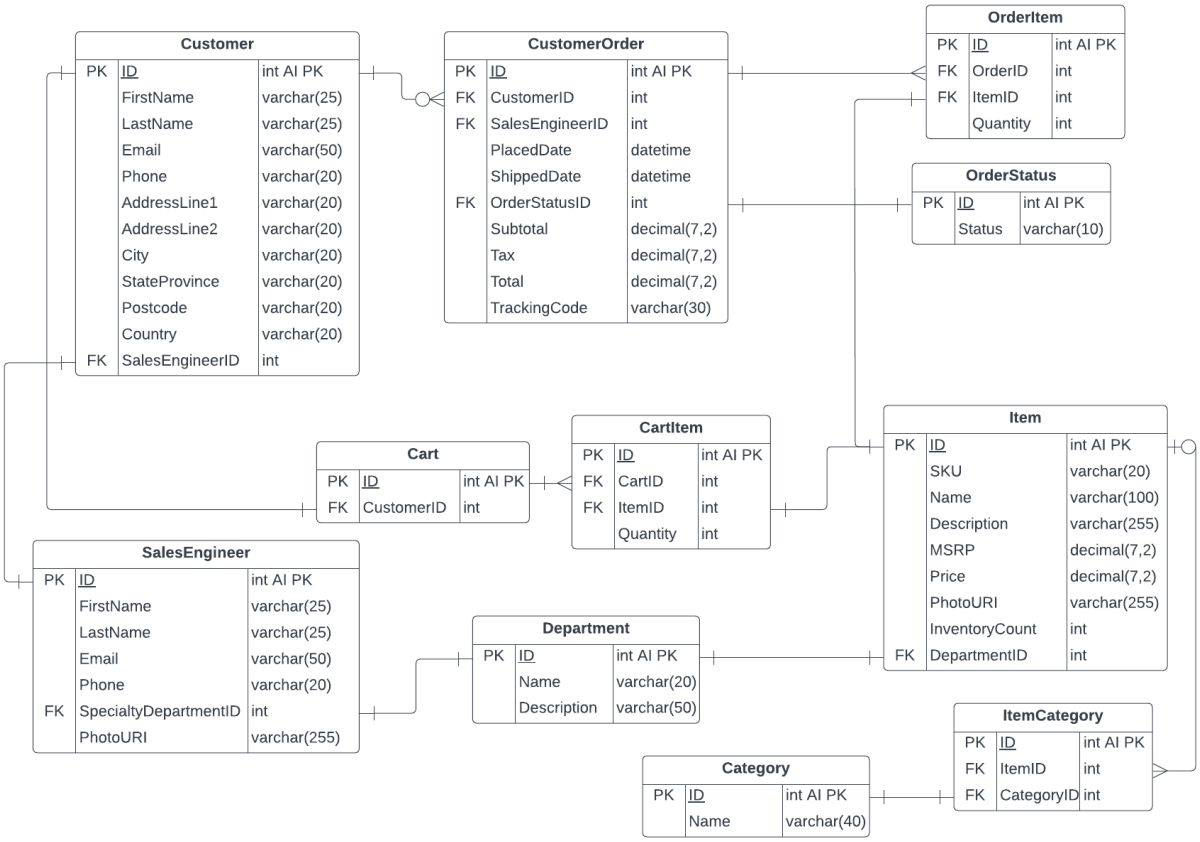
Out of scope

- Product reviews
- Payment gateway
- Customer login / authorization (project will be from the viewpoint of a single test user)

Schema changes since project proposal

Feedback was received that the number of entities was small. Through the course of implementation, some new entities have been identified, and relationships have been re-imagined. An updated ER diagram is below. You can also view this diagram in LucidChart here.

Dan Waters (danwaters@my.unt.edu)
CSCE 5350.002 | Fall 2022
Professor Dinayadura



Schema / cardinality justifications

- One Customer can have zero or more CustomerOrders associated with them.
- One Customer can have one Cart (this would be created when a user creates a new account). Technically, the system could support having multiple Carts (e.g. earlier abandoned carts that can be followed up on by the sales team), but the current implementation makes some assumptions in the business logic that Customer to Cart is 1:1.
- One Customer has one Sales Engineer, but a different Sales Engineer can facilitate a sale if the Customer's assigned Sales Engineer is on vacation or is unavailable.
- A Customer's Cart can have multiple Cart Items.
- Upon placing an order, a Customer's CartItems become OrderItems associated with an Order.
- A Cart Item is 1:1 with Item and OrderItem.
- An Item (e.g. product) can have multiple Categories (e.g. Guitar, Electric Guitar, 6-String Guitar). These are not added by the SQL, but can be added through the UI.
- ItemCategory is 1:1 with Category.
- A sales engineer specializes in one Department.
- An item is part of one Department.

Dan Waters (danwaters@my.unt.edu)
CSCE 5350.002 | Fall 2022
Professor Dinayadura

- A customer order has one OrderStatus.

Assumptions

- Since there is only one location (the warehouse), it is not needed to model employee hierarchies at multiple store locations, nor is it necessary to model or track inventory or sales across multiple locations.
- Billing address for customers is the same as the shipping address. No need to model this additional complexity. Payment gateway will be mocked anyway.
- Sales engineers are not assigned to a department; they can assist customers with any order, but there is a department where they specialize and are very knowledgeable about the products.

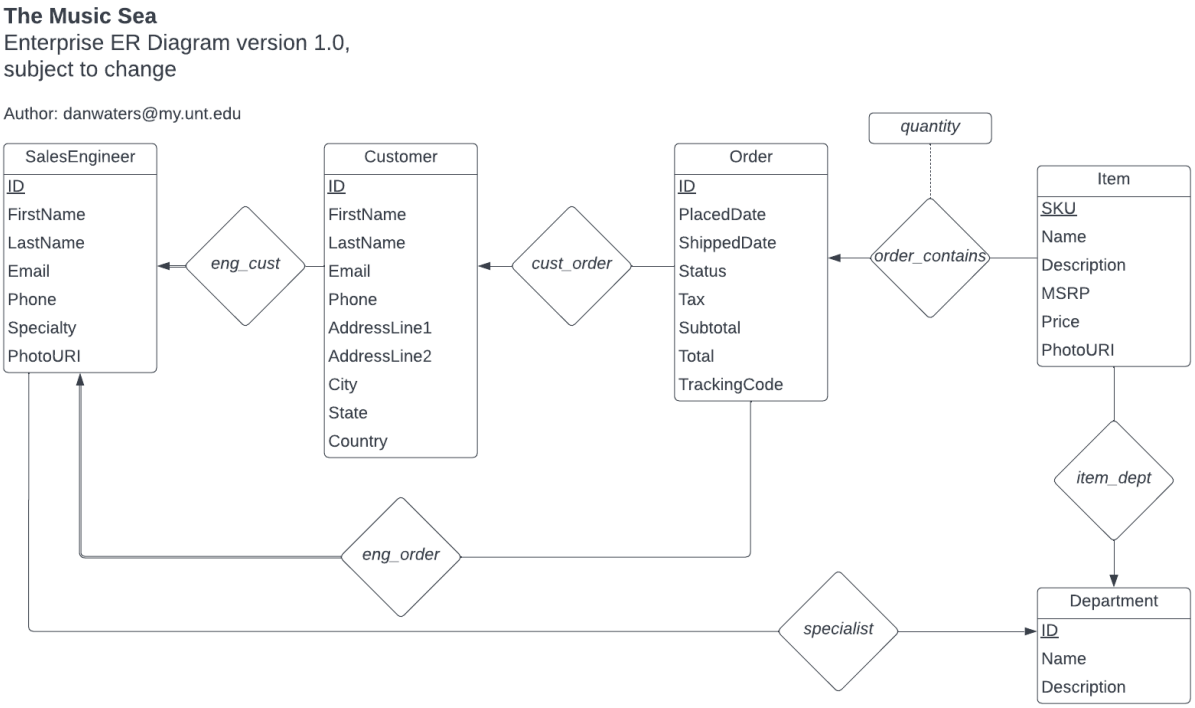
Tools

- MySQL, locally hosted
- Application: ASP.NET Core 6.0 using Razor Pages
- Modeling tool: LucidChart

Data Model

The updated, functional version of the ER diagram is here and shown above. The original conceptual version is shown below.

Dan Waters (danwaters@my.unt.edu)
CSCE 5350.002 | Fall 2022
Professor Dinayadura



Original ER diagram.

Dan Waters (danwaters@my.unt.edu)
CSCE 5350.002 | Fall 2022
Professor Dinayadura

Code Walkthrough

In the zip file / GitHub repo, there are a few key files to watch out for:

1. DDL.sql: creates the tables for the 11 entities from the ER diagram.
2. load_data.sql: Loads a bunch of test data.
3. DataService.cs: Contains all the SQL statements necessary for the application to function.
4. Entities folder: Serves as a light ORM layer for the underlying entities. Each entity implements a FromDataRow method enabling it to load data from MySQL.
5. MySqlService.cs: Light wrapper for MySQL to create tailored insert, update, read, and delete paradigms.

How to run the code

You will need:

- A MySQL instance with the DDL and load_data scripts already run.
- To modify the project's appsettings.json and appsettings.Development.json with the correct connection string for your environment.
- Visual Studio running on Windows with prerequisites installed for Web development with ASP.NET 6 Razor Pages. Entity framework is not required for this project.

UI Design Patterns

You'll see a folder called Pages which contains ASP.NET Razor Page items and their corresponding code-behind files. The solution leverages basic OnGet/OnPost action handlers with the occasional custom handler (for example, in the ViewCart page where a user can remove an item from their cart).

The solution attempts to follow best practices wherever possible in terms of page lifecycle management. Each database transaction opens and closes its own connection, which is not ideal for performance, but works for consistency and for this project. If such a project were to be scaled to production, every page request would have its own transaction scope via a DbContext rather than opening and closing connections for every query.

Bootstrap is the style framework used here.

Within Pages is an Administration section, which is unauthenticated (out of scope for this project). From the administration section, a user can modify many of the more static entities, change order status, and more.

Dan Waters (danwaters@my.unt.edu)
CSCE 5350.002 | Fall 2022
Professor Dinayadura

The default user is set to ID=1 (which is created from the DDL). Of course, in production, there would be multiple users with OAuth tokens stored on each client and a User ID readily accessible, but that is beyond the scope and material of this course.

Dan Waters (danwaters@my.unt.edu)
CSCE 5350.002 | Fall 2022
Professor Dinayadura

Future enhancements

Some things to think about for potential future enhancements:

- Implementation of a unified scheme for error handling and validation.
- Integration with a payment gateway.
- ETL for new products.
- CI/CD integration for a larger team.
- Automated tests for middleware and UI.