Database Design Document

Black Cats

Daniel Armenta
Aaron Aranda
Mualla Argin
Emily Hotz

Executive Summary

The purpose of this project is to create an efficient database for Layne's Restaurant. This system should help with sales and inventory for the restaurant. It will make for an easier experience for employees and managers of Layne's by simplifying the ordering process and automating behind-the-scenes inventory management.The system will have 5 entities. The inventory entity will keep track of each item used by the restaurant. This includes menu items and tableware items. This entity will let the user know how much of each item is on hand and help show when more items are needed. The menu entity will keep track of all menu items. It is the smaller umbrella of the inventory entity. It shows things like the price and calories of menu items. It also helps calculate trends and keep track of the overall inventory. The orders entity is where all order information is held. The things that were ordered are kept using menu IDs. The order entity keeps track of how many orders there are and which location they are being ordered at. It also shows employees the name of the person who ordered and shows when an order has been completed. This entity also helps calculate trends at different locations. The trends entity calculates how often menu items are being ordered. It then tells the inventory whether or not we should get more than usual of a certain item to account for the high demand. The location entity splits most of the data between two separate locations. It allows for two separate inventories to be watched and managed. Each entity in the database model will interconnect with one another to organize information so that it is easily understood when read by a user, and can be applied in other areas of the database. The inventory table manages listings based on item name, type, amount available in storage, ideal amount in storage, location, and expiration in order to keep managers up to date on supply and demand needs. The menu table tracks the restaurant's available menu options by name, price, calories, and inventory items required for creation so that details on a customer's order are clear to servers and kitchen workers. The order table tracks individual customer orders by customer name, date of order, menu items ordered, total price, and order completion, allowing servers to easily keep track of orders to be served and providing order history for future data analysis. The trends table tracks how many of each order type has been placed in a reasonable timeframe by taking information from the menu and orders tables, and synthesizes its own positive or negative recommendation value for the order based on recent customer trends. Each location in the location table keeps track of its respective orders, inventory entries, address, and amount of seating, separating the two current restaurant locations. Additionally, if the restaurant chain were to expand, it would be simple enough to add a new location to the table without disrupting the rest of the database. Measurable risks and assumptions have been taken into consideration in the development of this database design. For example, a key assumption we are making is that the data is clean and does not need to be parsed further or go through error handling. Therefore if the data is erroneous, our database will fail. We are also assuming that we are provided with starter data consisting of all attribute data. Hence, any missing data would be considered faulty input resulting in possibly faulty output. Finally, we expect that all users of our platform are familiar with the structure of the system and are equipped with technology to handle that. This would only be of concern if employees weren't given proper onboarding and training for the database.

Purpose

- **Problem Statement** - Layne's Restaurant needs a functional database. This should be a point of sale and inventory system for the business. It will need to be able to track orders, analyze trends, and update inventory needs. This database should be able to handle multiple types of items Layne's offers on their menu. It should also handle all aspects of the ordering process.
- **Target Audience** - The employees of Layne's Restaurant will be using this system to keep track of inventory, customer orders, and analyze trends. As such, the restaurant's managers will need to be able to check inventory and analyze trends in order to maximize restaurant efficiency, while its servers will need to be able to start and easily manage ongoing orders in order to keep up with customers' demands. To design this database for the whole target audience, its different aspects must match its respective users' needs.

- **Proposed Solution** - Our team will design a database housed on Amazon Web Services (AWS) through a TAMU engineering server. This database will show updated inventory as well as what items are being ordered most frequently. It will also house orders and order data. It will also provide order totals in a more efficient manner. The order data will be used to show trends of purchases and subtract totals from inventory.

High-Level Entity Design

Inventory

The inventory entity will keep track of each item to be stored and used by the restaurant, such as food, drinks, and tableware. This entity will mainly serve as an item availability check. If a manager needs to see how many items they have, then they will go to the inventory. It will also tell them if they need more items based on their availability. The inventory will tie into the menu by providing each menu item with the materials taken to make it. It will also tie into the location, as each location has one subset of the supplies filling the inventory. Lastly, it will tie into trends, providing the amounts of each item type available and expected for storage.

Menu

The menu entity will provide a list of available menu options which the customer can choose from, keeping track of what materials go into each item for sale. The menu will provide additional information about the items available to order, like price and calories. The database needs a separate table for items available to order, as while the inventory entity encapsulates all items in the restaurant, the menu entity narrows it down to make for a better order tracking experience. The menu will tie the inventory entity in order to keep track of the materials each order type contains. It will also tie into the orders entity by providing the separate menu items as possibilities for what may be ordered in an order. The menu also ties into trends in a similar manner, providing the possible orders a customer might request.

Orders

The orders entity keeps track of all orders made at a certain location. Each entry also indicates whether the order has been completed or not, allowing servers to easily manage active orders, while older orders can be kept in the records for analysis. The order entity will handle everything that occurs in the ordering process. It will account for easy and quick ordering, allowing servers to complete their jobs with more ease, while also helping automate all the background changes an order may cause in the database. Orders hold a set of menu items from the menu entity, showing what exactly was purchased by any one customer at a time. Orders also provide locations and trends with the information required to track what has been ordered in record.

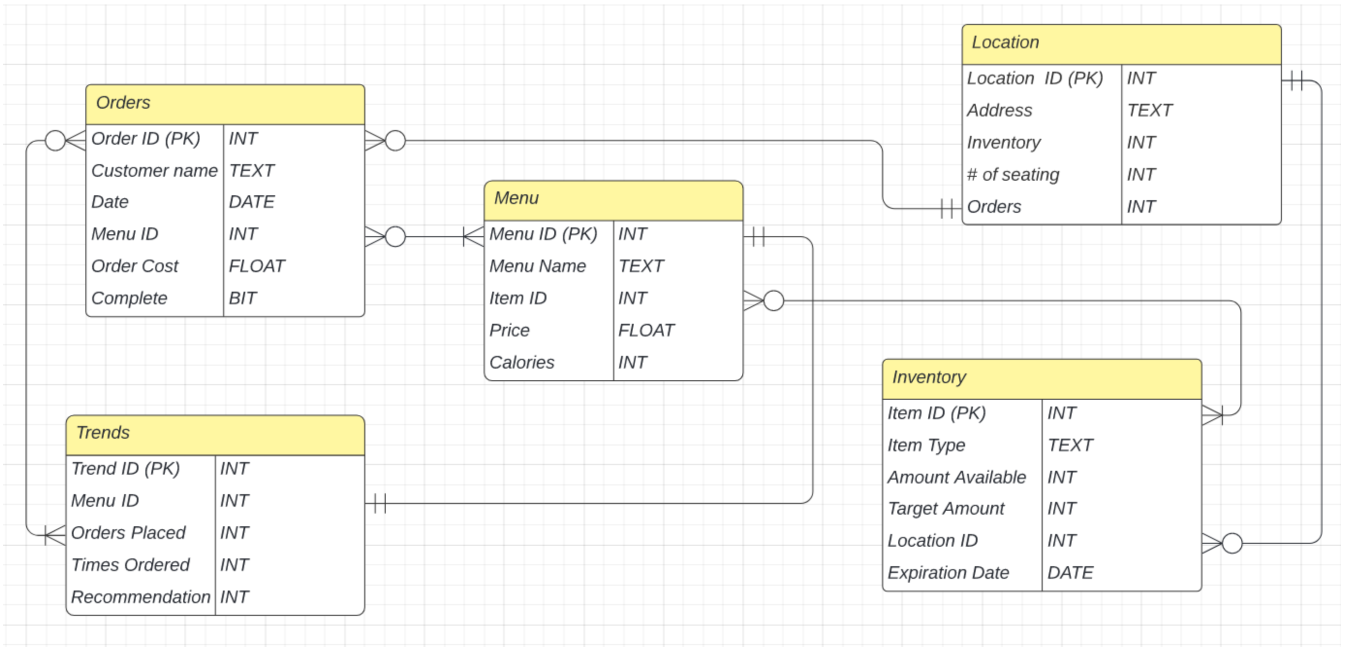
Trends

The trends entity allows managers to spot and analyze customer trends by pulling together useful information from the rest of the database and keeping it in one simpler location with relevant information. Through this analysis, Layne’s Restaurant can increase profit by altering the inventory contents accordingly. Additionally, the trends entity automates a few processes that managers would have had to spend long periods of time on if they had to keep up with new information by hand. The trends entity takes information from order history and menu item information to show how much each orderable item is in demand, and to recommend positive or negative changes in target storage values.

Location

The location entity exists so that data can be differentiated between the two different restaurant locations. Additional information on each location is provided.This entity allows for better item tracking and item availability based on the two different sites Layne’s owns. If there were no location entity, increases made in supply for items demanded in one location may be wasted in another. The location entity separates the inventory based on where each item is. It makes restocking items easier and more consistent. Each location also keeps track of its own respective order history. This also means that trends can be tracked based on where the orders were placed.

Low-Level Entity Design



This diagram includes 5 entities. Orders tracks where and when orders are being made and what menu items were ordered. Trends calculate how often certain items are ordered. Menu tracks the food, drink, and sides that are being ordered. Location splits everything into the two locations Layne's Restaurant owns. Inventory accounts for all the items the restaurant has on hand.

Entity Attributes

Each entry in the **inventory** will contain 7 different attributes. An item ID will be used for listing identification, and an item name for real-world object identification. An amount available will show how much of an item each listing contains in inventory. The target amount will show how much of an item should ideally be kept in storage at any given time based on trends and other data. A location ID will indicate where an item listing is stored. An expiration date indicates the date by which an item listing must be used up or disposed of. Lastly, a trend ID, if applicable, if an item listing is under scrutiny for at least one trend.

There are 5 attributes for the **menu** entity. A menu ID will be used for listing identification, and a menu name for real-world object identification. An item ID will be used to communicate with the inventory which items are being consumed by customers and need to be replenished. A price attribute will store the cost of specific deals & combos. A calories attribute will allow customers to make smarter choices about overall consumption.

For the **orders** entity, we have 5 attributes : Order ID, Customer Name, Date Ordered, Menu ID, and order cost. Order ID acts as the primary key for sorting orders. Hence, it can store only integers and all its values must be unique and non-null. The Customer Name allows employees to alert customers that their order is ready. Date & Time Ordered enables employees to queue and serve orders in chronological order. Menu ID ties into the menu entity to provide additional information like price and calories about specific items ordered. Finally, order cost will be used to calculate profits for the day.

The **trends** entity is composed of 4 attributes. Trend ID will be used for listing identification. Menu ID is added so trends can be grouped homogeneously and sorting is more efficient in the trend analysis phase. The orders placed attribute enables specific past-ordered menu items to be acquired. The times ordered attribute utilizes order data from orders placed to increment the count of a specific item ordered. The recommendation attribute tells whether more or less should be added into the inventory.

Within the **location** entity there are 5 attributes. Location ID acts as the primary key for sorting locations. The address attribute groups by address. The inventory ID returns the number of items remaining in the inventory. "# of seating" is populated with the number of people seated in each location so that the manager can determine whether a location should expand its capacity. Orders acts as a counter for the number of orders at a specific location.

Entity Relationships

The relationship between orders and location helps calculate how many orders are made at the different locations. Each order can be made at one and only one location. However, a location can have zero or many orders.

The relationship between inventory and location helps keep track of how many items we have left at each location. A location can have zero or many items in inventory. An item can only be housed at one and only one location.

The relationship between inventory and menu keeps track of which inventory items are included in which menu options. A menu option may contain one or more items from the inventory, while an item from the inventory may be included in zero or many menu items. For example, hand sanitizer would not be offered as a part of any menu item.

The relationship between menu and orders keeps track of how many menu items were ordered. An order must contain at least one menu item but can also contain many. A menu item can be ordered zero or many times.

The relationship between menu and trends serves to track trends relating to each item offered in the menu. A trend may only track one menu item, while each menu item belongs to only one trend.

The relationship between orders and trends helps track how often items are ordered and helps determine favorite items. An order helps calculate one or many trends. A trend takes into account zero or many orders containing a specific menu item.

Interactions

The main interaction is how the inventory is updated when someone orders at the restaurant. The order entity will apply an Order ID to the order. It will track where the order was made. It will also state what items were ordered. These items will then translate into menu IDs. The menu entity will see what items were used to make the menu item ordered. It will then translate to the inventory entity using an item ID. The inventory will then subtract from that item's available attribute. This is all done in each location's respective inventory.

Another interaction is how the trends are calculated. When someone makes an order, the order entity creates an order ID that is used in the trend entity. The trend entity tracks trends for certain menu items that it gets using the menu ID from the menu entity. The order IDs that are brought in are then broken down to find how much of the menu item was ordered. It is then counted using the times ordered attribute. Based on how many times, and how often the menu item was ordered, we calculate a trend for the menu item. This is put into the recommendation attribute. A negative number meaning “we need less” and a positive number meaning “we need more”. This is taken into account when refilling inventory.

We also implemented a location entity that accounts for the two locations of Layne’s. When a menu item is ordered, the order ID is shared with the location entity. This is to help track trends based on location. If the Texas location orders more toast than the Wellborn location, then the Texas location should have more inventory for toast than the Wellborn location. The location entity also directly interacts with the inventory entity to tell how much each location has on hand.

Assumptions and Risks

Assumptions:
We are assuming that all data entered is clean and does not need to be parsed further or go through error handling. Another assumption that we are making is that we are provided with starter data consisting of all attribute data. We expect that all users of our platform are familiar with the structure of the system and are equipped with technology to handle that.

Risks:

The data could be erroneous. This might cause the system to fail. The database might be accounting for more than it needs to. This will mean the structure of the database system will have to be changed around. The way relationships were formed between entities might not meet the needs of the company.