

**PROJECT REPORT**

**INST733: Database Design**

Bar Inventory management

at

Mulligan’s Grill & Pub

**Submitted by:**

**Janhavi Chitale**

**Kanishka Ramamoorthy**

**Akshata Salehittal**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Topic** | **Page** |
| 1. | Introduction | 2 |
| 2. | Design Changes | 3 |
| 3. | Lessons Learned | 3 |
| 4. | Future Improvements | 3 |
| 5. | Logical Design | 3 |
| 6. | Physical Design | 4 |
| 7. | Sample Data | 4 |
| 8. | Queries | 5 |
| 9. | User Interaction | 7 |
| 10. | Appendix | 9 |

**INTRODUCTION**

Effective management of information lays the foundation for growth in any organization. Data needs to be efficiently organized and stored in a way that provides easy access to the end users. Organizations should adeptly handle their inventory to make the best possible use of their available resources. Manually recording stock availability can be very time consuming and can result in reduced efficiency. A relational database model can solve this problem and improve productivity through automation of the manual process.

Our project aims at building a relational database model to help our client, *Mulligan’s Grill and Pub*, manage and organize their Bar Inventory. Through a deep understanding of user requirements, the database was designed to adeptly perform the task of interpreting and formally organizing gathered data to aid the management make better business decisions. This will help our client manage important information more efficiently as compared to their current practice of using excel sheets.

The major concern raised by our client was keeping track of available drinks in order to record stock differences. Our database serves this purpose by enabling accurate calculations of differences in actual and theoretical stocks to identify missing amounts and to monitor over-pouring of drinks. The system will also relieve the staff from the task of checking alcohol availability manually, thus saving a lot of time and effort. As an added advantage the database will be capable of querying information about the drinks in highest demand. The management can use this information to identify current trends in sales and manage their inventory according to changing consumer demands.

**DESIGN CHANGES**

A comparison of the old and new ERD reveals that the old ERD did not contain any of the ‘Tab’ entities, as we intended to obtain alcohol sales information from the point of sale system. However, we did not have access to this information so significant changes were made to the ERD to include sales details in the form of Tabs opened by the customer.

**LESSONS LEARNED**

One of the biggest challenges we faced through the course of the project was building a model to incorporate the variations in the nature of cocktail components. Calculating stock balance is a primary part of our database model. However, we did not have a standard method to calculate the sold quantities of cocktails which contained only a drink category and not a specific drink itself. The TAB\_DRINKS\_COMPONENTS entity paved a solution to this problem as it stores the specific drink that goes into the cocktail according to customer preference. Information from this table was used to calculate the available stock of drinks used in cocktails.

**FUTURE IMPROVEMENTS**

In the future, we wish to expand our model further to include detailed supplier and customer information. We would also like to devise methods to calculate cost variance by tracking the cost price of drinks bought in a particular order and comparing those values with the price at which they are sold at the bar.

**LOGICAL DESIGN**

The preliminary step of our project was to build the ERD model through thorough understanding of user requirements. After understanding the workflow at Mulligans, we were able to design 11 entities that primarily contained details about the drinks sold at the bar, their sales information and purchase details. The entities are described in detail in the appendix.

**PHYSICAL DESIGN**

The first step in the physical design was to create an Entity Relationship Diagram in *MySQL Workbench* using the logical design as reference. The ERD included all entities and the relationships between them. It also included the data types of all the attributes along with specifications like primary keys, foreign keys, required fields and auto increment values. The cardinalities were included as to have a clear representation of the modelling of the data. In the next step, we used the forward engineering function in workbench to create a schema based on our ERD.

**SAMPLE DATA**

The sample data contains actual as well as made up values. We were able to gain access to the information about the complete list of drinks sold at the bar along with the units in which they are available. We also received details about the approximate prices at which drinks are purchased from the sellers and the prices at which they are sold at the bar. All other data about the purchase and sales of alcohol were made up. This includes details about the purchase date and the quantities of alcohol purchased in any particular order, including the units in which they were purchased. Similarly, the sales data, which includes the quantities and cost of drinks sold on any particular day was made up. We had detailed data about the cocktail recipes including the percentage of each alcohol component used in the cocktails, to help us accurately calculate the stock variance and manage the inventory.

**QUERIES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Query Name** | **Requirement A** | **Requirement B** | **Requirement C** | **Requirement D** |
| **Query 1** | **x** | **x** | **x** | **x** |
| **Query 2** | **x** | **x** |  | **x** |
| **Query 3** | **x** | **x** | **x** | **x** |
| **Query 4** | **x** | **x** | **x** |  |
| **Query 5** | **x** | **x** | **x** | **x** |
| **Query 6** | **x** | **x** |  |  |
| **Query 7** | **x** | **x** |  |  |
| **Query 8** | **x** | **x** |  | **x** |
| **View 1** | **x** | **x** | **x** |  |
| **View 2** | **x** | **x** | **x** |  |
| **View 3** | **x** |  |  |  |
| **View 4** | **x** |  | **x** |  |

**Query 1**

*Display the most sold alcohol on a selected date along with the quantity and unit at which it is sold.*

The tab details are displayed date-wise. Using this information, the query returns the alcohol that was sold the most on that particular day. The management can use this information to determine which drink must always be stocked up in their inventory.

**Query 2**

*Display a list of all cocktails whose components have a particular drink category. [Eg 'Vodka']*

This query returns a list of all the cocktails which have Vodka as one of their components. The query can be modified to check for other drink categories as well. When a particular alcohol that goes into a cocktail runs out of stock, this query will help Mulligans identify such cocktails and not sell them for that day.

**Query 3**

*Display a list of all drinks purchased between two dates along with the quantity purchased.*

This query helps the management identify changing trends in alcohol demand based on any particular days or the week. The management can use this information to set discounts and happy hour prices and increase their profits.

**Query 4**

*Calculate total price for Tab\_view with taxes for any displayed Tab ID.*

This query will run at the backend whenever a tab is closed to calculate the total amount to be printed on the check for that particular TAB ID.

**Query 5**

*Display revenue generated for all the Drink Categories on a particular day.*

This query can help the management keep track of the drink categories that generate the highest revenue.

**Query 6**

*Display all the units and prices in which a particular alcohol is sold. Eg: Miller Lite*

This query informs the bartender about the different units in which a particular alcohol is supposed to be sold and displays the price of the unit.

**Query 7**

*Display Stock Report for a particular Date.*

The stock report contains details of all the drinks currently present in the inventory. It includes the opening balance, total purchase quantities and total sale quantities in ounces for each alcohol on any particular date. The ‘Theoretical Closing Balance’ is calculated as the difference between the ‘Opening Balance’ and ‘Total Sales’, which becomes the ‘Opening Balance for the next day’. The ‘Actual Closing Balance’ is fetched from the data inserted in the inventory table. The difference between the Theoretical Closing Balance’ and ‘Actual Closing Balance’ shows the missing quantities of alcohols.

**Query 8**

*Display the list of drinks sold during happy hour.*

This query helps the bartender let customers know which drinks are sold during the happy hour.

**USER INTERACTION**

Mockups were designed for data input and data retrieval to better understand the flow of the database. This also helped in modeling our database keeping the user’s perspective (in our case Mulligan’s management) in mind.

1. **Add New Drink**

This mockup was designed as an input screen for the user to add a new drink to the database. The user can select the category of the drink from the predefined drink categories. Users can also assign different units associated with the drink along with selling prices and happy hour availability for each unit. This form will run insert queries on the backend to insert data into the Drink and Drink\_Details tables.

1. **Add Purchase Details**

This mockup includes the interface to add purchase details into the database. The data entered in the form will be stored in Drink\_Purchase and Drink\_Purchase\_Details tables. The form lets us input a new purchase order; all the drinks purchased in that order, their units and their respective quantities. It also provides provisions to include the date of the purchase, total discount and total tax.

1. **Tab**

This mockup provides an overview of how the tab part of the database works from the front end. This includes data input as well as data retrieval queries. The first form displays the categories of all the drinks sold at the bar. When a category is selected, all the drinks listed in that category will be displayed. In case a cocktail is selected as a category, the subsequent form will display the categories of all the cocktail components (cocktail with unspecified drink) or the form will display the specific drinks that form the cocktail (cocktail with specified drink). The ‘Display Open Tabs’ button runs retrieval queries at the backend to output the list of all tabs that remain open at that time.

1. **Add to Inventory**

This mockup is designed to be used at the end of each day to input the actual closing balance in the database. The staff need to input the available stock values of the drinks in the text box against the drink name. This data will be inserted into the Inventory table and then be used to generate the stock report.

1. **Stock Report**

This mockup is designed as a front end to the Stock Report query. The user can select any day for which actual closing balance is already inserted into the Inventory table and generate a stock report. This form will thus give a brief summary of the transactions and the stock for a particular day.

**APPENDIX**

This appendix describes all the entities of the logical design in detail.

1. **Drink**

This entity contains the entire list of alcohol and cocktails sold at the bar. Each drink is identified by a unique name and “Drink\_ID”, which acts as the primary key for this entity. The attribute “Drink\_Category\_ID” specifies the category the drink belongs to and acts as a foreign key to the “Drink\_Category” entity.

1. **Cocktail Component**

This entity displays the list of alcohol components used in each cocktail. If the “Specified\_Drink” bit is set to 0 the “Component\_ID” is derived from the DRINK\_CATEGORY entity. If the bit is set to 1 the ID of the specific drink is derived from the DRINK entity. (e.g. Long Island Ice Tea contains categories tequila, vodka, light rum, triple sec, gin as its components while Irish Ginger contains a specific drink ginger ale as it’s component). The attribute “Quantity” will contain the weight of each alcohol component in the cocktail in ounces.

1. **Drink Category**

This entity groups drinks according to their categories. The “Drink\_Category\_Name” attribute contains the following categories: Tequila, Vodka, Gin, Rum, Bourbon, Scotch, Whiskey, Rail Cordials and Schnapps, Cognac & Brandy, Premium Cordials, Mixers, White Wine, Red Wine, Champagne and Turn.

1. **Drink Unit**

The drinks are purchased and sold in different units at Mulligan’s. They need to be represented in a common base unit to enable us to calculate the difference between purchase and sales cost. The “Unit\_Name” attribute contains a list of all the different units (e.g. Cans, Bottles, etc.) Each unit has a predefined base value in ounces.

1. **Drink Details**

This entity displays a list of all the drinks along with the various units in which each drink is sold at the bar. “Unit\_Price” displays the price at which a particular unit of the drink is sold at the bar. (eg: 1 bottle of miller lite costs 2$ and 1 can of miller lite costs 3$). “Happy\_Hour\_Availability” bit specifies if the drink is available at Happy hour or not.

1. **Drink Purchase and Drink Purchase Details**

This entity contains details about the purchase date and total bill amount of any particular purchase order. DRINK\_PURCHASE\_DETAILS entity displays the list of all drinks bought in a particular order along with their quantities. “Unit\_Price” displays the cost price of the specified unit of the drink.

1. **Tab, Tab Drinks and Tab Drink Components**

Every time a customer orders a drink at the bar the entries are made to a new tab or to an existing tab already opened by that particular customer. The Tab entities aim at replicating this process in our database. The TAB entity displays the open and close time of any particular tab along with the total bill amount. TAB\_DRINKS entity contains detailed information about all the drinks ordered under a particular Tab\_ID. TAB\_DRINKS\_COMPONENTS entity comes into play only when a cocktail is being ordered. Most of the cocktails contain only the drink category as their component and do not have specific drink components, hence do not have a fixed selling price. “Component\_ID” attribute in this entity contains the IDs of the specific drinks used in the cocktail according to customer preference. This aids in the calculation of selling price of cocktails by aggregating the selling prices of each specific drink used in the cocktail based on the quantities in which they were used.

1. **Inventory**

This entity helps the inventory management process by keeping track of stock balances. The “Actual\_Closing\_Balance” values are manually keyed into the database at the end of each day. The theoretical closing balance for any particular day can be calculated using the formula T = Opening Balance + Total Purchase - Total Sale

The Quantity difference in actual and theoretical values will help the management identify any missing products in the inventory and help maintain control on over-pouring of drinks.