Matthew Pascual-Mead

CPSC-408-01

Prof. German

5/17/21

*Tilly’s Database Application*

**Overview**

During the summer following my freshman year in college, I worked as a sales associate for Tilly’s, an American retail clothing company. After working a few shifts, I immediately realized that all the sales I had generated from assisting customers were often going unnoticed. I was putting in an immense amount of effort to generate sales for the store and received little to no recognition. I soon came to realize that this lack of acknowledgment was because of one glaring problem: Tilly’s had no way of tracking the sales of each of their employees. Because there was no system to track employee sales, managers were forced to gauge employee performance solely on their observations and without the support of any data. As a result, working at Tilly’s became very uninspiring. Many employees, including myself, felt there was no point in putting in the extra work to produce sales if these efforts would go unrecognized. This, in turn, led to an unhealthy work environment where employees dreaded going to work, and when they were working, they just wanted to get by doing the bare minimum. So to help solve this problem, I felt that it would be a great idea to design an application that could effectively track employee sales, allowing managers to track performance more reliably and efficiently. Furthermore, since the data being collected for this application goes beyond what is needed to record employee performance, this application would also add the extra functionality to provide insightful reports to support all levels of management. Ideally, implementing this application would hopefully increase employee satisfaction with their jobs while also helping Tilly’s improve as a corporation.

**Related Work/Applications**

In terms of related work, many retail companies track employee sales within their database systems. However, even for the stores that track employee sales, this information is often overlooked, unused, or even unreliable. As a result, the Tilly’s Database Application would be an excellent solution for the company because it offers insightful reports to all levels of management and is also very easy to use. Outside of retail clothing companies, there are various tools for tracking sales performance, such as Xactlty, Anaplan, and Varicent. But having access to the Tilly’s Database Application would be ideal for Tilly’s because it is already geared towards their business, is easily adjustable to fit their specific needs, and gives them complete control over their data.

**Application Framework**

The frontend UI of the Tilly’s Database Application is created in python. For the backend database component, the python application is connected to a MySQL instance in the cloud (Google Cloud Platform). Packages used to create this application include mysql-connector-python, pandas, re, and pyfiglet.

**Database Design**

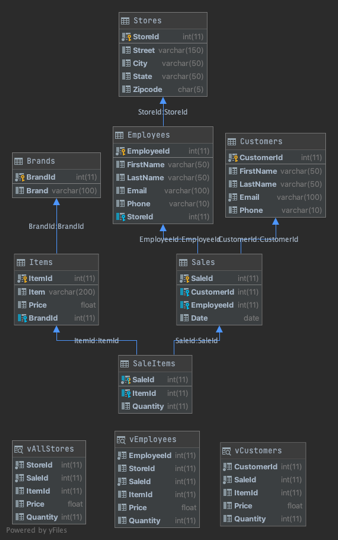
For reference, a visualization for the database schema is provided in the appendix (Figure 1.0). As an overview of the database design, this application consists of a total of seven tables: ‘Stores’, ‘Employees’, ‘Customers’, ‘Sales’, ‘SaleItems’, ‘Items’, and ‘Brands’. All foreign keys in these tables utilize cascading updates and deletes to ensure that all changes within the parent tables are also applied to the referenced records in the child tables. To start, the ‘Stores’ table contains Tilly’s store data and consists of five columns: ‘StoreId’, ‘Street’, ‘City’, ‘State’, and ‘Zipcode’. For this particular table, ‘StoreId’ is set as the primary key. Next, the ‘Employees’ table stores employee data and consists of six columns: ‘EmployeeId’, ‘FirstName’, ‘LastName’, ‘Email’, ‘Phone’, and ‘StoreId’. The primary key for this table is ‘EmployeeId’ and the foreign key is ‘StoreId’, which refers to the primary key in the ‘Stores’ table. In addition, an index labeled as ‘StoreId\_index’ built from the ‘StoreId’ column was implemented into this table to boost search performance for queries utilizing store IDs. The following table, ‘Customers’, stores customer data and consists of five columns: ‘CustomerId’, ‘FirstName’, ‘LastName’, ‘Email’, and ‘Phone’. Next, the ‘Sales’ table stores sale data and consists of four columns: ‘SaleId’, ‘CustomerId’, ‘EmployeeId’, and ‘Date’. For this table, the primary key is ‘SaleId’ and the foreign keys are ‘CustomerId’ and ‘EmployeeId’, which refer to the primary keys in the ‘Customers’ and ‘Employees’ tables. Furthermore, indexes labeled as ‘CustomerId\_index’ and ‘EmployeeId\_index’ built from the ‘CustomerId’ and ‘EmployeeId’ columns were implemented into this table as well. Once again, the purpose of these indexes were to boost search performance. Next, the ‘Brand’ table stores brand data and contains two columns: ‘BrandId’ and ‘Brand’. The primary key for this table is ‘BrandId’. The next table, ‘Items’, stores data about store items and contains four columns: ‘ItemId’, ‘Item’, ‘Price’, and ‘BrandId’. The primary key for this table is ‘ItemId’, and the foreign key is ‘BrandId’, which refers to the primary key in the ‘Brands’ table. This table also contains an index labeled ‘BrandId\_index’ built from ‘BrandId’ with the purpose of improving search performance. Lastly, the ‘SaleItems’ table stores data describing the contents of each sale and contains a total of three columns: ‘SaleId’, ‘ItemId’, and ‘Quantity’. The primary key for this table is ‘SaleId’ and the foreign keys for this table are ‘SaleId’ and ‘ItemId’, which refer to the primary keys in the ‘Sales’ and ‘Items’ tables. Furthermore, indexes labeled as ‘ItemId\_index’ and ‘SaleId\_index’ built from the ‘ItemId’ and ‘SaleId’ columns were implemented into this table. As an additional feature, a total of three views were implemented into this database as well: ‘vAllStores’, ‘vEmployees’, and ‘vCustomers’. The primary intention of the ‘vAllStores’ view is to simplify complex queries being performed for store reports within the application. The ‘vEmployees’ and ‘vCustomers’ were created in the case that we wanted to conceal sensitive data in the ‘Customers’ and ‘Employees’ tables. Columns in the original tables excluded from these views include ‘FirstName’, ‘LastName’, ‘Email’, and ‘Phone’.

**Application Results and Features**

The Tilly’s Application Database offers a wide range of features that the user can perform. The application allows users to display records, delete records, update records, create new records, and generate data reports (Figure 2.0). For displaying records, the user is given the option to display data from all tables (Figure 3.0). For deleting records, the user is given the option to delete data by ID from all tables except for the ‘SaleItems’ table (Figure 4.0). I exclude the option to delete from the ‘SaleItems’ table as it is somewhat of an illogical operation. Instead, I believe it is more desirable to have complete data providing the contents for every sale. As a result, I utilize a cascade on delete constraint so that if an entry in ‘Sales’ is deleted, it is also deleted in ‘SaleItems’. I would also like to note that deletes for this database are not soft deletes, however I would like to implement soft deletes in the future as it is ideal to have access to deleted records. For updating records, the user is given the option to update data by ID from all tables (Figure 5.0). All updates also implement a rollback feature to reject transactions if they are incomplete. Furthermore, for each update feature, the user is given the option to either update the entire record or specific attributes. For creating records, the user can insert data into any table (Figure 6.0). Like the update feature, all inserts implement a rollback feature to reject transactions if they are incomplete. For generating data reports, the user is given the option to choose between ‘Company Reports’, ‘Store Reports’, ‘Employee Reports’, ‘Customer Reports’, ‘Brand Reports’, and ‘Item Reports’ (Figure 7.0). Within ‘Company Reports’, users can execute queries that return information such as total revenue, top five revenues by state, total number of employees, and the total number of stores. Within ‘Store Reports’, users can execute queries that display the top five stores with the highest revenues and the five stores with the lowest revenues. In ‘Employee Reports’, users can access a wide variety of insightful reports. These reports include the top five performing employees, the bottom five performing employees, top employees by state, employees above a user specified sales amount, employees below a user specified sales amount, employee sales by store, and the option to search for employee sales by ID. Within ‘Customer Reports’, users can generate a report on Tilly’s top five highest paying customers. For ‘Brand Reports’, users are able to access information such as Tilly’s five most popular brands, their five least popular brands, their five highest revenue brands, and their five lowest revenue brands. Lastly, within ‘Item Reports’, users can execute queries that output information such as Tilly’s five most popular items, their least popular items, the five items that generate the most revenue, and the five items that generate the lowest amount of revenue. As an addition to the reports feature, users are given the option to export reports to a csv file for their own usage. An example for one of these reports has been provided in the appendix for reference (Figure 7.5).

*Appendix*

1. **Figure 1.0**



1. **Figure 2.0**

Text

Description automatically generated

1. **Figure 3.0**

Text

Description automatically generated

1. **Figure 4.0**

Text

Description automatically generated

1. **Figure 5.0**

**Text

Description automatically generated**

1. **Figure 6.0**

**Text

Description automatically generated**

1. **Figure 7.0**

****

1. **Figure 7.5**

**Text

Description automatically generated**