**Project: Database System for E-Commerce**

Created by:

Kayla Mormak (kmm137)

Max Kryvenko (mak276)

Adam Kurta (adk55)

INFSCI 2710 – Fall 2014

Tuesday 3:00PM – 5:50PM

**Overview**

This project was designed for a sporting goods company that uses a web-based front-end with phpMyAdmin as the database back-end. The functionality that this database provides includes the following:

* Product browsing and search for the general public
* Account inquiries and payments for existing customers
* Support for regular administration and business operations
  + Modify inventory
  + Keep track of sales
  + Create and update customer accounts
* Restricted access to day-to-day operations limited to existing employees
* Restricted access to add or edit database web app users
* Summarized data based on interesting queries or questions
* Error-handling for common mistakes, like invalid IDs and insufficient stock.

**System Users**

Three different types of users will be able to access the system: Public, Employees and Managers. A public user refers to the store’s customers. An employee refers to the salespeople that will be servicing the customers as well as reporting to the store’s management. A manager refers to the people managing the salespeople that are working in the stores.

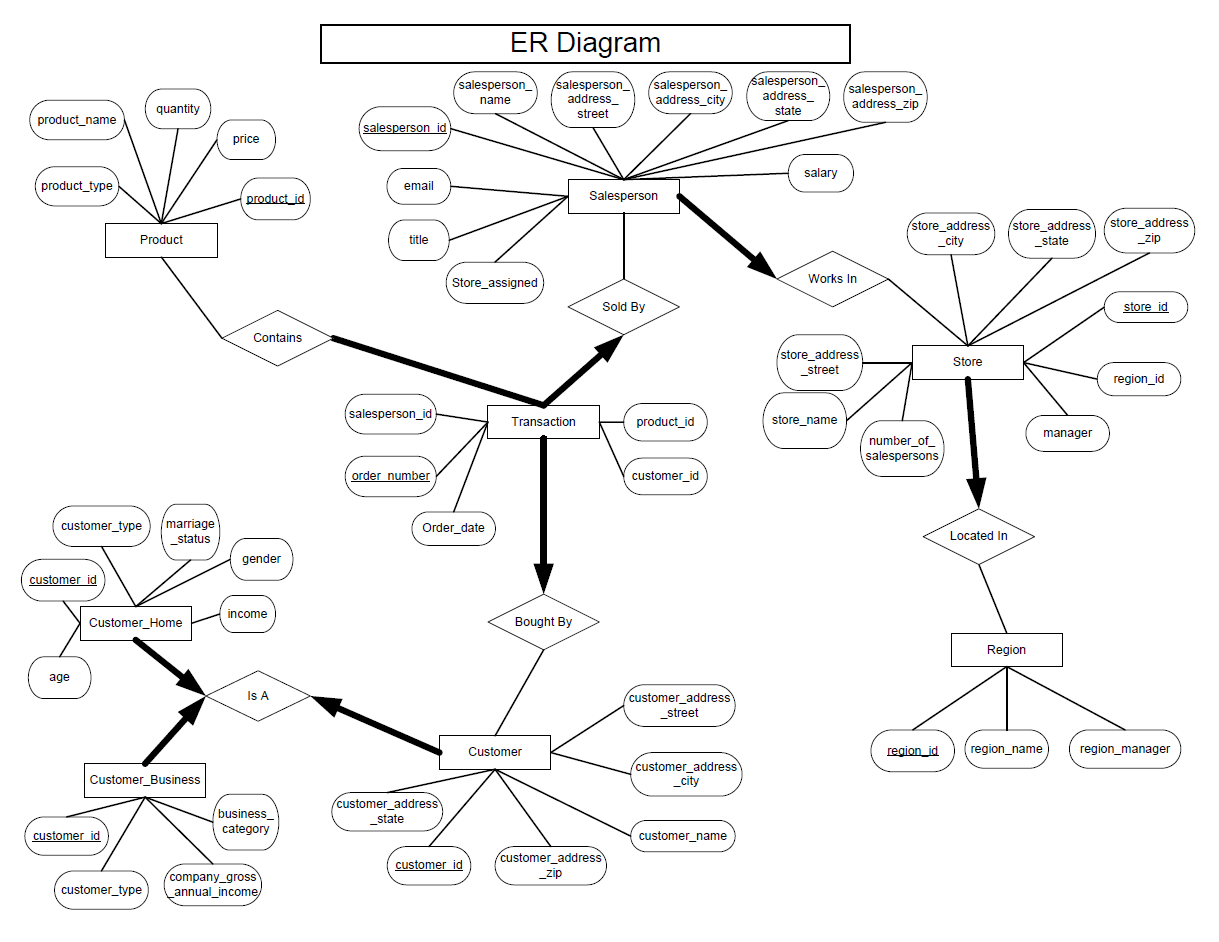
|  |  |
| --- | --- |
| **User** | **Functions** |
| Public | Browse catalog  Search for a specific item by attribute  View data aggregation reports |
| Employees | Add/edit customers  Add/edit transactions  Add/edit inventory |
| Managers | Add/edit/delete users from the system |

**List of Assumptions**

The following is a list of assumptions that have been made regarding the system.

* A customer is free to make purchases at any store.
* While all web forms are visible to all users, the functionality of certain forms is only available to users with the appropriate authorization.
* Once a customer is a business or a home entity, it will never change and/or will create a new account.
* Products remain in the inventory list once they are sold out for future reordering.
* Administrators are available to manually delete data that would be infrequently deleted.

**ER Diagram**



**Relational Schema**

Customer (*customer\_id*: integer, *customer\_name*: string, *customer\_address\_street*: string,

*customer\_address\_city*: string, *customer\_address\_state*: string, *customer\_address\_zip*: string, *customer\_type*: string)

Customer\_Business (*customer\_id*: integer*, business\_category*: string, *business\_income*: integer)

Customer\_Home (*customer\_id*: integer, *marriage\_status*: string, *gender*: string, *age*: integer, *home\_income*: integer)

Accounts (*customer\_id*: integer, *balance*: double)

Salesperson (*emp\_id*: integer, *store\_id*: integer*, title*: string, *salary*: integer)

Store (*store\_id*: integer, *store\_name*: string, *store\_address\_street*: string, *store\_address\_city*: string,

*store\_address\_state*: string, *store\_address\_zip*: string, store\_*manager*: string, *number\_of\_salespersons*: integer, *region\_id*: integer)

Product (*product\_id*: integer, *product\_name*: string, *quantity*: integer, *price*: double, *unit\_cost*: double, *product\_type*: string)

Region (*region\_id*: integer, *region\_name*: string, *region\_manager*: string)

Transaction (*order\_id*: integer, *order\_date*: date, *salesperson\_id*: integer, *product\_id*: integer, *customer\_id*: integer, *product\_quantity*: integer, *price*: double)

Employees (*emp\_id*: integer, *emp\_name*: string, *emp\_address\_street*: string, *emp\_address\_city*: string, *emp\_address\_state*: string, *emp\_address\_zip*: string, *email*: string)

Metadata (*table\_id*: integer, *table\_name*: string, *number\_of\_attributes*: integer, *number\_of\_foreign\_keys*: integer, *number\_of\_indexes*: integer)

**DDL Statements & Normal Forms**

CREATE TABLE Metadata

(table\_id INTEGER,

table\_name CHAR(30),

description CHAR (100),

number\_of\_fields INTEGER,

number\_of\_primary\_keys INTEGER,

number\_of\_foreign\_keys INTEGER,

number\_of\_indexes INTEGER,

PRIMARY KEY (table\_id))

The primary key table\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

CREATE TABLE Customer

(customer\_id INTEGER,

customer\_name CHAR(20),

customer\_address\_street CHAR(50),

customer\_address\_city CHAR(30),

customer\_address\_state CHAR(20),

customer\_address\_zip CHAR(10),

customer\_type CHAR(10),

PRIMARY KEY (customer\_id))

The primary key customer\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

Note: Although infrequent, zip codes do not always determine the state or the city. One zip code can span multiple state and city borders.

CREATE TABLE Customer\_Business

(customer\_id INTEGER,

business\_category CHAR(20),

business\_income INTEGER,

PRIMARY KEY (customer\_id),

FOREIGN KEY (customer\_id REFERENCES Customer))

The primary key customer\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

CREATE TABLE Customer\_Home

(customer\_id INTEGER,

customer\_type CHAR(20),

marriage\_status CHAR(10),

gender CHAR(6),

age INTEGER,

home\_income DOUBLE,

PRIMARY KEY (customer\_id),

FOREIGN KEY (customer\_id REFERENCES Customer))

The primary key customer\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

CREATE TABLE Accounts

(customer\_id INTEGER,

balance INTEGER,

PRIMARY KEY (customer\_id),

FOREIGN KEY (customer\_id REFERENCES Customer))

The primary key customer\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

CREATE TABLE Product

(product\_id INTEGER,

product\_name CHAR(20),

quantity INTEGER,

price DOUBLE,

cost DOUBLE,

unit\_cost DOUBLE,

product\_type CHAR(20),

PRIMARY KEY (product\_id))

The primary key product\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

CREATE TABLE Employees

(emp\_id INTEGER,

emp\_name CHAR(20),

emp\_address\_street CHAR(50),

emp\_address\_city CHAR(30),

emp\_address\_state CHAR(20),

emp\_address\_zip CHAR(10),

email CHAR(50),

PRIMARY KEY (emp\_id))

The primary key salesperson\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

Note: Although infrequent, zip codes do not always determine the state or the city. One zip code can span multiple state and city borders.

CREATE TABLE Salesperson

(emp\_id INTEGER,

store\_id INTEGER,

title CHAR(20),

salary INTEGER,

PRIMARY KEY (salesperson\_id),

FOREIGN KEY (store\_id REFERENCES Store))

The primary key salesperson\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

Note: Although infrequent, zip codes do not always determine the state or the city. One zip code can span multiple state and city borders.

CREATE TABLE Store

(store\_id INTEGER,

store\_name CHAR(20),

store\_address\_street CHAR(50),

store\_address\_city CHAR(30),

store\_address\_state CHAR(20),

store\_address\_zip CHAR(10),

store\_manager CHAR(50),

number\_of\_salespersons INTEGER,

region\_id INTEGER,

PRIMARY KEY (store\_id),

FOREIGN KEY (region\_id REFERENCES Region))

The primary key store\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

Note: Although infrequent, zip codes do not always determine the state or the city. One zip code can span multiple state and city borders.

CREATE TABLE Region

(region\_id INTEGER,

region\_name CHAR(30),

region\_manager CHAR(50),

PRIMARY KEY (region\_id))

The primary key region\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

CREATE TABLE Transaction

(order\_id INTEGER,

order\_date DATE,

salesperson\_id INTEGER,

product\_id INTEGER,

customer\_id INTEGER,

product\_quantity INTEGER,

price DOUBLE,

PRIMARY KEY (order\_id),

FOREIGN KEY (salesperson\_id REFERENCES Salesperson),

FOREIGN KEY (product\_id REFERENCES Product),

FOREIGN KEY (customer\_id REFERENCES Customer))

The primary key order\_id functionally determines all attributes. There are no other FD’s. It is in BCNF.

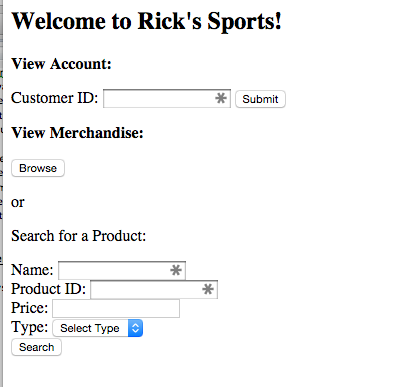
**Front-End & Back-End**

The web application front-end is designed in PHP with JQuery for browser-side scripting. The back-end database is a MySQL database hosted on a local MySQL server, created as described above. Queries to the database are handled using MySQL and the $\_POST method of PHP. The web app is intended to guide the user between pages and should not be navigated to directly. Most pages require variables from the previous page’s $\_POST; this is to prevent unintentional errors in copying data between screens. Some functionality may change (or be lost) if the $\_POST variables are not set.

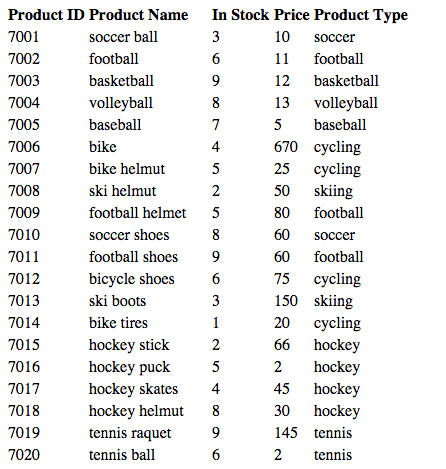
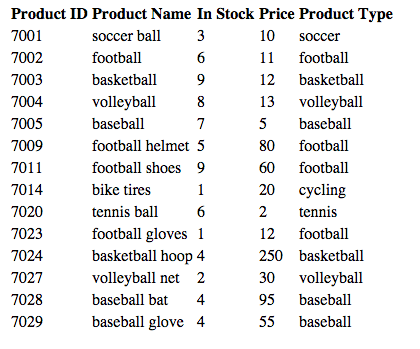
A combination of query types were used to achieve the functionality of the web application. INSERT queries are used to create new users, customers, inventory, and transactions, as well as to generate related tuples in different tables (ex. Customer -> Account). UPDATE queries are used to manually update fields in the employee pages and in the modification of values like account balances and product inventory. SELECT queries provide the results for data aggregation pages as well as to validate users attempting to make changes to the data. DELETE is used just once, to allow managers to delete employees/salespersons. An administrator is required to make any other deletions to maintain a paper trail of business transactions.

There are two general types of pages: the first loads existing data into a form to be updated or added, and the second loads query results into a table for easy viewing. Both the forms and the data output require a successful query prior to use. Forms to add a new tuple to the database provide the next ID, while forms to edit must load all available attributes for the tuple(s) selected. Data output only requires a successful query (it may return zero rows) and returns a set of selected attributes from all rows that satisfied the query.

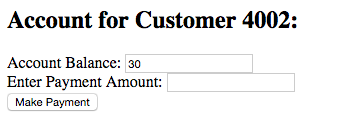
**System Implementation**



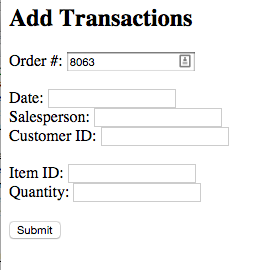
Browse and Search Products:

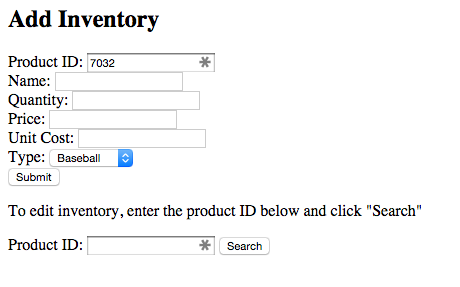
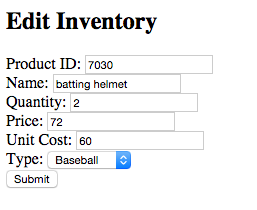
Payment:



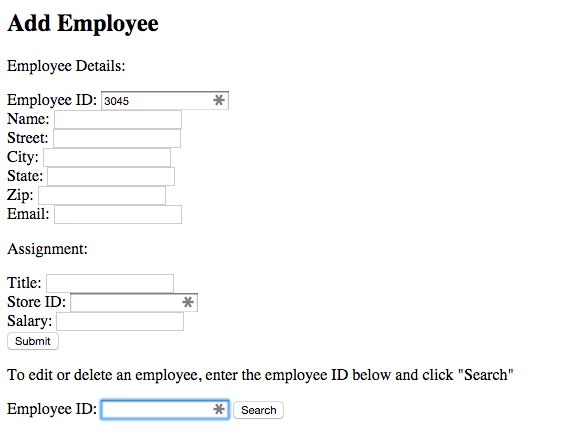
Sales:

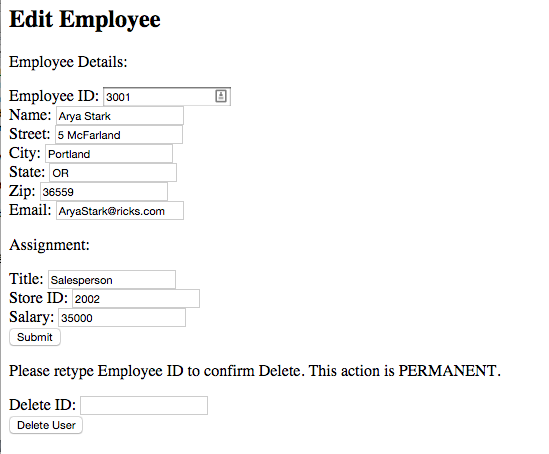


Inventory:

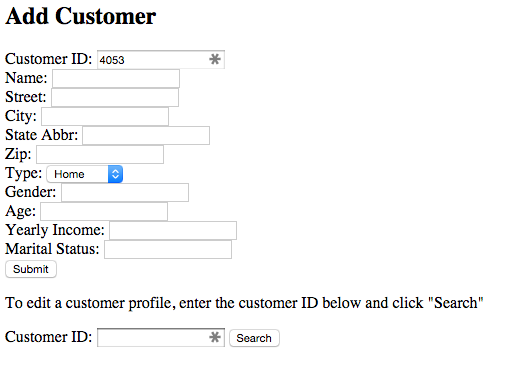
 

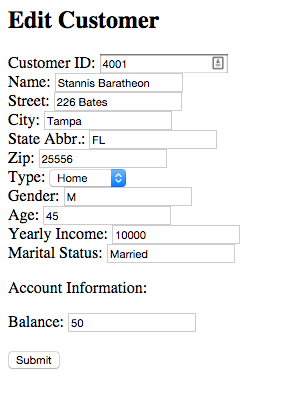
Users:



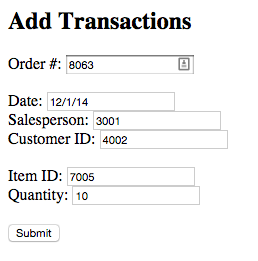


Customers:





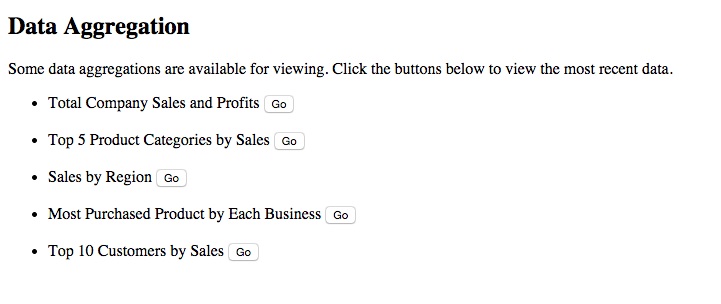
Error-Check:

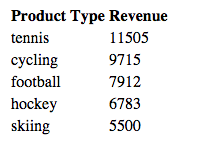
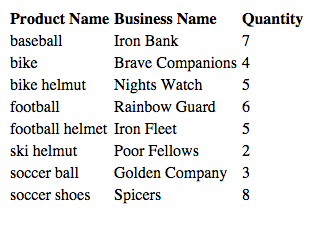


There are only 7 baseballs (item 7005).

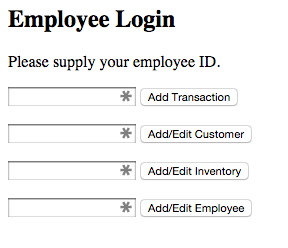
Kayla's Mac:Users:Kayla:Desktop:db screenshots:error_message.png

Data Aggregation:

’

Login:



**System Limitations & Possible Improvements**

* Limitations
  + Entering only one product per transaction; need to re-enter transaction\_id to have multiple products on one transaction
  + Cannot edit employees that are also managers; edit to employees generates a change to salespersons as well
  + No authentication measures; a valid ID is sufficient to make changes to the data in the system
  + Strictly numeric IDs within a range place a limit on the number of IDs that can be used for a given purpose
* Improvements
  + Include ability to show sales growth
  + Functionality to delete a customer or inventory item
  + Methods to add/edit/delete managers, stores, and regions