CS405G Mid-Project Report

Mati Turner, Matt Ruffner, Matt Dunbar

March 21, 2016

1 Group Information

For the purposes of this document, team members will be referred to by their last names to avoid confusion.

The group for this project will consist of Mati Turner, Matt Ruffner and Matt Dunbar. All of us contributed to the creation and design of the database schema. We collaborated to find the simplest design that would allow for efficient queries while still supporting all necessary, and some extra, features.

Moving forward, Turner and Ruffner will focus on creating PHP pages to facilitate DB connection and querying, this will be known as the back-end. Ruffner and Dunbar will work on the user interface which will be constructed using Google's PolymerElements [1], this will be known as the front-end.

Data transfer between the front and back ends will be done using AJAX requests provided by the `iron-ajax` Polymer Element. [2]

All group members will work on deciding a standardized data structure to be passed from the back to front when requests or actions are initiated, e.g. a search for items or user login.

2 Database Needs

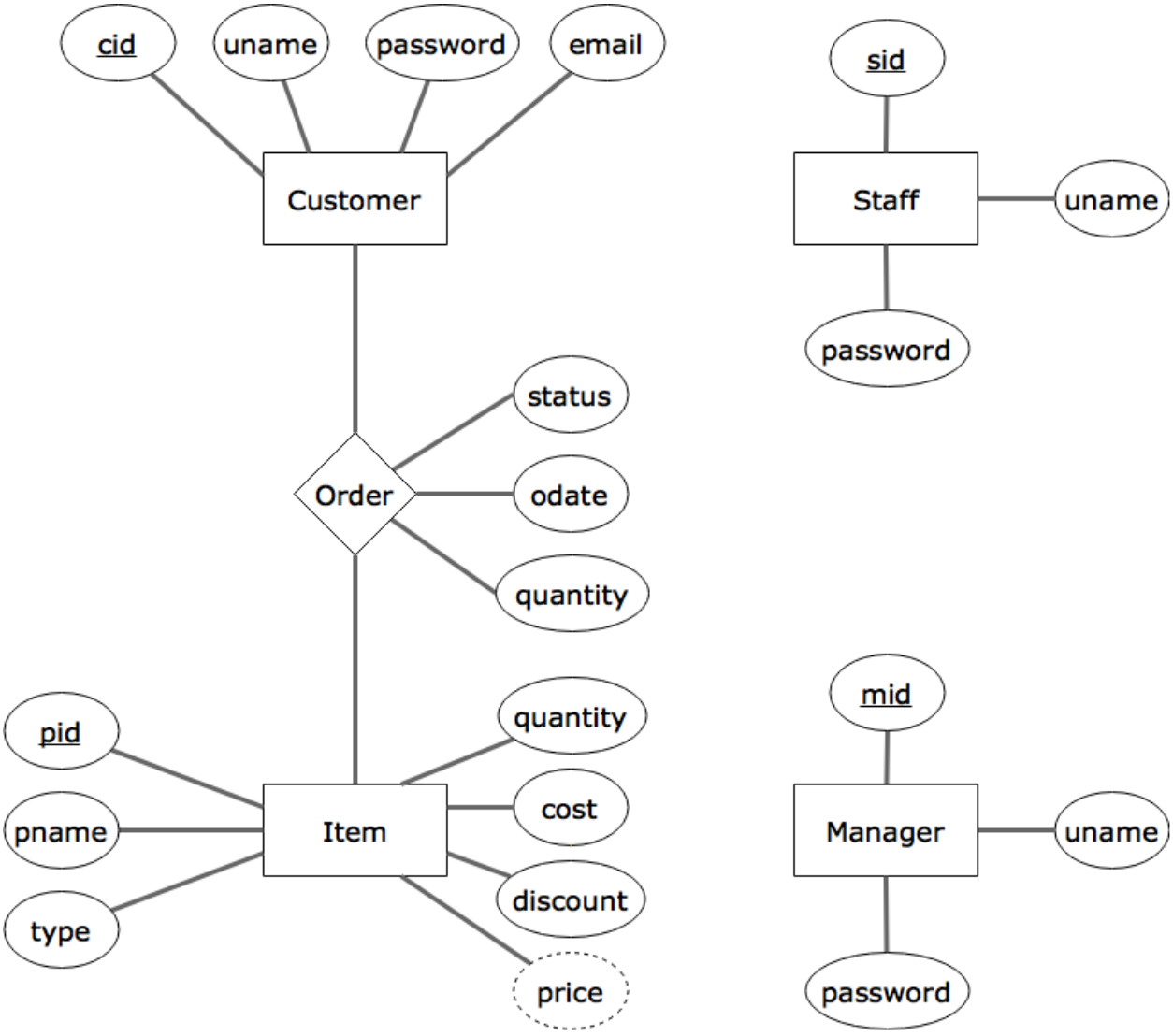
Our database will need the ability to:

- **Create tables** While it may only be once in the inception of our service, we will need to explicitly create tables to store our data.
- **Insert data into tables** Upon explicitly adding items or upon customer interaction, our database system will need to insert entries into the created tables following certain restraints.
- **Update data in tables** Based on our following implementation of the prompt, updating entries within the tables will be crucial to our ordering process. As such, our database will need to be able to access entries within a table and change the value of one or more of that entrys attributes.
- **Check statements to ensure referential integrity** For consistencys sake, we will need to have access to Check statements to make sure no invalid data gets entered into the tables.
- **Trigger statements** For sound design, it is important that the database have access to being triggered to warn for possible malicious or accidental input.

[1] https://www.polymer-project.org/1.0/

[2] https://elements.polymer-project.org/elements/iron-ajax

3 ER Diagram


cid
uname
password
email
Customer
sid
Staff
uname
password
status
Order
odate
quantity
pid
quantity
pname
Item
cost
discount
type
price
mid
Manager
uname
password

4 Database Schema Design

Customer(primary key *cid*: int, *uname*: string, *password*: string, *email*: string)

The cid will be a number that will be assigned in the back-end by auto-incrementation. (Each entry will make the cid be the next consecutive number starting at 0). The *uname*, *password* and *email* are all given by the user.

Staff(primary key *sid*: int, *uname*: string, *password*: string)

The *sid* will be a number assigned in the back-end similar to the *cid* for customers. The *uname* and *password* will be set up either by the staff register or explicitly made for the purpose of testing the system.

Manager(primary key *mid*: int, *uname*: string, *password*: string)

The *mid* will be a number assigned in the back-end similar to the *cid* for customers. The *uname* and *password* will be set up explicitly made for the purpose of testing the system.

Item(primary key *pid*: int, *pname*: string, *type*: string, *cost*: real, *discount*: real, *price*: real, *quantity*: int)

pid will be a four-digit number much like the cid, mid, and sid of earlier relations.

pname is the name of the item.

type is toy or game (for sorting purposes).

cost is a hidden value that is the raw cost of the item. There will be a check on creation and update to ensure this is a positive value.

discount is a real between 0 and 1 which signifies the discount on the item currently (set to 0 by default). There will be a check on update to make sure that this value stays between 0 and 1 and there will probably be a confirmation check if a greater than 50% discount was indeed intended.

price is a derived quantity that is found from the following expression: cost-(discount)*cost

quantity is pretty self explanatory. There will be a check to make sure this value stays above zero in both creation and update.

Order(foreign key *cid*: int, foreign key *pid*: int, *status*: string, *quantity*: int, *odate*: date)

The foreign keys *cid* and *pid* reference the tables Customers and Items. There will be a check to ensure that these fall within the bounds of the current maxima and minima for these IDs (because they are consecutive in creation).

The string for *status* will contain the values pending, shipped, or cart. These refer to when an item has been bought, shipped, and put in a cart respectively. When a user chooses to check out, the order will be updated from cart to pending. When a staff member ships, the *status* will update from pending to shipped.

The *odate* is a time stamp specifying when the item was bought not carted. This is done to ensure that items bought within the same cart will have the same *odate*. This will also allow a user looking through their past orders to see them grouped by date and time. When an item is carted the *odate* will be NULL.