**TECHFAM**

**CMPSCI 465**

**Phase One: Shoe Retailers’ Database**

**Ben Myers**

**Chao Guo**

**Hiren Patel**

**Guojing Wang**

**Jeff Li**

**Dishti Gurnani**

**TABLE OF CONTENTS**

Table of Contentsi

List of Figuresiii

**1. Introduction1**

**2. Analysis and Conceptual Design2**

2.1. Sales Item.2

    2.2. Categories4

2.3. Suppliers.5

2.4. Registered Users.6

    2.5. Rating.7

    2.6. Browsing.9

2.7. Searching.10

2.8. Sale.11

2.9. Bidding.12

2.10. Order and Sale Reports.13

2.11. Delivery.15

2.12. Custom Shoes.16

2.13. Customer Service.17

**3. Conclusion19**

**Appendices20**

Appendix A – GitHub Graphs20

Appendix B – Asana Summary.21

Appendix C – Category Tree.22

**Works Cited23**

**LIST OF FIGURES**

Figure 1 - Sales\_Item ER Diagram2

Figure 2 - Sales\_Item and children ER Diagram3

Figure 3 - Address ER Diagram4

Figure 4 - Image ER Diagram4

Figure 5 - Category ER Diagram5

Figure 6 - Suppliers ER Diagram6

Figure 7 - Registered\_Users ER Diagram7

Figure 8 - Rating ER Diagram8

Figure 9 - Searching Process Diagram10

Figure 10 - Sale Diagram11

Figure 11 - Auction ER Diagram13

Figure 12 - Bid ER Diagram13

Figure 13 – Order Sale Report Process Diagram14

Figure 14 – Shipping and UPS ER Diagram15

Figure 15 - Custom\_Shoe ER Diagram17

Figure 16 - Customer\_Service ER Diagram18

**1. INTRODUCTION**

When forming our startup, we considered several different items to sell. We wanted to distribute a popular product that was easy to categorize for simple website navigation. We brainstormed and decided to join the $52-billion-dollar shoe industry. [5]

After choosing to create an online shoe store, we looked at competing companies to see how they displayed their products. We noticed Amazon provides a name, description, price, image, company name, delivery information, and availability (sometimes including number in stock). [1] While looking at eBay, we found additional important details, like current bid amounts and condition of the product. [3] This gave us a general idea of what information we needed to store for each sales item.

Next we looked directly at the competition, and began browsing shoe sites. We noticed that stores like Designer Shoe Warehouse allowed you to search for shoes based on category, brand, size, and color. [8] We also discovered that shoe stores often sell other products like socks, backpacks, and shoe care items. [2]

Using our knowledge on Database Management Systems we will develop an envisioned model that helps customers shop for shoes in the most intriguing way possible. We will meet the following requirements described further in this analysis.

**2. ANALYSIS AND CONCEPTUAL DESIGN**

In the following section, we analyze the different entities that our database will store, as well as several processes that will be implemented on our site. The goal is to provide a brief introduction to the user experience and explain some of the behind the scenes functionality of the website.

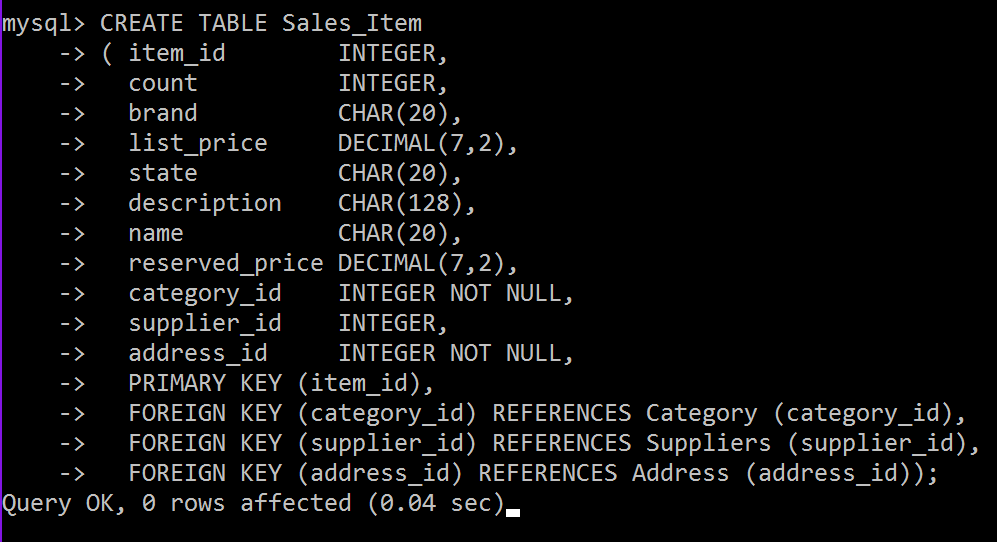
**2.1. Sales Item**

With our initial research complete, we designed the **Sales\_Item** entity to represent each product (See Figure 1). It is identified by its unique key, *item\_id*. Every sales item has a *name* and *description* so sellers can properly market their product. We added a *brand* attribute so users can search for their favorite brands (See Section 2.6). Each sales item has two prices recorded, the *list\_price* and the *reserved\_price*, which are explained in more detail in Section 2.8 and 2.9. The *state* attribute describe the quality of used products. Finally, the *count* attribute tracks how many items are in stock.

https://lh4.googleusercontent.com/kXdlumYU3gmWRWM1WC_agxkJSXZdsoOfff6C8eiuX5KyXlhYLFZXxH8ojeV-Tj2SvEmnCpUZajl63AA3ZIn3XAdZZ3IIIprF_KHLVQA5RVz9_3E493ACbgV2vCYOLCzsiDsfurKz

**Figure 1 - Sales\_Item ER Diagram**

The ER diagram displayed in Figure 1 was later converted into a table in our database (See Figure 2), with **Sales\_Item** set as the name of the table. *Item\_id* was set as the primary key and is stored as an integer value. The rest of the attributes were added as regular columns, where *count* was set as an integer, *brand*, *state*, *description* were set as string values. Also, the *list\_price* and *reserved\_price* were added as decimal values, only allowing 2 digits after the decimal, thus using the notation for currency.



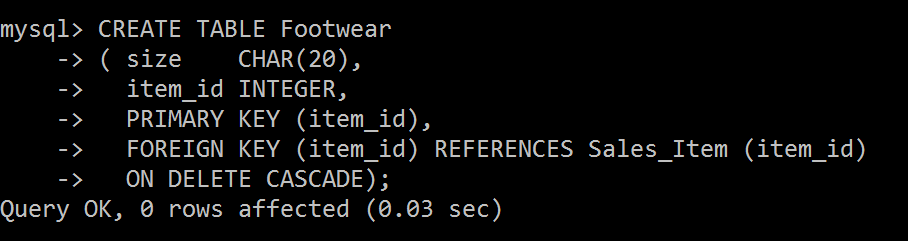
**Figure 2 - Sales\_Item Table Creation**

Like other shoe websites, we wanted to sell non-shoe items. We provide bags, sunglasses, shoe care items, and socks. Bags, sunglasses, and care items can be fully described by the **Sales\_Item** entity, but socks and shoes require additional information. As a result, we created the **Footwear** entity (See Figure 3) to account for these missing fields, which includes the *size* attribute (for socks and shoe size).

figure1_2.png

**Figure 3 - Sales\_Item and children ER Diagram**

In order to show this ISA relationship in the database, we made a new table called **Footwear** (See Figure 4), where a foreign key (*item\_id*) from the **Sales\_Item** table is used as the primary key. That way, every **Footwear** field is required to have a corresponding **Sales\_Item** field, which holds the rest of the data for the **Footwear** field. We also added ON DELETE CASCADE line to the foreign key, so that if the **Sales\_Item** field is deleted, the **Footwear** item field that corresponds to the given **Sales\_Item** will also be deleted. Lastly, the **Footwear** table has a column called size which holds the size of the shoe (recorded as a string).



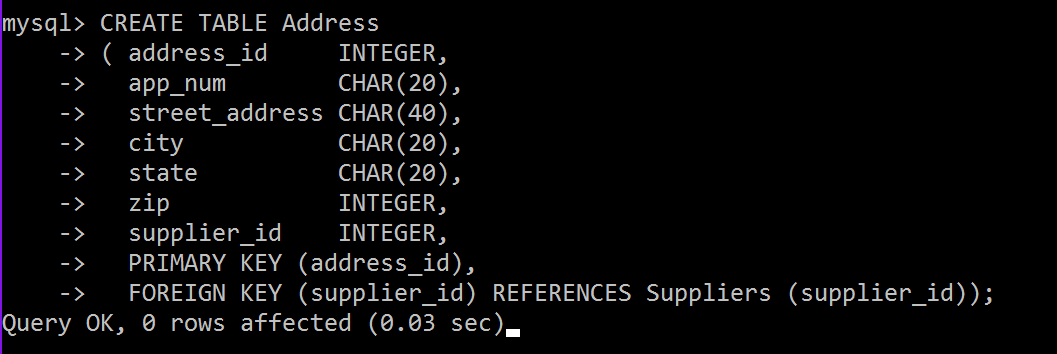
**Figure 4 - Footwear Table Creation**

Next, **Sales\_Items** require information that is too complex to represent as attributes. The **Address** was created as a separate entity because attributes may be reused by other entities, such as users and shipping. Note that every **Sales\_Item** is required to have exactly one address. Attributes of the **Address** entity include the *street\_address*, *app\_num* (apartment number), *city*, *state*, and *zip\_code*. Each **Sales\_Item** has its own visual, where color is an attribute of the relationship between the S**ales\_Item** and the visual. This attribute allows the user to filter out products when searching or browsing, based on the color of bag, shoe, or sock that they desire (See sections 2.7 and 2.8). Visual is represented by the ***Image*** entity (See Figure 5). Every ***Sales\_Item*** can have one or more image, which will be displayed on each product's webpage. Each image has a unique *img\_id*, as well as an *image* attribute, which holds the actual photo data.

figure1_2_3_4.png

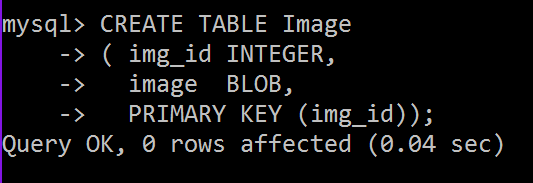
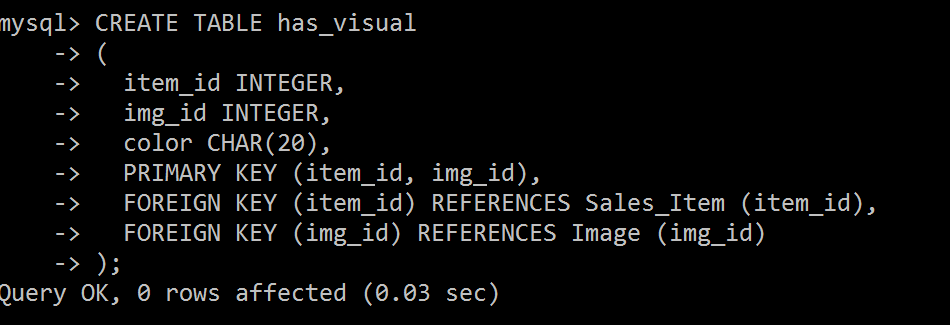
**Figure 5 – Sales\_Item and Relationships**

As shown in the Figure 5, the **Address** entity is too complex to be record as a single attribute, so it was turned into its own table called **Adress** (See Figure 6). Again, the primary key is the *address\_id* (set as an integer) and the rest of the attributes are recorded as strings (or as an integer for zip) in the table. Since a **Sales\_Item** is required to have exactly one **Address**, we added an *address\_id* foreign key to the **Sales\_Item** table, and force all entries in that column to be not null. That way, every field in **Sales\_item** will have an address. Note that an address does not necessarily have to have a relationship with a **Sales\_Item**.



**Figure 6 – Address Table Creation**

The **Image** table (Figure 7) on the other hand, only needs a primary key *image\_id­* (integer) and a column of *image* blobs. As shown in the ER diagram, there is a many-to-many relationship between **Sales\_Item** and **Image**, so the **has\_visual** table was created to record this relationship. In the **has\_visual** table (Figure 8), the primary keys of a **Sales\_item** and **Image** are stored, and the combination of these two foreign keys act as the primary key of the table. In addition, the relationship in the ER diagram contains the attribute *color*. This was added to the table and is represented as a column of strings.



**Figure 8 – Image Table Creation Figure 9 – has\_visual Table Creation**

**2.2. Categories**

There is a variety of footwear to categorize. This categorization will dictate the databases needed for this online business. TechFam must distribute enough styles of footwear to become a central hub for all individuals looking to purchase shoes. However, it is critical to be efficient when creating the databases, avoiding any data redundancies.

TechFam’s shoe selection targets people of all genders and ages. As a result, the root node, “All,” must have three children, “Men”, “Women”, and “Kids”. The children nodes of these three nodes are categories of the different types of shoe, such as “Boots,” “Sneakers,” etc. Next, the different types of shoes are broken down even further by subtypes. In addition to the first three children nodes, there is a fourth node, “Accessories,” contains items that are not shoes. See Appendix C for an in-depth visualization of the categories.

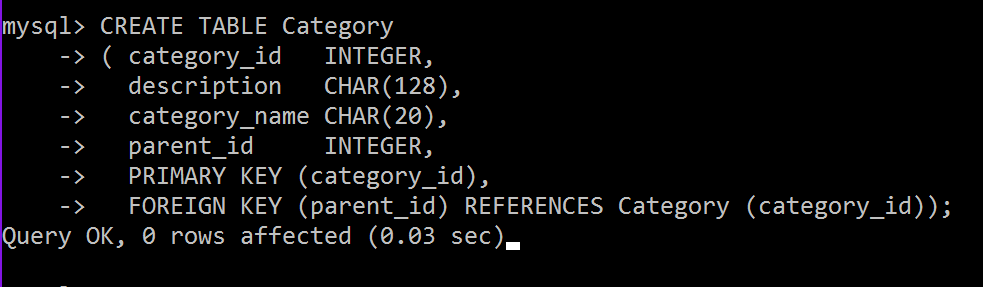
We researched a few different shoe websites to obtain the categories.Zappos [7] is a general shoe distributor that contains the vastest selection, contributing to the majority of our category ideas.JustFab [11] delved deep into the various types of women’s shoes.

The ER diagram in Figure 10 explains the basic outline for how each category will be stored in the database. **Categories** have primary key, *categoryID*, to uniquely identify each other. The *description* attribute briefly details the kind of product that is contained within the category itself. Note that there are relationships between the category and its parent node and children node(s) which follow the classification tree in Appendix C. Each child node points to its parent, and can have a max of one parent. The parent, on the other hand, can have multiple children. This is shown by the **Parent\_of** relationship.

figure5.png

**Figure 10 - Category ER Diagram**

When this ER Diagram was converted into a table, the **Category** entity and the **Parent\_of** relationship were combined into one table called **Category**. This table uses the *category\_id* **(**Integer) as the primary key, and contains *description* and *category\_name­­* columns. These components make up the **Category** entity section of the table. Next, the *parent\_id* was added so that each child could identify its parent **Category** field. Since *parent\_id* is an attribute, we guarantee that each child can only have a max of one parent (or no parent at all if it’s the root node), while parents can still have multiple children.

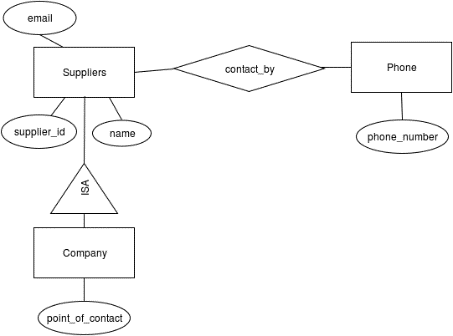
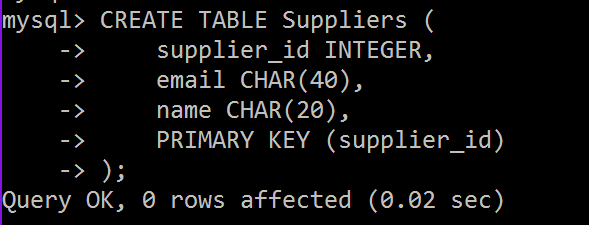


**Figure 11 – Category Table Creation**

There is one more important relationship in the ER diagram called **member\_of**. This shows that every **Sales­\_Item** is required to have an identifying category. This relationship was merged with the **Sales\_Item** table (Figure 2) with the *category\_id* column. This is a foreign key that points to the **Category** the **Sales\_Item** is a member of. Note also that *category\_id* is set to NOT NULL, so that every **Sales\_Item** is required to have a **Category**.

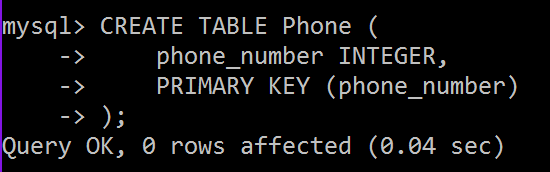
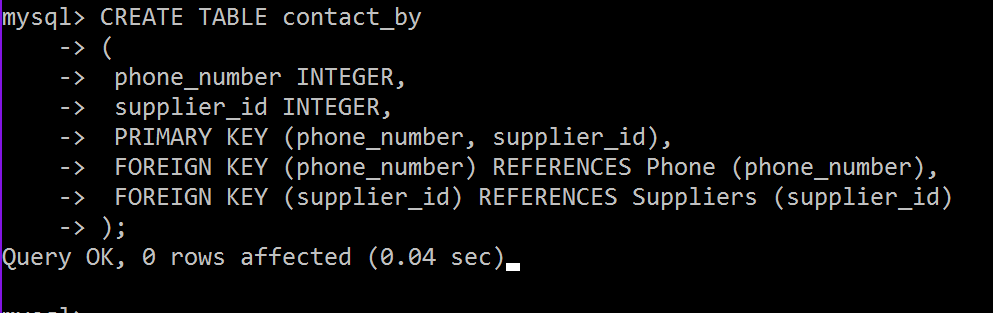
**2.3. Suppliers**

Suppliers is a parent entity of both companies and registered users. A **Suppliers** job is to sell shoes that they manufacture on our website. The attributes for the **Suppliers** entity include the *supplier\_id*, *email* and *name*. The supplier is assigned a *name* and is used to identify the supplier.

**Figure 12 - Suppliers ER Diagram Figure 13 – Suppliers Table Creation**

Multiple phone numbers of the suppliers is also recorded, as represented by the **contact\_by** relationship. The point of contact helps the customer contact the seller in case there are any questions or concerns. This relationship requires two additional tables (besides the **Supplier** table). The first table is the **Phone** table, which simply has a primary key and a phone number. The second table is the **contact\_by** table. This table takes two foreign keys, the primary key of **Suppliers**, and the primary key of **Phone**. This then matches a given phone number with a supplier. The primary key of this table is the combination of both foreign keys.

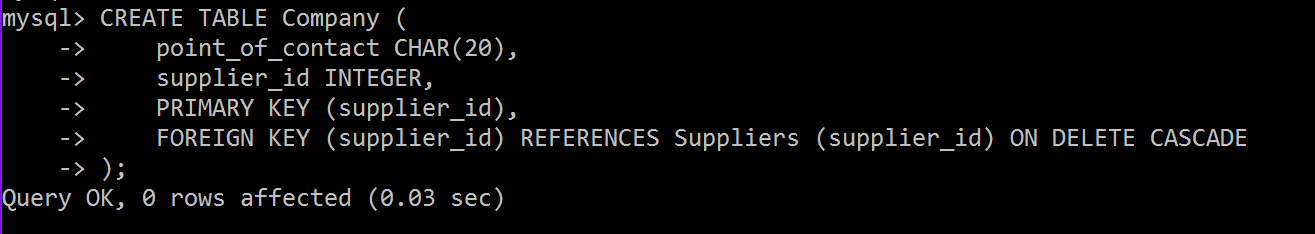
**Figure 14 – Phone Table Creation** **Figure 15 – contact\_by Table Creation**

The supplier also has an **Address**, shown by the *location\_adddress* relationship. A **Supplier** can have many addresses, but an **Address** can only point to a single **Supplier.** Therefore, there is a one-to-many relationship between **Suppliers** and **Address**. The database records this information by adding the *supplier\_id* attribute to the **Address** table (Figure 6), which points to the primary key of a **Supplier**.

**2.4. Company**

**Company** is a child entity of **Supplier**. It contains all the information of supplier, with the addition of the *point\_of\_contact* attribute, which the name of the person to contact for the given company. **Company** can use the name attribute of **Suppliers** to record the company name, and also use the email attribute as well as the phone and address relationships to record where the company is located, and how to contact them. An important thing to note is that companies cannot buy products, they can only sell.

The **Company** table is fairly simplistic. It uses the **Supplier’s** *supplier\_id* as its primary key. If the **Supplier’s** field is deleted, the corresponding **Company’s** field will also be deleted. The Company table also has the *point\_of\_contact* column, which stores strings.



**Figure 15 – Company Table Creation**

**2.4. Registered Users**

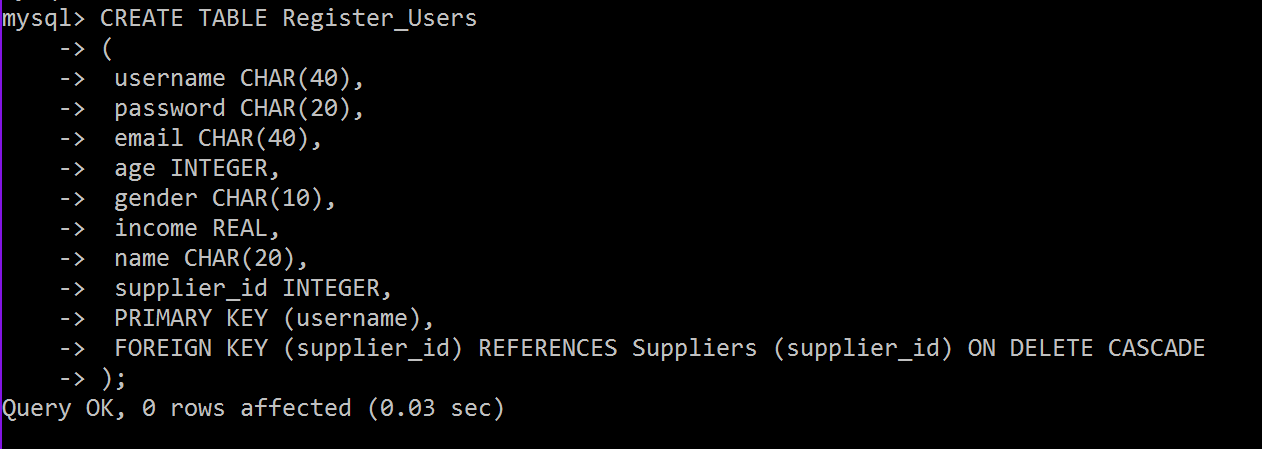
**Registered\_Users** is the second child entity of **Suppliers**. Along with selling products, **Registered\_Users** can bid or directly buy shoes. This feature asks the user to fill out the following information while they register:

* A user ID or username, which helps users stay anonymous and allows us to identify them.
* A password, to secure their account.
* Gender.
* An email and multiple phone numbers as a mode of contact to the registered user.
* Multiple credit cards and addresses (as described in section 2.1 in figure 5) of the registered user, which helps process payments, billing information, and return information.
* Age and income is used to calculate a statistics for each user. The income here stands for the income the user has made on our website.

figure6_7_full.png

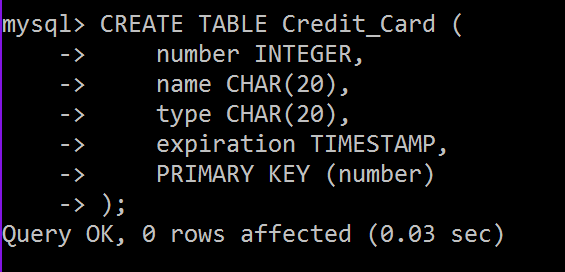
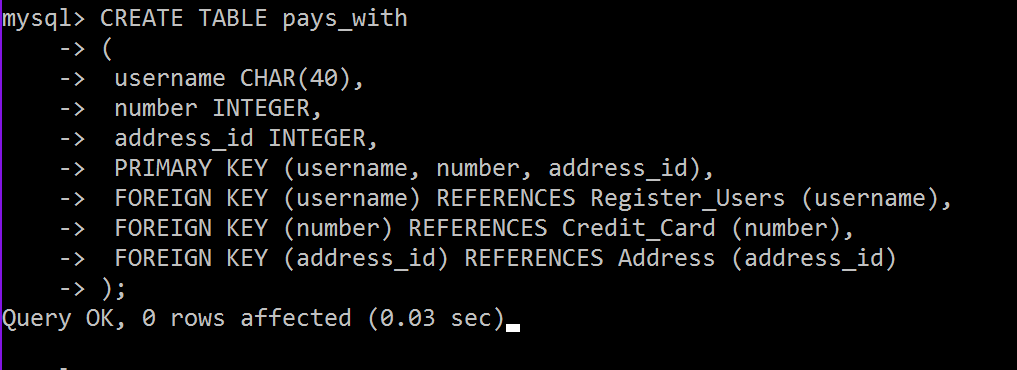
**Figure 16 - Registered\_Users ER Diagram**

Similar to the **Company** table, the **Register\_Users** table has a foreign key that links to a unique *supplier\_id*. The *supplier\_id* cannot be null, that way every **Register\_User** has a corresponding **Supplier** entity. And like the **Company** table, if the **Supplier** field is deleted, so will the **Register\_User** field. Unlike the **Company** table, the **Register\_Users** use their own *username* attribute as the primary key. Since every *username* must be unique, and since that is the main way other users will identify one another, we thought *username* would suite as a better primary key.



**Figure 17 - Registered\_Users Table Creation**

One important relationship to note with **Register\_Users** table is the *pays\_with* attribute. This links the **Credit\_Card** entity, **Register\_User** entities, and **Address** entities together. This allows each **Register\_User** to have multiple credit cards and multiple billing addresses. This information is stored in the table **pays\_with**. It records the *username*, credit card *number*, and *address\_id* of each member of the relationship. It also uses these three foreign keys as the table’s primary key.

** **

**Figure 18 – Credit\_Card Table Creation Figure 19 – pays\_with Table Creation**

**2.5. Rating**

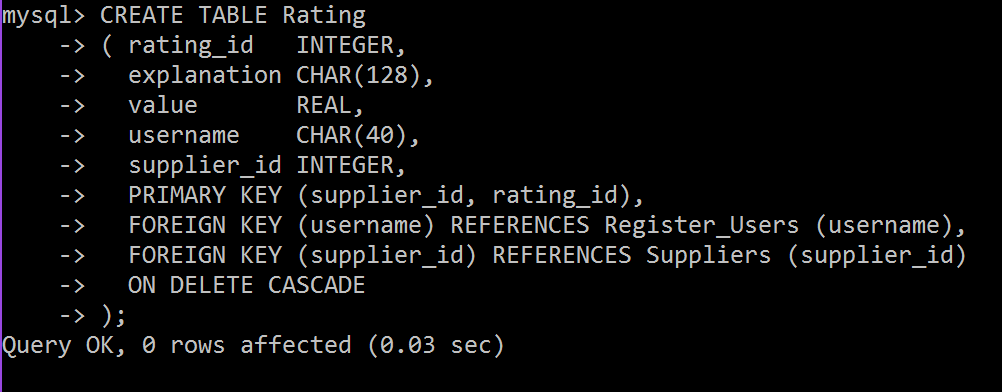
Rating is an important measure for a seller’s reputation. A user might want to know what rating a seller might have. Many websites now have rating system for sellers. For example, sellers on eBay can earn stars based on the number of ratings they get [4]. The rating system on this website can be modeled by a **Rating** entity, which has a relationship with a single **Register\_User**, who create the rating, and a single **Supplier**, who the register user rates. The application quality entity **Rating** is created with primary key *rating\_id*, a scale based non-key attribute *value*, and a textual non-key attribute *explanation*.

figure8.png

**Figure 20 - Rating ER Diagram**

The *value* is a scale attribute that can be selected from 1 star to 5 stars. The *explanation* is a textual attribute that allows a user to explain why one gives such scaled-rating or any additional comments. A seller will also have an average rating showing on his/her profile. The average rating calculates the mean of each rating value then rounds to the nearest star scale.

For the table for **Rating**, there is a partial key *rating\_id*, in addition to the *explanation* and *value* attributes. There is also a foreign key pointing to the *username* of the reviewer and the *supplier\_id* of the company or user being reviewed. These two foreign keys help combine **Rating** with the **by** and **has** relationships. Also, in the ER diagram, **Rating** is a weak entity to **Supplier**. So if the **Supplier** deletes their account, then all the ratings for them will also be deleted. This is done with the ON DELETE CASCADE line where the *supplier\_id* is declared a foreign key. **Rating** is given a primary key which combines the *supplier\_id* with the partial key *rating\_id*.

****

**Figure 21 - Rating Table Creation**

**2.6 Browsing**

Users are able to browse the items by traversing the category tree (Figure 2.1). When a user chooses a category, the user will want to know all of the categories that are directly linked to that current category. The tree traversal algorithm to be implemented will follow a derivative of the breadth-first search algorithm. At each point, they are given a summary of all the items that appear in that category. For example, if the user is in the “Women’s” category, the provided categories are “Boots”, “Flats”, “Pump”, “Sandals”, “Sneakers”, and “Wedges”. However, in traditional breadth-first search algorithms, the other nodes in the same level as the previously mentioned categories, will be shown. So our form of the algorithm will limit the traversing within the current node and its children.

Another important browsing feature is to allow the user the ability to traverse both ways through the tree, including moving downwards and upwards through the tree. The user will be able to return to the parent node of their current category. In addition, there will be a home button to traverse the user back to the root node.

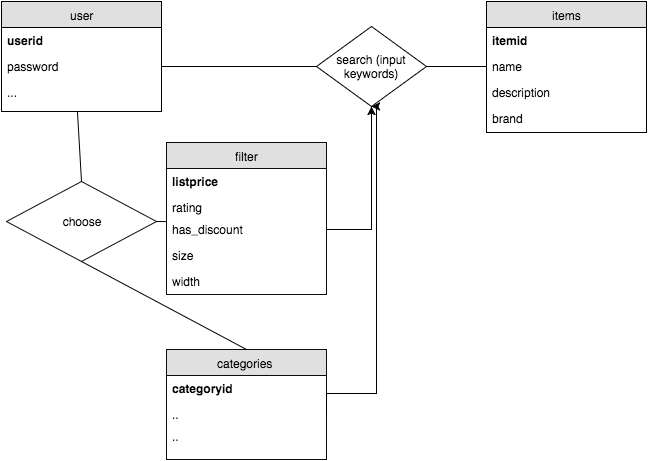
In the browser bar, user can access their account information. This includes their profile, feedback, and settings. Profile contains their personal information, items they are selling, current bids, and history. Feedback has ratings and reviews. Settings include login information, credit card information, and shipping/billing address.

There will be a homepage which consists of the current deals and other features usually found in homepages. This homepage is not located anywhere in the classification tree, yet contains various items from different categories.

During our research of other shoe/clothing sites, we were attracted to a certain type of user interface that made browsing very easy and fast. Particularity, Macy’s online store featured a row of tabs near the top of the page. These tabs were the major categories of clothing, “Men”, “Women”, etc. When the user selects one of the tabs, a drop down menu appears, listing all of the categories within that major category. Furthermore, under those categories were subcategories that were indented to easily show its lower levelness. We found this interface in other sources as well and we find this way of browsing to be the most user-friendly because it gives the user a broad overview of the classification tree. TechFam will follow this structure of browsing.

**2.7. Searching**

On shopping websites, like Amazon, a user might want to search for a series of specific items. This can be modeled by two kinds querying, “User search Items”, and “User choose Categories then search Items”, as shown in Figure 9.

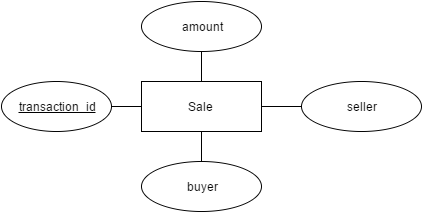


**Figure 22 - Searching Process Diagram**

If a user chooses to input keywords, the database will look for items that contains these keywords. A user can also choose certain categories or other filters first then input keywords to narrow down the results. Attributes in **Sales\_Item** like *name, description, brand,* etc. shall be searchable.

**2.8. Sale**

Items on sale will be the ones not auctioned off. We will create an option for buying the item now. When the user enters a specific item description, that item information will be pulled from our database. The user interface will display all the items matching that description. The user will be allowed to add items to the cart, use their credit card information and check out. After the transaction has completed the item count will decrement. The transaction information will remain in the system for six months and will be stored using the ER diagram in Figure 10.

****

**Figure 10 - Sale ER Diagram**

**2.9. Bidding**

Unlike many shoe websites, our database will have the option of auctioning items. A user can access these items via searching the pull-down menu and clicking on an item through the categories included, as well as entering a specific item name through the search bar. The database will retrieve information and after clicking the auctioned items option the user will be able to view the item. The user has the option to sort items based on price from highest to lowest or vice versa.

*Bidding Rules:*

•    Users must remember each bid is a contract.

•    Each item will have a specific time stamp listed under description.

•    Bids can only be canceled 24 hours prior to time stamp deadline.

•    Bids will be in 2 dollar increments.

*Selling an item:*

•    Under options, select selling button and Users can only sell one item at a time.

•    Select the item for sale and give a brief description.

•    List the timestamp value for the starting and ending.

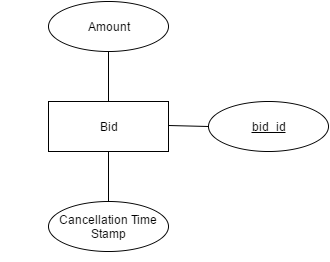
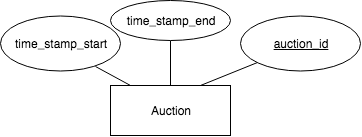
•    Select the reserve price for the item.

•    Review everything and select the confirm button.

•    You will be notified when the auction has ended and who had the winning bid.

We will feature manual bidding and automatic bidding. First let’s talk about the manual way, where the user enters a bid value that is at least $2 more than the current highest bid. The user will be notified when someone overtakes them as the lead bidder. On the other hand, we have automatic bidding, where the user will be prompted to enter the maximum value they are willing to pay for the item being auctioned off. The system will bid $2 more than the current highest bid until your bid reaches the maximum value you are willing to play or the auction ends. The user with the highest bid wins, and pays the amount they bid. For both bidding options at the end of the auction, the user with the winning bid will be notified, that he or she has won. The system will charge the user’s credit card with winning bid amount and after receiving the payment, the seller will be required to ship the item. The transaction will stay in the system for 6 months. The user can view their current bids under the account options.

The **Auction** will have an *auction\_id* pertaining to each item in system that is on auction. It will have a *time\_stamp\_start* for the start of the auction and a *time\_stamp\_end* for the end of the auction. The auction will record all the bids that users place and these will have a relationship to the sale items.



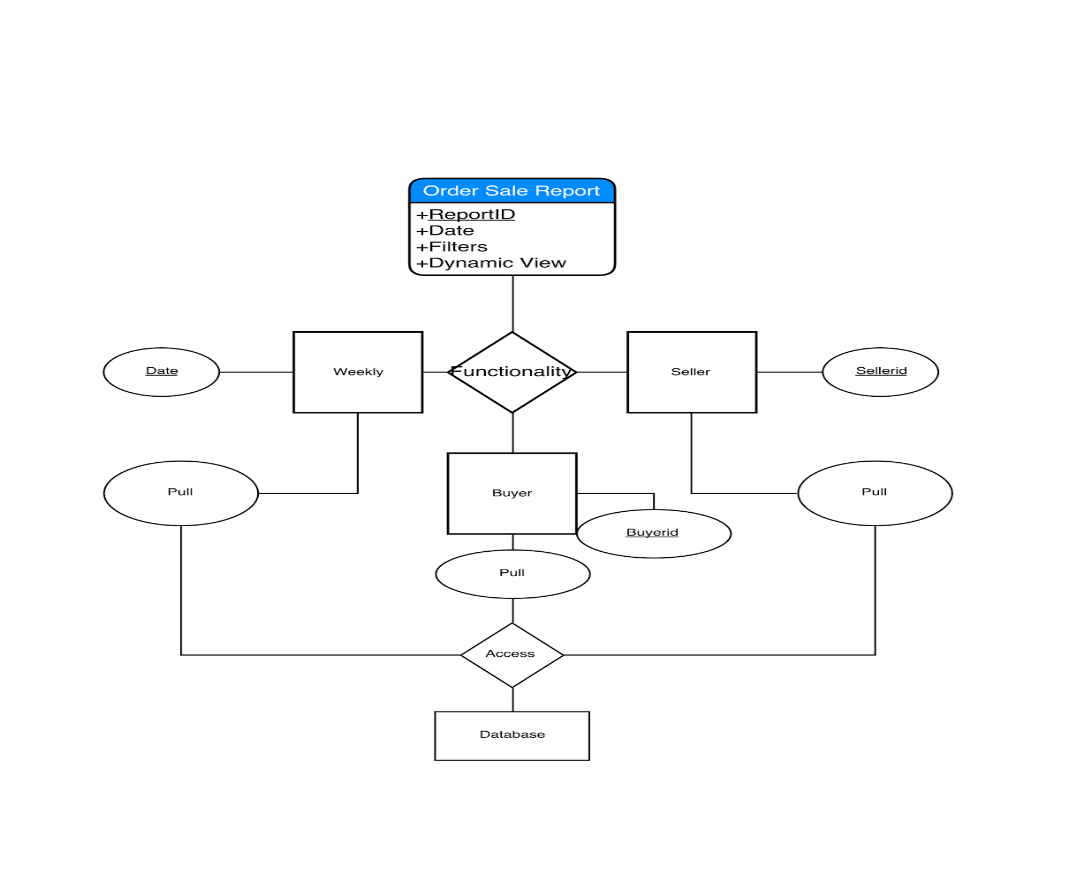
**Figure 11 - Auction ER Diagram        Figure 12- Bid ER Diagram**

**2.10. Order and Sale Reports**

With everything set up, it is important to consider how our system keeps track of all the orders that go through the system. We want to be able to generate a weekly report for each category that we are selling. This will not only inform us on how we are doing in terms of revenue but also help us monitor consumer behavior. By recording and analyzing reports of user activity, companies can improve their sites functionality.[9] Having this functionality in our application and design, we believe our marketing team will be able to take a full advantage of this report function and produce the most effective marketing strategy that will help to maximize company’s profit.

Moving into the implementation of our report, each report will carry a specific report id that is relevant to the seller account detail and the time period for which the report is on. Our goal is to make our report as informative and dynamic as possible. We want to create an interface that allows our users to easily assess their data with a few clicks of a button.

Our design will include a bunch of useful keywords that relate to type of data that will be displayed such as total revenue, total cost, number of purchases, etc. In the backend of this functionality, we will use these keywords and put together a query that will access our database and pull up the request data and present them in a neat view.

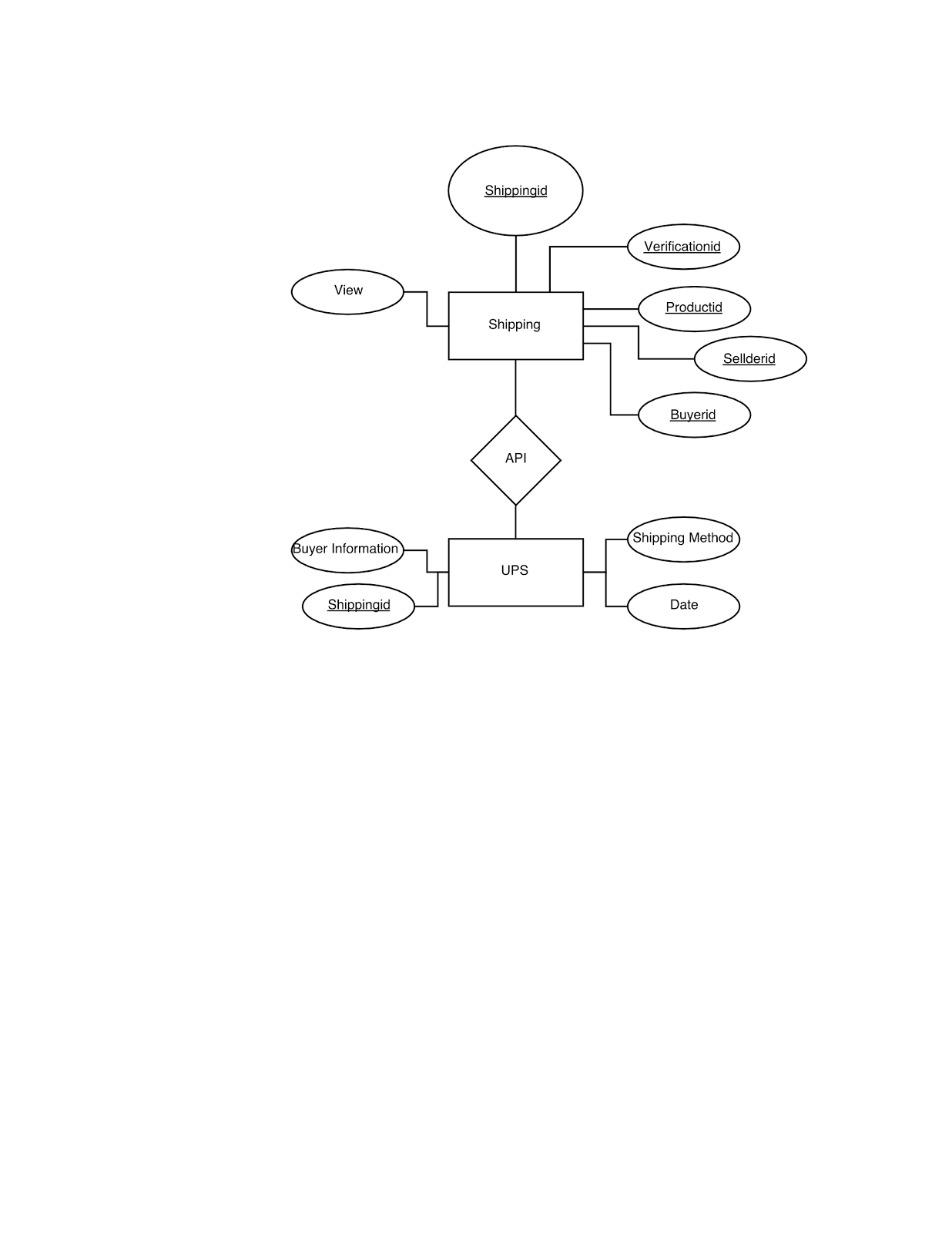


**Figure 13 - Order Sale Report Process Diagram**

**2.11. Delivery**

The other important thing that we are focusing on is our delivery system. We understand that it is our duty to fulfill our customers’ orders with the best service possible. This is why we are taking extra steps to secure every single transaction amongst our users.

With our delivery system, we will generate a unique id for every delivery that is only available for the buyer, seller, and our engineers. Nevertheless, this id will only be created after a confirmation of payment has been posted. This means there will be less of a chance of messing up shipping confirmation and confusing our shipping API providers. Why compete with shipping companies when companies like UPS, FedEx, and DHL have proven themselves to be among the best in the industry. While companies like UPS help to provide their customers with useful developer API, we aim to utilize their API to create the best interface for our users without traversing to another website. [10] We want to provide the tracking number for the shipment to allow users to see the status of their shipment from our website. We hope that our efforts in providing a friendly user interface will keep our customers satisfied with our products and services.



**Figure 14- Shipping and UPS ER Diagrams**

**2.12. Custom Shoes**

Unlike most sites, our team decided to create a custom shoe feature. We researched Nike’s custom shoe system for guidance. [6] First, the user will click on the custom shoe feature in the browser bar. Then, they can pick between a men’s, women’s, boys’, or girls’ shoe. After that, they can pick a particular category of shoe, such as athletic, casual, etc. and then pick a subcategory, such as running shoe or cleat. From there, the user has several features they can choose from a multiple choice list. These features include the size of the shoe, the width of the shoe, the side, back, and front image, the shoe style, tong style, the shoe sole style, the types of laces, and the coloring of the shoe.

The way the custom shoe information is stored is in the **Custom\_Shoe** entity. This entity is a sub-entity of the shoe entity, containing several additional information for the customize process. This includes a side, back, and front image labels, a tong style, a shoe style, a sole style, a laces style. It also has a *custom\_shoe\_id* for the unique key. In our database, we will store every possible combination of shoes we will allow the user to make as a **Custom\_Shoe** entity. Then, during the creation process, as the user selects new attributes, different custom shoes will be displayed on screen, based on the default selections and the user selections.

https://lh4.googleusercontent.com/0w1r_LXVv_GYh1kt-lHFym3VQgCGhfr-ZbNqqjCevUp2Kfym1upQ2Fti_8tzHdzBHDaQHWijXtgoBhlZDTinprCxTFQx3LoDEcCBdkAkEh4MEwWp4lRusEld6zX4M4hcD6UiKbLB

**Figure 15 - Custom\_Shoe ER Diagram**

**2.13. Customer Service**

We are providing users a way to contact us. This bonus feature allows us to connect us with the users. Users can email or call us. We also provide a FAQ page that helps users navigate better. This page also helps users get answers to common questions. Customer service options will be located at the bottom of every page. This feature also helps us get feedback about our services, which provides a scope for improvement.

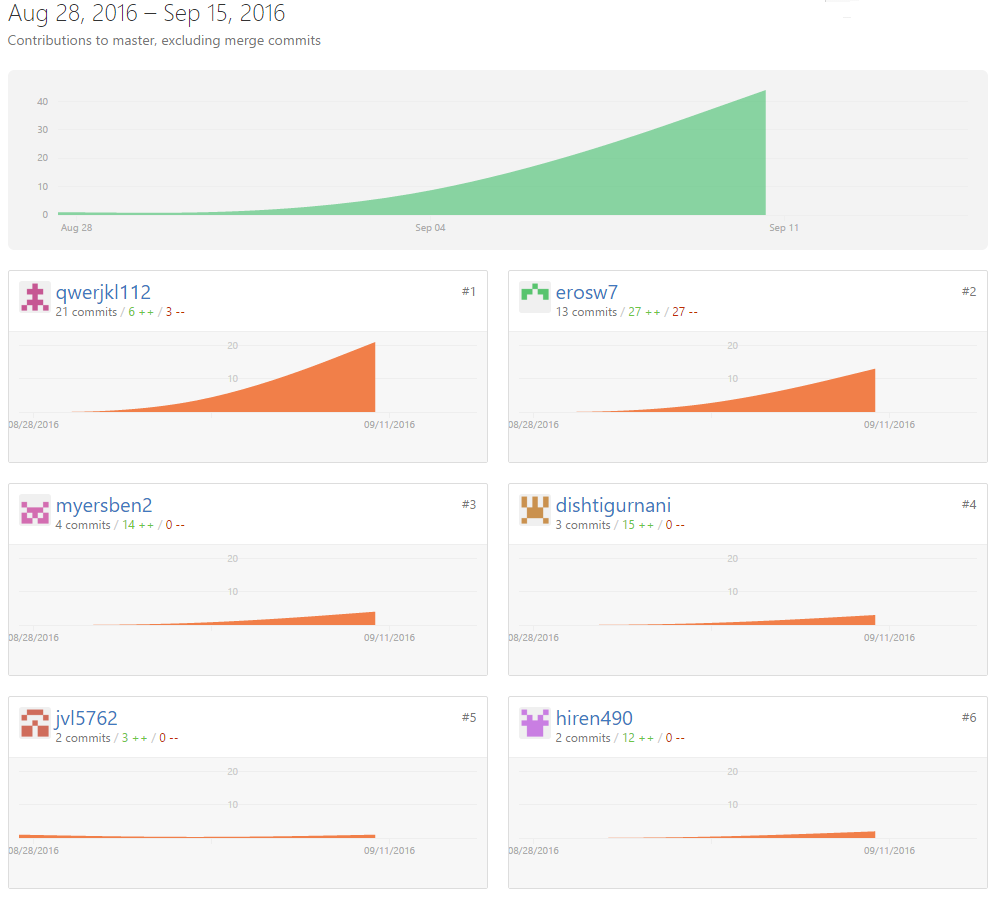
https://lh4.googleusercontent.com/K4yR7AUfwQyPRtQ80SXhcCujRodS9PQL1N7RLFgZyMgfwZdLj3B43rPQ0U_eZ44oj4-LP6EC6vab4PyBLWNyGi7gL2ZwCZhlEk_MRoRqbLNhYraQ9F64OjgbNfV5-aPW_nOjUSM2

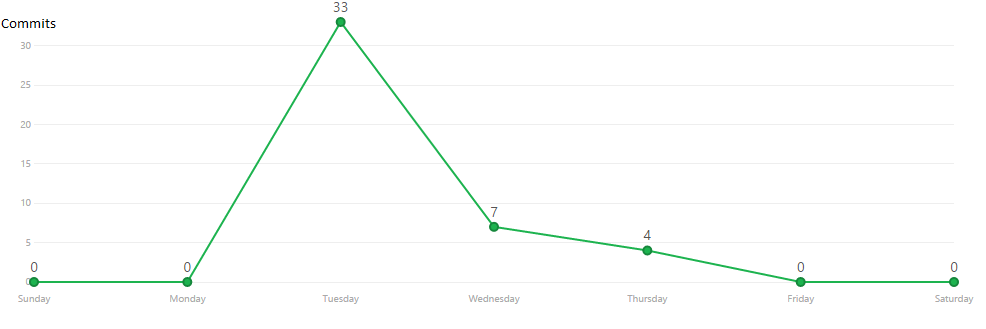
**Figure 16 - Customer\_Service ER Diagram**

**3. CONCLUSION**

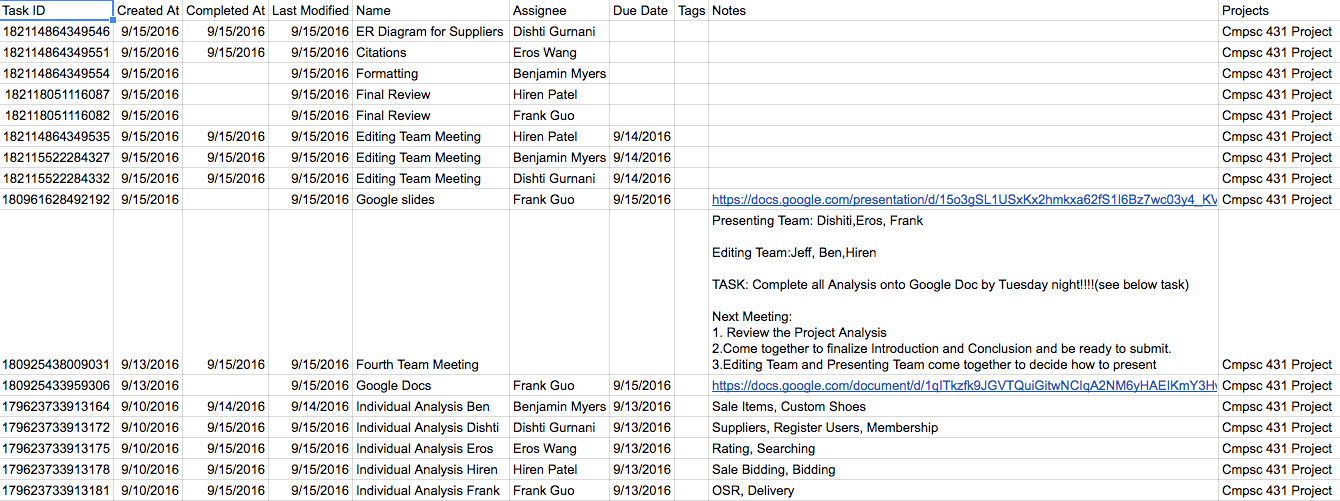
There were many options to choose from when selecting a database. After our first meeting we gathered ideas from everyone and ended with a shoe retailers’ database. The shoe industry is enormous and there were many features that we wanted to talk about besides the given ones. Everyone did their research on ideas and how we would incorporate the given features. The bonus features we selected were custom shoes and customer service. We assigned each person separate features, where each would do research on their specific topic and write a rough draft for analysis, we discussed how each person should implement certain ideas pertaining to their feature. Next, we broke up into 2 teams, we had 3 on editing, while the rest on presentation. The biggest challenge we faced was making sure the information lined up perfectly between each topic because some features are closely connected to each other. Our main goal was making sure we developed a database that was user-friendly, yet offered complex features. Overall, phase one was a success where we implemented the design specifications according to the criteria.

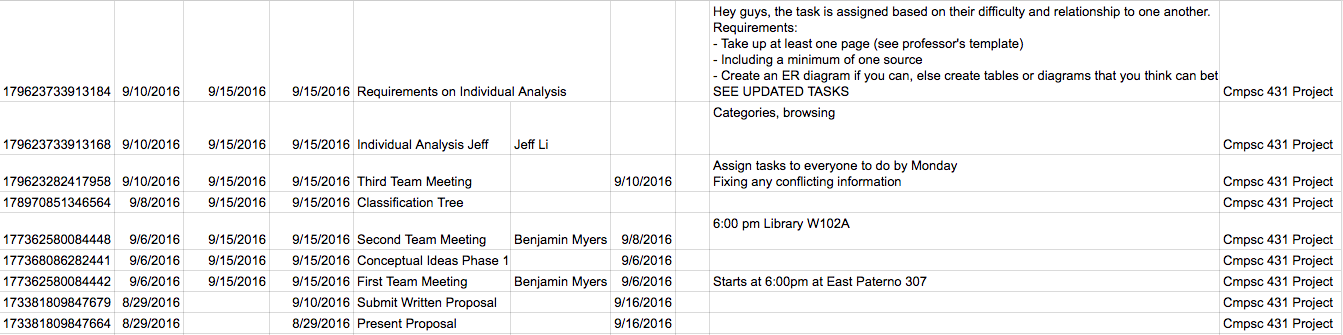
**APPENDIX A – GITHUB GRAPHS**

****

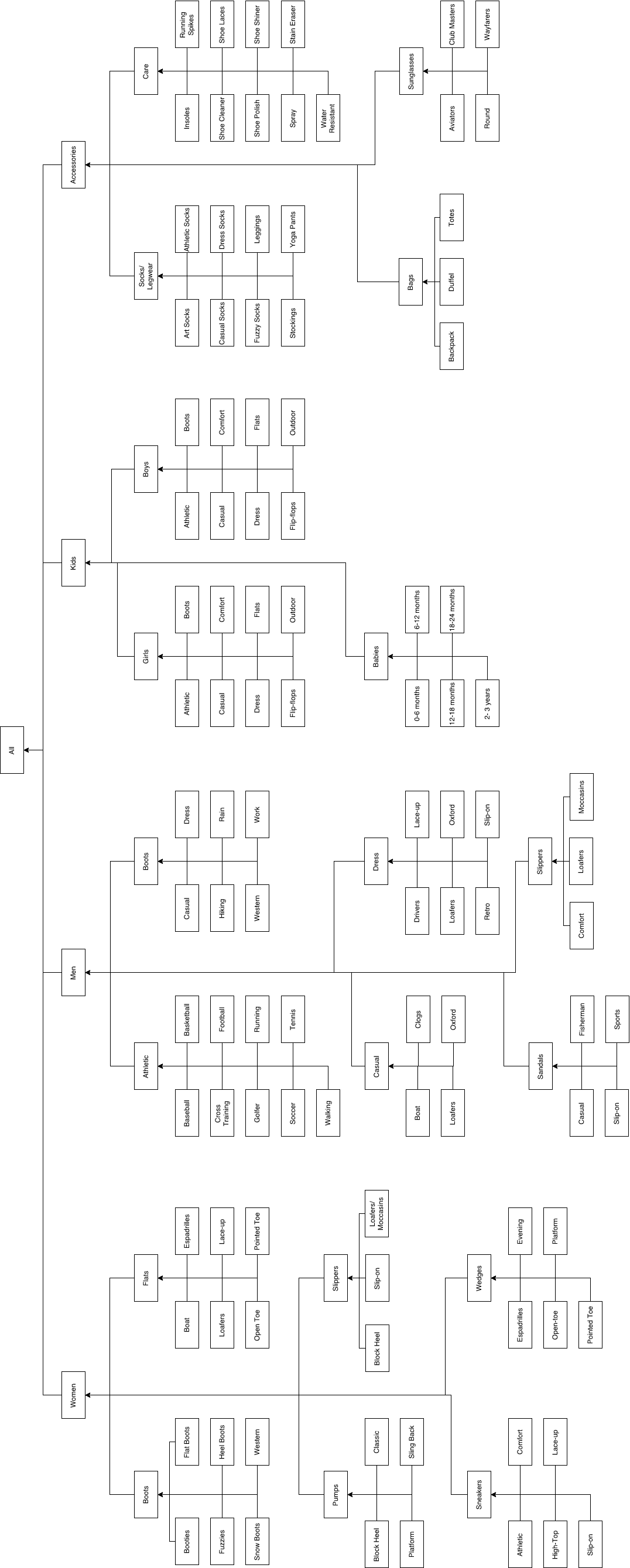
****

**APPENDIX B – ASANA SUMMARY**

****

****

**APPENDIX C- CATAGORY TREE**



**WORKS CITED**

1. "Amazon." Amazon. N.p., n.d. Web. 13 Sept. 2016.

2. Caleres. *Shoes, boots, sandals - famous footwear online*. 2016. Web. 15 Sept. 2016.

3. "EBay." *Electronics, Cars, Fashion, Collectibles, Coupons and More*. N.p., n.d. Web. 15 Sept. 2016.

4. "Feedback Scores, Stars, and Your Reputation." Feedback Scores, Stars, and Your Reputation. N.p., n.d. Web. 13 Sept. 2016.

5. Harden, Seth. "Footwear Industry Statistics." Statistic Brain. N.p., 03 Sept. 2016. Web. 15 Sept. 2016.

6. "Official Store. Nike.com." N.p., n.d. Web. 15 Sept. 2016.

7. "Online Shoes, Clothing, Free Shipping and Returns| Zappos.com." Online Shoes, Clothing, Free Shipping and Returns| Zappos.com. N.p., n.d. Web. 15 Sept. 2016.

8. "Shoes, Boots, Sandals, Handbags, Free Shipping! | DSW.com." *Shoes, Boots, Sandals, Handbags, Free Shipping! | DSW.com*. N.p., n.d. Web. 15 Sept. 2016.

9. "The Purpose of Weekly Status Reports." Reportingadvice. N.p., 2013. Web. 14 Sept. 2016.

10. "United States." UPS Developer Kit. N.p., n.d. Web. 14     Sept. 2016.

11. Women’s shoes, boots, Handbags & clothing online. 2010. Web. 15 Sept. 2016.