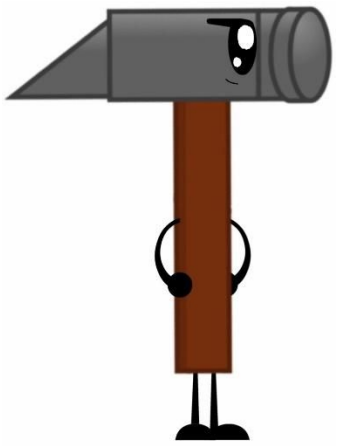


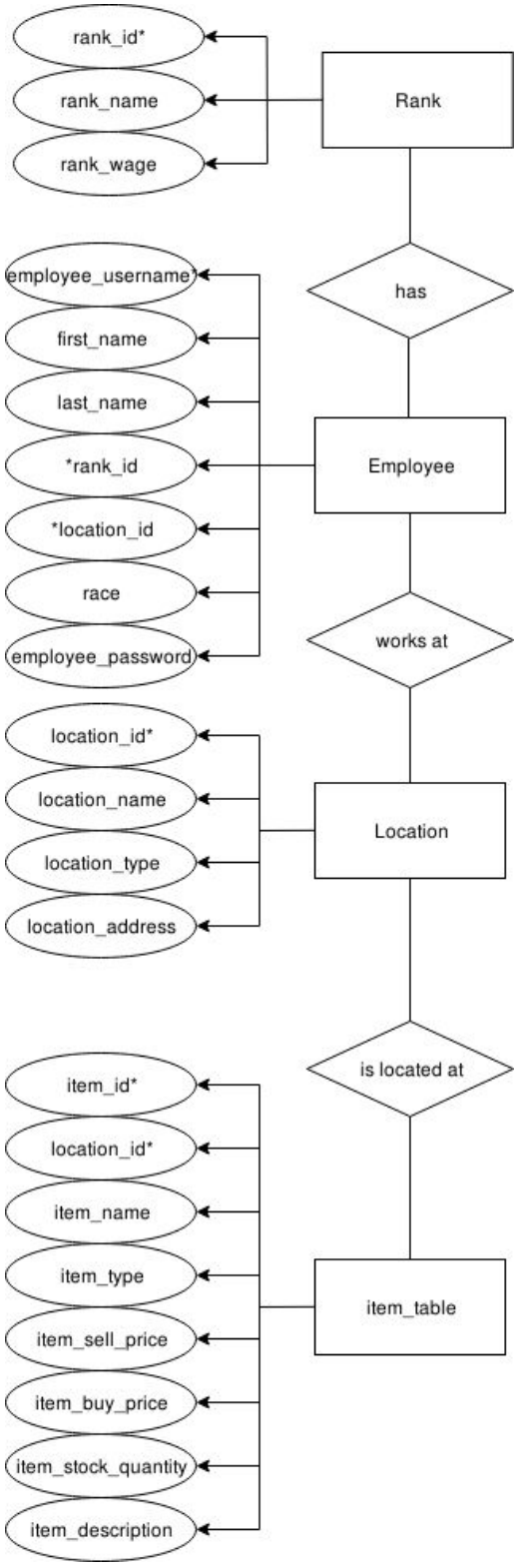
Ye Olde Shoppe Database Website

A Brief Overview

Ye Olde Shoppe is a state-of-the-art virtual online way of shopping for classic to legendary arsenal, gear, elixirs and more! Our group consists of the true Baron himself, Delroy Fong, the One-True-King Jim Bui and the King-of-the-North Cleston Sanders. The purpose of this website / project / database is to create an easily accessible method of viewing what items are available and where they are located, so no more do consumers need to wearily travel to each and every location to find their much needed equipment and grocery.

We hope to provide a more dynamic graphical user interface in the future, as well as utilizing modern technologies such as JSTL, JSP, JDBC, JavaScript servlets and cookies in the web application, and one day rival Amazon.

The Entity-Relationship Diagram



How The Site Works

This section will go over how the website navigation works. In general, there are two main menus that the user can access - one for an employee who has an account, and one for a guest.

The employee who is logged in can access the employee menu, where they have access to all functions of the website. The guest goes to the guest menu, which only has basic viewing information.

For most pages, there is also a back / previous button, as necessary for user convenience.

Welcome Page

This is the first page that the user encounters. There are two buttons here: one to go to the Guest Menu, and one to continue to the Employee Login.

Guest Menu

The guest menu contains the available functions for those who are not logged in, which consists of only viewing and searching. There are four buttons: one to VIew All Items, one for Search For Item, one for View All Locations, and one to return to the Welcome Page.

Employee Login

On this page, the user can enter employee credentials to log in and continue to the Employee Menu, or continue to the Employee Registration page.

Employee Registration

Here, the user can register to make an employee account necessary for management of the database. The user inputs relevant data, and then reroutes the user back to the welcome page. The username must not be already taken.

Employee Menu

This is the main menu that contains all possible functions of the site. Employee credentials are needed to access this page. The options are: View All Items, Search For Item, Create Item, Remove Item, Update Item, View All Locations, Create Location, Remove Location, View Employees at Location, Update Employee, Delete Employee, View Payroll, and an option to logout and return to the Welcome Page.

View All Items

Here, the user can see all items and the information about them.

Search For Item

Here, the user can optionally select a location and type of item to filter the items. After submitting, the page will display all the items that fulfill the criteria.

Create Item

The user can create an item here to add to a specific location. The Item ID must be unique.

Remove Item

The user can remove an item here by specifying what the Item ID and Location ID is.

Update Item

The user can update items here. The user first specifies what item at which location, whose information is then retrieved from the database and displayed. The user can then edit the information (with the exception of the Item ID and Location ID) and update the database.

View All Locations

The user can view all the locations in the database here.

Create Location

The user can create a new location here by filling out all the data. The Location ID must be unique.

Remove Location

The user can remove a location by specifying the Location ID.

View Employees at Location

Here the user specifies a location and all the employees who work at that location will be displayed.

Update Employee

The user can update employee information here by entering a valid employee username that matches the employee_username parameter in the EMPLOYEE_TABLE. The employee information is displayed in text boxes allowing the user to change the old information and press the update button to store the new information.

Delete Employee

The user can delete a user from the database. We are currently aware of the situation where if a user deletes themselves, they will still be logged in. In the future, we

hope to implement a function to avoid this anomaly.

View Payroll

The user can view the company payroll based on the employee rank.

Tables of the Database

These are the tables located in our database, as described in the ER diagram.

Employee Table

```
CREATE TABLE EMPLOYEE_TABLE (
      employee_username VARCHAR (50) NOT NULL ,
      first_name VARCHAR (50) NOT NULL ,
      last_name VARCHAR (50) NOT NULL ,
      rank_id VARCHAR (50) NOT NULL ,
      location_id VARCHAR (50) NOT NULL ,
      race VARCHAR (50) NOT NULL ,
      employee_password VARCHAR (50) NOT NULL ,
      PRIMARY KEY (employee_username)
) ;

This table holds our employee accounts and any relevant data, including where
they work and rank.
```

Location Table

```
CREATE TABLE LOCATION_TABLE (
      location_id VARCHAR (50) NOT NULL ,
      location_name VARCHAR (50) NOT NULL ,
      location_type VARCHAR (50) NOT NULL ,
      location_address VARCHAR (50) NOT NULL ,
      PRIMARY KEY (location_id)
) ;

This table holds all the location of our stores, and any relevant data.
```

Item Table

```
CREATE TABLE ITEM_TABLE (
      item_id VARCHAR (50) NOT NULL ,
      location_id VARCHAR (50) NOT NULL ,
      item_name VARCHAR (50) NOT NULL ,
      item_type VARCHAR (50) NOT NULL ,
      item_sell_price INT NOT NULL ,
      item_buy_price INT NOT NULL ,
      item_stock_quantity INT NOT NULL ,
      item_description VARCHAR (100) NOT NULL ,
      PRIMARY KEY (item_id , location_id)
) ;
```

```
This table holds the items that we have in the stores.  There are two values
that make up the primary key.  There can be the same item located at multiple
locations.
```

Rank Table

```
CREATE TABLE RANK_TABLE (
      rank_id VARCHAR (50) NOT NULL ,
      rank_name VARCHAR (50) NOT NULL ,
      rank_wage INT NOT NULL ,
      PRIMARY KEY (rank_id)
) ;

This holds the ranks and the wage that comes with the rank.  Every employee has
a rank.
```

Queries of SQL

Our database uses four main types of queries to perform each of its tasks on the Employee Menu and the Guest Menu. The four different types of queries we use are Search Queries, Insert Queries, Update Queries, and Delete Queries. We have multiple pages dedicated to searching our database for item information or employee information, and it's the same for removing, creating, and inserting said items. Below we will explain how each of the different types of pages use these four queries.

Search Pages

We have four pages that rely solely on a search query. Some of these include the view_payroll_page and the item_search_page. The search is used to take a parameter specified by the user and return the results in a neatly formatted table so that they can either edit, view, or remove data. An example of a typical search query we would use for one of these pages is shown below:

```
SELECT *
FROM EMPLOYEE_TABLE AS employ, RANK_TABLE AS rank
WHERE employ.rank_id = rank.rank_id AND employ.rank_id ='<SpecifiedID>'
```

Of course almost every other page in our database has a search query, but these queries are used mainly to validate the information entered by the user rather than to view the subject of the search outright. An example of this type of query is shown below:

```
SELECT *
FROM RANK_TABLE
WHERE rank_id = '<SpecifiedID>'
```

Create Pages

Our create pages use the Insert query to add new information to the database. This query was commonly used on pages such as the create_new_item and create_new_employee. The query would take all of the information entered by the user and put it into the correct table in order. An example of one of the queries is shown below:

```
INSERT
INTO EMPLOYEE_TABLE (employee_username , first_name , last_name , rank_id
, location_id , race , employee_password)
VALUES (? , ? , ? , ? , ? , ? , ?) ;
*The ? stands for the <SpecifiedData>
```

Unlike the search queries, insert queries were not often used in other pages because they are only useful for completely new data. This is where we brought in the next query to help use keep like data and make slight changes.

Update Pages

In order to change old data in already existing subjects we had to find a way to update the table without inserting a whole new item or employee, or deleting the old one and reinserting a new one. The Update Query came in handy for pages like our update_employee_page and our update_item_page. For these pages we would use a simple search query to find an employee or item based on their username or item_id. We would then validate the subject and proceed to update the information already in the tables by displaying it in text boxes on the page for the user to edit. Once the information had been edited the user could hit the "Update" button and the update query would take all of the information in the textboxes and change the information in the tables. An example of this is query is shown below:

```
UPDATE EMPLOYEE_TABLE
SET first_name = ? , last_name = ? , rank_id = ? , location_id = ? , race
= ? , employee_password = ?
WHERE employee_username = ?
*The ? stands for the <SpecifiedData>
```

Remove Pages

Last but not least we needed some way to remove or delete any of the information from the item or employee tables that we did not want or need (just in case we decided to fire someone). So we used Delete queries to remove this unwanted information in our remove_item and remove_employee pages. Like in the update and create pages, we would use a simple search query to find and validate the item or employee we were

deleting. Then we would use a query like the one shown below to remove said subject:

```
DELETE
FROM EMPLOYEE_TABLE
WHERE employee_username = ?
*The ? stands for the <SpecifiedData>
```