CSE 345 Database Project

Calvin Reilly

***Abstract:* The goal of this database project was to test what was learned in the Winter 2017 CSE345 class. This project is meant to be a rudimentary representation of a package delivery service, similar to the outside appearance of enterprises such as UPS and FEDEX; except with limited functionality. A user interface was also included, but that was not the main goal of this project and is therefore very rudimentary.**

**II. Platforms and Requirements**

This project was implemented and tested on a Windows 10 operating system (though that should not discourage the usage on other platforms). The Eclipse IDE for Java Developers (version: 4.5.1.20150917-1200) was used to code the java portions / UI, and JDBC 5.1.41 was used to form connections to the database from the java back-end. The database consists of a locally hosted server running MySQL version: 5.7.17. In order to populate the database a PHP script, running PHP version 7.1.3, was used to pull data from several csv files. While not tested, it is assumed that versions as old as PHP 5 should be sufficient to run the database population script.

**III. Assumptions**

**Package Table**

* I made the assumption that both sender and receiver have an account (are a customer) and there are only one of each
* Weight will always be supplied
* It is assumed that one order comprises of one package that is sent from a singular location and received at a singular location

**Paymethod Table**

* customer\_id refers to the receiver\_id
* payment\_type is either ‘credit’, ‘contract’, or ‘prepaid’. With credit referring to the customer paying with a credit card at that time, contract implies that the cost is debited to the customer’s account, and prepaid implies that the payment has already occurred.
* each package will have a shipping\_cost, it just might be 0 in the case of a prepaid type
* contents and value are only supplied if it Is international or hazardous

**Tracking Table**

* Stops between the start and end location will be defined in the intermediate table

**Intermediate Table**

* The reason describes why the package stopped at a certain point, eg. Layover as well as if it was damaged
* Only the time of arrival at an intermediate location is recorded, it is assumed that turnaround time at an intermediate location is marginal

**Customer Table**

* A customer does not have to be a person, it can be a company too
* Due to the above mention, more descriptive customer data is not included

**Outstanding Fees Table**

* Outstanding fees can be applied in cases of a charge to an account (in paymethod) or other miscellaneous fees charged by the shipper
* The amount owed can be deducted from until it reaches zero, at which point that record is removed from the table
* It is assumed that no interest will be applied to the amount owed until after the due date, at which time it will be accrued monthly until the amount owed reaches zero

**Database**

* It is assumed that the ‘oldtest’ database is being used with a similar user/password combination.
* It is assumed that the local database will be utilizing port 3306 for connectivity

**IV. Entity Relation Diagram**

The picture below contains the entirety of the ERD, it is also contained in the ‘ERDiagram’ Visio file that is included in the submission.



**V. Relational Schemes**

CREATE DATABASE oldtest;

CREATE TABLE customer (

customer\_id int NOT NULL AUTO\_INCREMENT,

email varchar(255),

address varchar(255),

PRIMARY KEY (customer\_id)

);

CREATE TABLE outstandingfees (

outstandingfees\_id int NOT NULL AUTO\_INCREMENT,

customer\_id int NOT NULL,

amount\_owed FLOAT(10,2) NOT NULL,

due\_date date NOT NULL,

interest int NOT NULL DEFAULT 0,

PRIMARY KEY (outstandingfees\_id),

FOREIGN KEY (customer\_id) REFERENCES customer(customer\_id) ON DELETE CASCADE

);

CREATE TABLE paymethod (

paymethod\_id int NOT NULL AUTO\_INCREMENT,

customer\_id int NOT NULL,

payment\_type varchar(255) DEFAULT 'credit',

payment\_cost FLOAT(10,2),

shipping\_type varchar(255) DEFAULT '10 day',

shipping\_cost FLOAT(10,2),

international TINYINT(1) DEFAULT 0,

hazardous TINYINT(1) DEFAULT 0,

contents varchar(255) DEFAULT '',

value FLOAT(10,2) DEFAULT 0,

PRIMARY KEY (paymethod\_id),

FOREIGN KEY (customer\_id) REFERENCES customer(customer\_id) ON DELETE CASCADE

);

CREATE TABLE tracking (

tracking\_id int NOT NULL AUTO\_INCREMENT,

start\_timestamp timestamp DEFAULT CURRENT\_TIMESTAMP,

start\_location varchar(255) NOT NULL,

end\_timestamp timestamp,

end\_location varchar(255) NOT NULL,

PRIMARY KEY (tracking\_id)

);

CREATE TABLE intermediate (

intermediate\_id int NOT NULL AUTO\_INCREMENT,

tracking\_id int NOT NULL,

int\_timestamp timestamp DEFAULT CURRENT\_TIMESTAMP,

int\_location varchar(255) NOT NULL,

int\_reason varchar(255),

PRIMARY KEY (intermediate\_id),

FOREIGN KEY (tracking\_id) REFERENCES tracking(tracking\_id) ON DELETE CASCADE

);

CREATE TABLE package (

package\_id int NOT NULL AUTO\_INCREMENT,

reciever\_id int NOT NULL,

sender\_id int NOT NULL,

tracking\_id int NOT NULL,

paymethod\_id int NOT NULL,

weight\_id FLOAT(10,4) NOT NULL,

PRIMARY KEY (package\_id),

FOREIGN KEY (reciever\_id) REFERENCES customer(customer\_id) ON DELETE CASCADE,

FOREIGN KEY (sender\_id) REFERENCES customer(customer\_id) ON DELETE CASCADE,

FOREIGN KEY (tracking\_id) REFERENCES tracking(tracking\_id) ON DELETE CASCADE,

FOREIGN KEY (paymethod\_id) REFERENCES paymethod(paymethod\_id) ON DELETE CASCADE

);

**VI. Description of Code**

As I am not as experienced in java as I am in PHP, I decided to write the database population code as a PHP script which populates one database at a time when given an appropriate csv file. Some other benefits of using PHP are that it is native to a large number of operating systems and it is most likely what a MySQL web server would be running. The script takes two arguments, a csv file and the desired table to output the contents into. Being that I did not want to set up a server for this project, the user interface and its corresponding queries are done using java.swing and java.jdbc respectively. The UI portion contains a single frame with multiple fields and buttons for different functions. Button events run said functions and each handler will open (and close) its own database connection. I chose to go with this implementation due to time constraints, something which will be covered in the section on future work.

**VII. Conclusion**

This project was a fantastic opportunity to create a standalone system from scratch, and to see how different languages send and receive data from a MySQL database. It was also interesting to see the affects that a typed language had on the overall lines of code compared to an un-typed language, as well as error checking and enforcement of an input’s type. The difference between PHP and java was particularly pronounced when it came to the steps required to open and close a MySQL connection as well as run queries, as java has substantially more lines of code. To conclude, it is easy to see that despite the ease of set up regarding the UI in java, it is not an optimal language in terms of lines of code efficiency; and while PHP may require more setup it appears to be easier in regards to data insertion and withdrawal.

**VIII. Future Work**

As referenced several times during this paper, I am unhappy with the sloppiness of the java code that I wrote and give time I would like to fix that. I might also look into setting up a PHP server (maybe apache) and changing all the code over to PHP as I am more comfortable on that language. I would also like to include more functionality in the UI as well as more pages in order to meet the need of different types of users.

**IX. Appendix**

The script to populate the database will be included in this folder under the name ‘populate\_db.php’. Also included are various csv files that are named after the corresponding table that they fill, when run with the script. An example command would be ‘$php populate\_db.php tracking.csv tracking’ provided the csv is in the same directory as the script.