Imex Cargo Database design and UI recommendations

Project By: Mansi Nagraj,
Saish Pai,
Rueben Philip,
Apurva Kalyankar
Prof: Himlona Palikhe

Contents

1. Summary

Imex Cargo is a woman owned air cargo general sales agent company located in Boston, MA. The project focusses on recommending and developing solutions for Imex cargo to store the customer and transactional data while maintaining the data consistency. The proposed solution consists of 2 segments, first building a UI where the customer or the company agents can key in the required information in the desired format and second a database where this information can be stored to be retrieved for future analysis.

2. Propose Solution

Initially when we discussed the project it focused on developing a database for the company to store the customer records and daily order transactions. This currently was done using an Excel which resulted in data inconsistency owing to the input methods and thus required data cleaning as an additional step prior to every analysis. To handle this issue, we recommended building a UI which could handle the data consistency and serve as an input to the database.

UI:

Owing to the lack of time and resources to meet the project deadline we decided to go forward with a basic UI, a form, which could be sent to the customer or filled by a company representative capturing all the necessary details required by the company. A google form was built for this. Factors like simplicity, ease of access and data security were considered while selecting this option for developing the form. The link to the form can be sent via email, used as a signature in the sales representatives email or be embedded in the company's website. The collaborators can access the form through this link. While the form serves the basic purpose, we tried exploring other options to improve the look and feel of the form. A similar form development service was provided by a company called polydojo, the UI we built using this application looked better but data security and data delivery options to be used as an input for the database were concerns which led us to reverting back to google forms.

Database:

The schema for the database was built using Toad data modeler. This software helps us built the Entity relationship diagram (ER diagram) with all the attributes and constrains for the tables. It also helps us generate a SQL script which can be imported in the required database editor enabling us to build the entire schema. The ER diagram is shown in the fig 1.0 below. The ER diagram, SQL script and the model generated using Toad are available on git hub repository (Imex cargo project) from where it can be cloned for interpretation and future analysis. For the project we selected to go with MySQL database according to the recommendations provided. We recommend using the MySQL Workbench UI to interface with the database as it has some user friendly features to develop the stored procedures (SP), functions and views etc. We have built a basic SP in the database which provides the order details of a customer by taking the first name as the input. Also, there is a master view to

visualize the most relevant columns across all the tables, the secondary purpose of this view is to be used with SP's and improve the query performance.

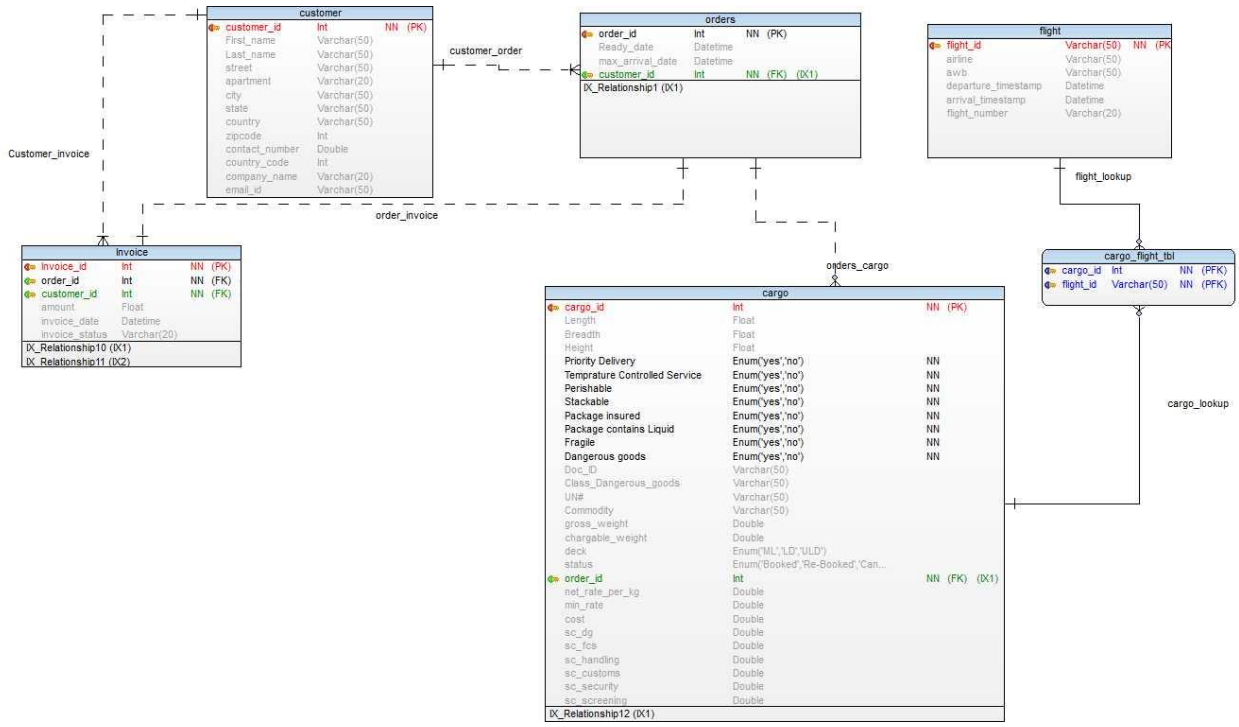


Fig 1.0: ER diagram generated using Toad data modeler

3. Implementation

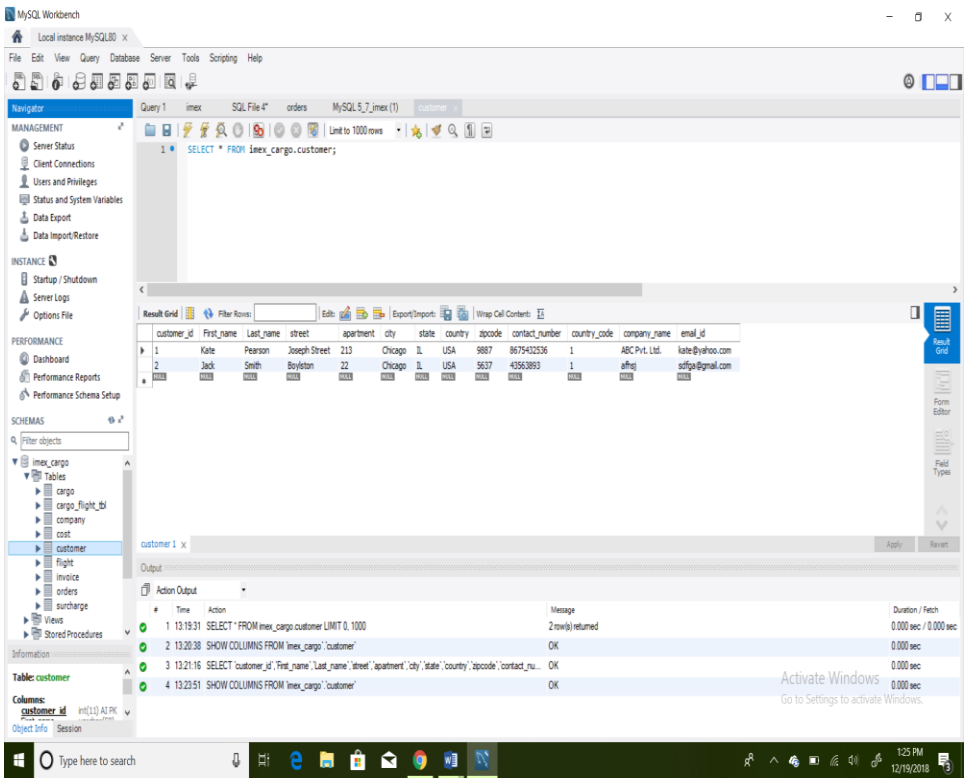
Data Entry techniques:

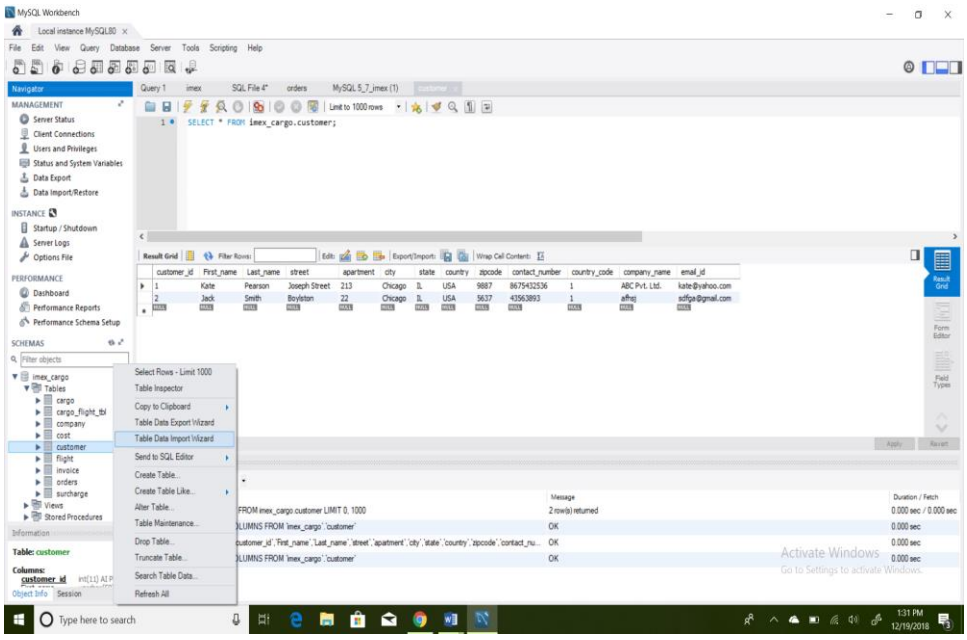
There are a couple of different ways to enter data in the MY-SQL database, 2 of them are listed below:

1. Table import/export wizard
2. Using SQL Queries

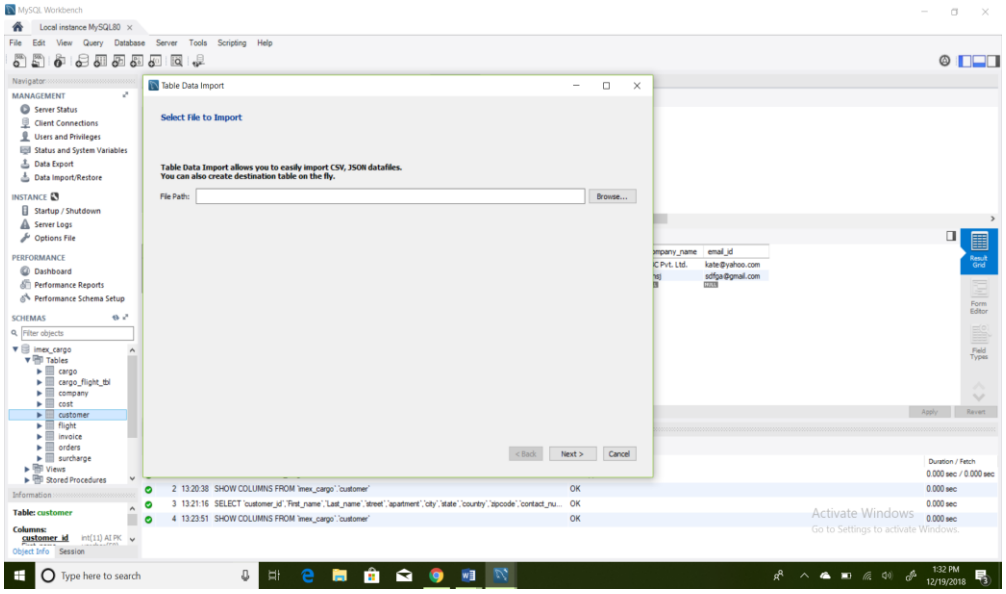
Steps:

1. Export the data from the table into an excel-sheet.
2. Make the necessary additions in the excel-sheet, assigning a unique customer_id to each customer, as it is the primary key in the customer-table in the database and save the file with a .csv extension using the Save-As option from the file-menu.
3. Import the updated excel-sheet into the database, using table import-wizard.

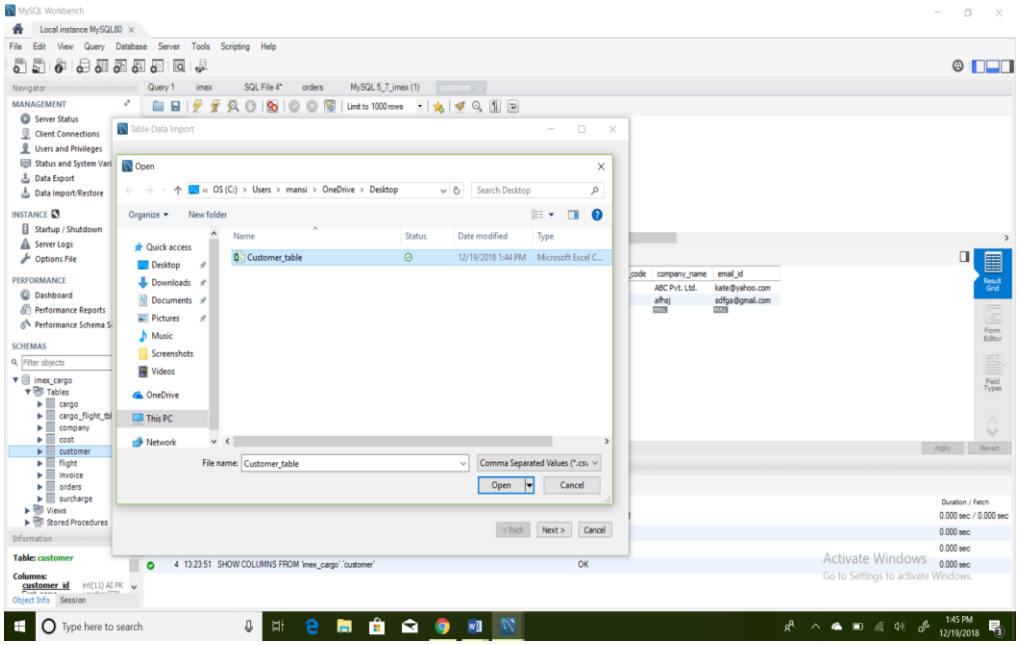




4. Provide the path of the updated excel(.csv) using Browse option.
   (Note: The wizard caches the path once provided and may not need to be changed every-time the file has to be selected. However, if the file is being kept at a different location every-time, the new path can be entered.)



5. Select the csv file that must be imported and click on ‘Open’.



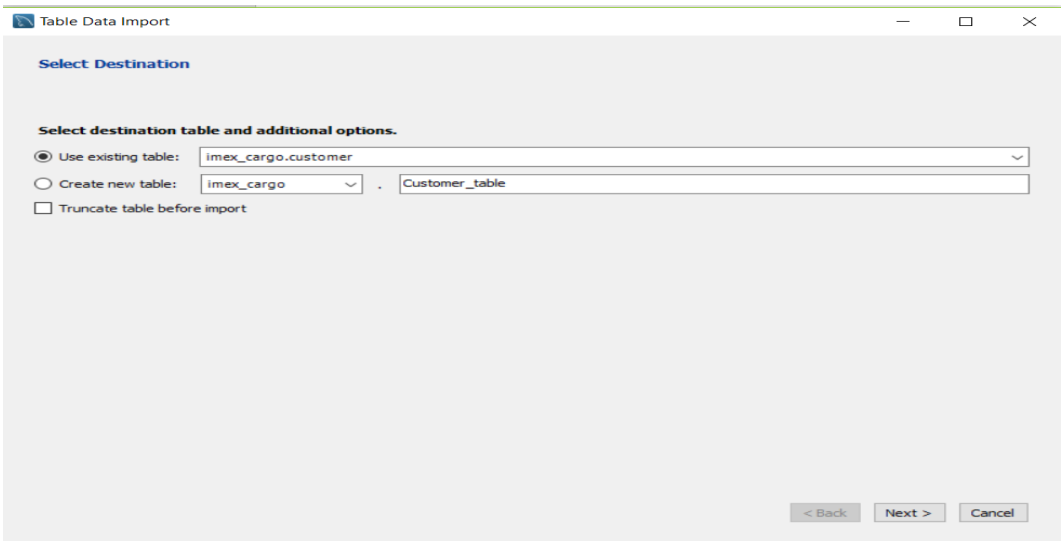


Table Data Import
Select Destination
Select destination table and additional options.
Use existing table:
imex_cargo.customer
Create new table:
imex_cargo
Customer_table
Truncate table before import
< Back
Next >
Cancel

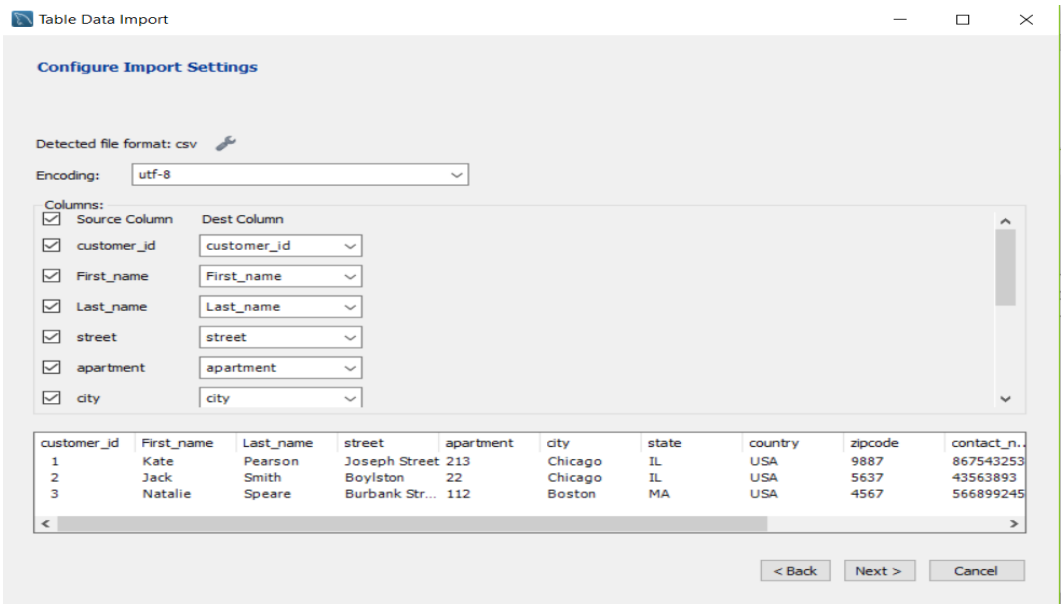


Table Data Import
Configure Import Settings
Detected file format: csv
Encoding:
utf-8
Columns:
Source Column
Dest Column
customer_id
customer_id
First_name
First_name
Last_name
Last_name
street
street
apartment
apartment
city
city
customer_id First_name Last_name street apartment city state country zipcode contact_n..
1 Kate Pearson Joseph Street 213 Chicago IL USA 9887 867543253
2 Jack Smith Boylston 22 Chicago IL USA 5637 43563893
3 Natalie Speare Burbank Str... 112 Boston MA USA 4567 566899245
< Back
Next >
Cancel


Table Data Import
Import Results
File C:\Users\mansi\OneDrive\Desktop\Customer_table.csv was imported in 0.132 s
Table imex_cargo.customer has been used
1 records imported
< Back
Finish
Cancel

The newly entered records will be added to the table.

Updating records in the existing database:

Use the following query to update the value of a column in the table:

Update *table_name* set column_name = new value where *condition*;

Example to update the company_name in Customer table:
Update imex_cargo.customer set company_name = 'Fidelity Group' where customer_id=2;

4. Project status and Conclusion

Currently, we have the google form as the UI to collect consistent input data and a robust database to store the same. The database has a stored procedure and view for easier analysis. The backend and frontend amalgamation will automate and improve the project significantly. We also have a couple of recommendations listed below which are crucial for the project utilization.

5. Recommendations

1. Keep all the csv's in the same location so that the path in the table import wizard need not be changed every-time and will reduce effort and time.
2. Run MY-SQL in Strict-SQL mode (In case the strict SQL mode is enabled, trying to insert an invalid ENUM (data-type used for some columns in the Cargo table) value will result in an error.)
   To enable the Strict-SQL mode, run the below mentioned query from the editor window:
   ```
   set global sql_mode='STRICT_TRANS_TABLES';
   ```
3. Data-type "ENUM" is not case-sensitive which implies that it can accept all possible combinations of the word yes (eg. YES, YEs, YeS, Yes…etc.)
4. Develop a UI which can be hosted on the company's website this data can be collected directly to the database, this will save the efforts of loading the .csv files
5. Add triggers on the tables so that the data collected from the website can be loaded directly making physical intervention irrelevant.

6. Appendix

Table-name: *Cargo*	
Column-Name	**Data-Type**
cargo_id	int
Length	Float
Breadth	Float

Height	Float
Priority delivery	ENUM (‘Yes’/’No’)
Temperature Controlled service	ENUM (‘Yes’/’No’)
Perishable	ENUM (‘Yes’/’No’)
Stackable	ENUM (‘Yes’/’No’)
Package insured	ENUM (‘Yes’/’No’)
Package contains liquid	ENUM (‘Yes’/’No’)
Fragile	ENUM (‘Yes’/’No’)
Dangerous goods	ENUM (‘Yes’/’No’)
Doc_Id	Varchar (50)
Class_Dangerous_goods	Varchar (50)
UN#	Varchar (50)
Commodity	Varchar (50)
Origin	Varchar (50)
Destination	Varchar (50)
Gross_weight	Double
Chargeable_weight	Double
Deck	Enum(‘ML’,’LD’,’ULD’)
Status	Enum(‘Booked’,’Re-Booked’,’Cancelled’,’Quote-Request’,’on-hold’)
Order_id	int
Net_rate_per_kg	Double
Min_rate	Double
Cost	Double
Sc_dg	Double
Sc_fcs	Double
Sc_handling	Double
Sc_customs	Double
Sc_security	Double
Sc_screeening	Double
Table-Name: *Cargo-Flight Table*	
Cargo_id	Int
Flight_id	Varchar (50)
Table-Name: *Customer*	
Customer_id	Int(11)
First_name	Varchar (50)
Last_name	Varchar (50)
Street	Varchar (50)
Apartment	Varchar (20)
City	Varchar (50)

State	Varchar (50)
Country	Varchar (50)
Zipcode	Int(11)
Contact_number	Double
Country_code	Int(11)
Company_name	Varchar (20)
Email-id	Varchar (50)
Table-Name: *Orders*	
Order_id	Int (11)
Ready_Date	Datetime
Max_arrival_date	Datetime
Cutomer_id	Int (11)
Table-Name: *Invoice*	
Invoice_id	Int (11)
Order_id	Int (11)
Customer_id	Int (11)
Amount	Float
Invoice_date	Datetime
Invoice_status	Varchar (20)
Table-Name: *Flight*	
Flight_id	Varchar (50)
Airline	Varchar (50)
Awb	Varchar (50)
Departure_timestamp	Datetime
Arrival_timestamp	Datetime
Flight_number	Varchar (20)