Loan data dashboard Report

DS 5420-01
Data Management Systems(2020S)

Jiayi Wang
Zekun Wang

04-10-2020

Table of Contents

1. Introduction

We built up an interactive live dashboard using Shiny R, and we use MySQL service as the back-end data source.

Both of us are interested in the financial industry and we chose this loan dataset to do our project. We downloaded it from Kaggle, and it is originally from the Lending Club website. And Lending Club is an American peer-to-peer lending company that operates personal loans. We believe by creating a database, and a simple, easy-to-use application with financial data might be a very good practice to help us build a better understanding of lending.

This dataset includes loans issued through 2007 -2015. It also has some additional features, including credit scores, the number of finance inquires, address including zip codes, states, and collections among others. And the original size of this dataset is 1.8GB, with 2,260,668 observations and 145 attributes. We noticed the running speed is pretty low when we run all of the data. So, we decided to only choose 46 attributes, and we only took the first million of the data. Our final dataset is 234.9MB. We want to use this dataset to answer questions such as loans issued by region, most popular job titles, analysis by income category and the risky side of the business.

The web framework we chose is Shiny R. Firstly, the Shiny R package is user friendly. And Zekun has some experience with Shiny R for his previous projects. We aim to build visualizations for attributes in the table to enable users to have an overview of the loan dataset, detailed distribution for attributes in each table, as well as manipulation of the dataset, including insert, delete and update.

2. Final Implementation

2.1a Use case

This live dashboard is for everyone who has an interest in taking a look at the loan across the states. By making selections, they can have access to details of the loan, borrower, bank accounts, bank card, inquires and installment, with some visualizations we provided. Users should be able to view, insert, delete, update data into the database as well.

2.1b Technologies

We use Mysql database, R as our primary language, and Shiny(R), which is an open-source from RStudio as our web framework.

2.2 Final database design

a. Normalization

We used the BCNF form for our database. When we first approached this dataset, we noticed that the id is missing. But it is okay since it will not affect the result as it is not

among the independent features. We generated id for each observation based on the length of this data. We then did functional dependency analysis on each table based on their description and selected id, which is a unique Lending Club ID for the loan listing as our primary key for all of the tables. And when we looked for functional dependencies, we noticed that this dataset has been cleaned and de-identified. For example, in the bank_accounts table, there is an attribute called months since most recent bankcard account opened, instead of the bank account opening time. We are unable to use bankcard id to represent a unique bankcard account.

b. Final database design

Based on the understanding of data and business rules, we divided our dataset into six tables, namely, loan, borrower, bank_accounts, bankcard, inquires and installment, and we have 46 attributes in total.

We used CHECK as constraints on tables. And we also included three functions: Views, Stored Procedures, Triggers.

Views:

1. I created this view with one parameter to get the loan information when the disbursement method is CASH.
2. I created this view with multiple parameters to show the borrowers' state, employment title, employment length and annual income by sorting the records in descending order.

Stored Procedures:

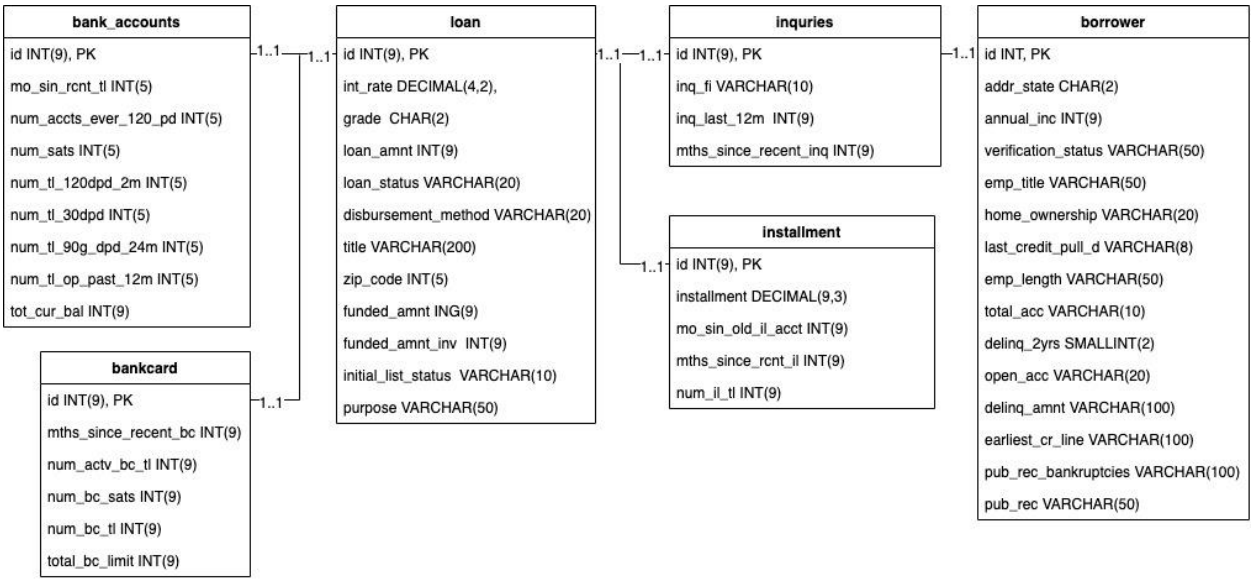
1. I stored this prepared SQL with multiple parameters to get the information for borrowers when the borrowers are from New York State and the borrowers' title is Tech.
2. I stored this prepared SQL with one parameter to get the loan information when the LC assigned grade is A.

Triggers:

1. I created an after DML trigger called after_installment_insert. MySQL will fire after the completion of the INSERT operation on the installment table. Once it completes inserting into the Installment table, it will start inserting into the Reminder table. I set up the condition that when the installment in Installment is null, it will insert into the Reminder table with values to remind the user to update installment.
2. I created a before DML trigger called bank_accounts_before_update. MySQL will fire this trigger before the UPDATE operation is executed. When users want to update the value, if the total current balance in the bank_accounts table is larger than 1,300,000, it will respond with error code 1264, saying the total current balance is more than 1300000. And when the total current balance is less than zero, it will respond with error code 22003 and error message.

Besides, we used DML(SELECT, INSERT, UPDATE,DELETE, CALL) and DDL (CREATE, DROP)as well.

c. UML diagram



2.3 Testing

We tested the features (views, stored procedures, triggers) and constraints (check) to make sure the queries are running without giving errors. We did this QA in MySQL and the front-end. We also tested using SET id in the dashboard.

3. Illustration of Functionality

We first set up the frame includes the user interface and server. We then went over the database and decided to present three major functions. Namely, the tables, attributes in each table, and then manipulation (update, insert, delete). We also include the resource of this dataset so that users can download it if interested.

The functions include three major parts.

3a. Overview of loan data (original table)

Step1: Users choose the table and the number of rows they want to see.

Step2: Shiny will integrate the users' input, including table and row number as SQL queries. RMySQL will transfer the SQL queries to MySQL database. MySQL database will then return the table to Shiny. Shiny presents the result as the user interface.

3b. Visualization of attributes in each table.

Step1: Users choose the table they interested in.

Step2: Users choose the attribute and the number of rows they want to see.

Step3: Shiny will integrate the users' input, including table, attribute and row number as SQL queries. RMySQL will transfer the SQL queries to MySQL database. MySQL database will then return the attribute content to Shiny. Shiny generates histograms based on the result.

3c. We provide UPDATE, INSERT, DELETE on tables for users.

UPDATE:

Step1: Users select the table they want to modify.

Step2: Users set the condition by choosing the attribute and its value.

Step3: Users select the attribute they want to be modified and the new value he/she wants to assign.

Step 4: Shiny will integrate all of the input as SQL queries.

Step 5: RMySQL will transfer the SQL queries to MySQL database.

Step6: MySQL database will then return the number of rows affected to Shiny.

Step7: Shiny will present this information with the update result. If this action violates the constraints, it will respond with an error.

INSERT:

Step1: Users select the table they want to modify.

Step2: Users input the values they want to insert for each attribute.

Step3: Shiny will integrate all of the input as SQL queries.

Step4: RMySQL will transfer the SQL queries to MySQL database.

Step5: If the action is successful, Shiny will how many rows are affected as well as the insert result. If this action violates the constraints, it will respond with an error.

DELETE:

Step1: Users select the table they want to modify.

Step2: Users set the condition by choosing the attribute and its value.

Step3: Users select the attribute they want to be deleted.

Step4: RMySQL will transfer the SQL queries to MySQL database.

Step5: MySQL database will then return the number of rows affected to Shiny. If this action violates the constraints, it will respond with an error.

4. Summary Discussion

1. Challenging parts
   First of all, determining dependencies took me a big chunk of time. I ran the data in R to find how many unique observations for each table to make sure our dependencies are correct.

   Secondly, we are supposed to write features in the database. And I want to write something to provide meaningful insights. I spent some time thinking about this.

   Besides, When Zekun is working on the connection between the front end and database, he ran into a lot of different bugs. However, the Shiny R is not that friendly since it does not provide error code. Thus, he has to go over the code line by line to debug. Additionally, the logic behind the scene took him time to clear.

   Last but not the least, keeping everything clean and tidy, keeping our steps on the right track is challenging as well as rewarding.

2. Jiayi built up the database, wrote the report and created the demonstration video. Zekun put a lot of time building up the dashboard. He also helped out my errors when building up the databases. Our work is very effective by dividing the work in this way.