# FINAL PROJECT
# BANKING ENTERPRISE DATABASE

## Introduction

Each student should complete this project individually. You are going to MySQL to perform some queries against a database. The schema and sample data of the database are provided. The goal of this project is to provide a realistic experience in the implementation, operation, and maintenance of a relational database. You will design a relational database schema and implement it on MySQL. Then, you will load data into your database, create queries, and update transactions.

## The Bank Database

You have been approached by a bank for the design and implementation of a relational database system that will provide information on the branches, customers, accounts, and employees. The system will be used mainly by the staff and bank administration. Assume that the following requirements were collected for this application:

- The bank is organized into *branches*. Each branch is located in a particular city and identified by a unique number and name. The bank monitors the assets of each branch. Branch name is the name of a branch of the bank.
- Bank *customers* are identified by their customer ID values. The bank stores each customer's name and the street and city of which the customer resides. Customers may have accounts and are permitted to take out loans.
- Bank employees are identified by their employee ID values. The bank administration stores the name and telephone number of each employee. The bank also keeps track of the employee's start date and length of employment.
- The bank offers two types of accounts - *savings* and *checking*. More than one customer can hold one account, and a customer can have more than one account. Each account is assigned a unique account number. The bank maintains a record of each account's balance.
- A loan originates at a branch and can be held by one or more clients. A loan is identified by a unique loan number. For each loan, the bank keeps track of the loan amount and interest rate.

## Logical Design

The full set of normalized tables for the Bank Database is as follows:

## Schema

**ACCOUNT** (account_number , branch_number , balance, type)

**BRANCH** (branch_number, branch_name , branch_city, assets)

**CUSTOMER** (customer_id, last_name , first_name , email_address, encrypted_password, street, city, state, zip_code, country, phone_number)

**LOAN** (loan_number, branch_number , amount, interest_rate)

**CREDIT** (customer_id, account_number)

**BORROW** (customer_id, loan_number)

**EMPLOYEE (**emp_id, lname, fname, initial ,hire_date, phone_number, branch_number)

## PART 1: Implement the Database [50 points]

1. Create the tables for the bank database using MySQL DBMS. To your report, you will add a list of the CREATE TABLE statements for the bank database. Specify as many constraints (key, referential integrity) as you can in the relational schema. Choose appropriate data types for each attribute.
   a. Define the primary key, foreign key, NOT NULL, CHECK and UNIQUE constraints in the CREATE TABLE statement. If not possible, use ALTER TABLE statement to add a constraint.
   b. If a table does not have a foreign key, leave the entry blank. (Note: Some tables have a composite primary key. Identify all composite key attributes for such tables.)

| Table | Primary key | Foreign key | Table Referenced |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

2. Load the records provided to you into each of the tables that you have created. Your data should be kept in a file so that it can be easily reloaded.

## PART 2: SQL Queries [50 points]

You should run a number of test queries to see that you have loaded your database in the way you intended. The second part of the project is to apply certain update transactions and retrieval queries.

Write SQL queries for the following and execute them:

1. Display the last names and cities of all borrowers.
2. Display loan data, ordered by decreasing amounts, then increasing loan numbers.
3. Display the average balance of all accounts.
4. Display the names of branches, having at least one account, with average account balances.
5. Display the names of branches, having at least one account, with average balances of accounts at each branch, if that average is above 500.
6. Display the names and cities of customers who have a loan at the Alameda branch.
7. Display the numbers of accounts with balances between 500 and 1200.
8. Display the names of customers on streets with names ending in "Lane."
9. Display all customer IDs with both accounts and loans at the Alameda branch. Use a subquery.
10. Display all customer IDs with an account but not a loan at the Fresno branch. Use a subquery.

11. Display all customer IDs with accounts at a branch where Smith has an account. Use a subquery.
12. Display the names of branches whose assets are greater than the assets of all branches in Palo Alto.
13. Display the last names of customers with an account but not a loan at the Orange branch.
14. Display the names of customers at the Orange branch in alphabetical order.
15. Display the numbers of branches, having at least one loan, with the number of customers.
16. Display the name(s) of the branch(es), having at least one account, with the largest average balance of all customers in San Francisco. Use JOIN (JOIN…ON)
17. For each branch, retrieve the branch name, number of employees in the branch, and their hire dates.
18. Display the list of all branches with the total number of employees in each branch.
19. Create a view named *branch_summary*. This view should return summary information about each branch. Each row should include branch_number, account_count (the number of accounts in the branch). Write a SELECT statement that returns all the columns from the *branch_summary* view.
20. Write a script that creates and calls a stored procedure named *insert_account*. First, code a statement that creates a procedure that adds a new row to the Account table. To do that, this procedure should have three parameters for the account number, branch number, and balance.

    Code at least two CALL statements that test this procedure (Note that this table doesn't allow duplicate account number).

## Submission:

- Documentation describing any assumptions you have made
- One or more SQL/Spool files of the SQL part of the project, including creating the tables and the query results. You will need to label your project with your first initial, last name, and the name of the project.
- Zip the files to upload to Canvas (yourname_project_part1.zip and yourname_project_part2.zip).
- The files that you submit should be sent via Canvas latest by 11:59 pm of the due date.

## Due Dates:

- Due Date for Part 1: **Monday, May 1st, 2017.**
- Due Date for Part 2: **Monday, May 12th, 2017.**

## Grading

The grade for this project will be broken down as follows:

- Part 1 – Create the Database / Populate Relations (50 points)
- Part 2 – Query the Database (50 points)

## Important Note:

Plagiarism is not permitted and will result in a grade of zero.

## Late policy:

For Part 1, there will be a 5% deduction for each day that it is late. Part 2 will not be accepted late.