

A Simple Database for a Financial Institution

By

Yassin Sebastien Bah 40077524

Joel Dusablon Sénécal 40035704

Feng Zhao 40021856

Alireza Sari 40032394

A project submitted in partial
fulfillment of the requirements
of COMP 353

Concordia University

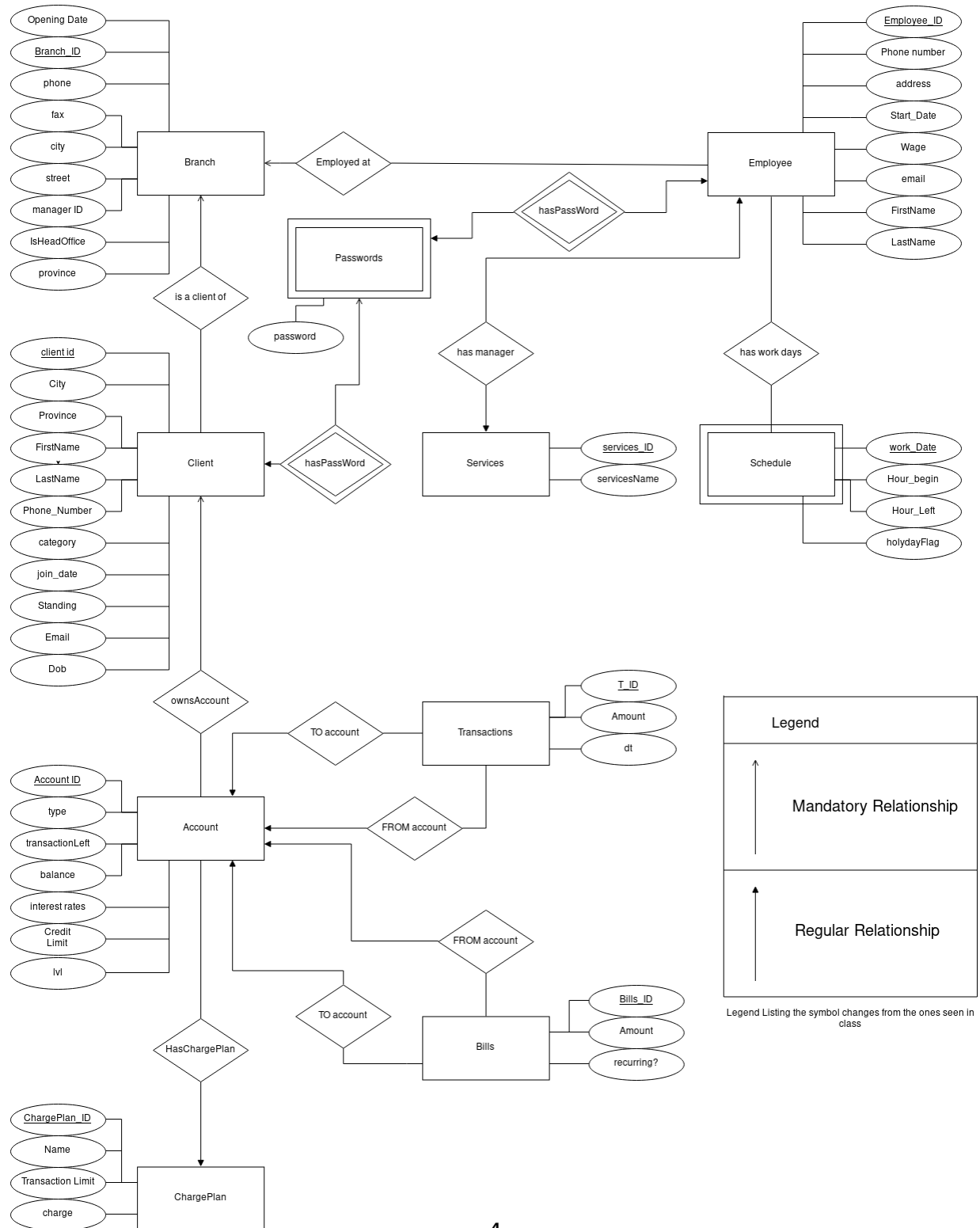
December 2018

Table of Contents

ER Diagram.....	4
Third Normal Form (3NF) Proof.....	5
Branch Relation.....	5
Functional Dependencies.....	5
Employee Relation.....	5
Functional Dependencies.....	5
Schedule Relation.....	6
Functional Dependencies.....	6
Services Relation.....	6
Functional Dependencies.....	6
Client Relation.....	7
Functional Dependencies.....	7
Account Relation.....	7
Functional Dependencies.....	7
Charge Plan Relation.....	8
Functional Dependencies.....	8
Transaction Relation.....	8
Functional Dependencies.....	8
Bills Relation.....	8
Functional Dependencies.....	8
Employee Login Relation.....	9
Client Login Relation.....	9
Discussion.....	10
Global Assumptions.....	10
Branch.....	10
Bills.....	10
Clients.....	11

Administrators.....	14
Abstractions.....	15
Security.....	15
Contributions.....	16
Joel.....	16
Feng.....	16
Yassin.....	17
Alireza.....	17

ER Diagram



Third Normal Form (3NF) Proof

Branch Relation

Branch(BranchID, OpeningDate, Phone, Fax, City, Street, managerID, IsHeadOffice, province)

(Assumption : one branch has only “main service phone number” and has only one location.)

Functional Dependencies

BranchID → openingDate, BranchID → OpeningDate, BranchID → phone, BranchID → Fax, BranchID → City, BranchID → managerID, BranchID → isHeadOffice, BranchID → Provence.

Since for all FD's over branch, the left hand side is BranchID, which is the key of this schema. Branch schema is in BCNF, thus also in 3NF.

Employee Relation

Employee(EmployeeID, Phone, address, StartDate, Wage, BranchID, FirstName, LastName, Email)

(Assumption: one Employee can only have one “working phone number” and has one “current address” also, he may only works for/reside in one Branch, has only one “working email”)

Functional Dependencies

EmployeeID → *phone*, *EmployeeID* → *address*, *EmployeeID* → *startDate*,
EmployeeID → *startDate*, *EmployeeID* → *wage*, *EmployeeID* → *BranchID*,
EmployeeID → *FirstName*, *EmployeeID* → *lastName*, *EmployeeID* → *email*.

Since for all FD's over Employee, the left hand side is BranchID, which is the key. Employee schema is in BCNF, thus also in 3NF.

Schedule Relation

Schedule(EmployeeID, Date, HourBegin, HourLeft, isHolyday)

Functional Dependencies

EmployeeID, Date → HourBegin, EmployeeID, Date → HourLeft, EmployeeID,
Date → isHolyday

EmployeeID and date make up the key and are on the LHS of all the related FD's. Therefore the relation is in BCNF thus in 3NF.

Services Relation

Services(ServicesID, servicesName, ManagerID)

(Assumption: the managerID linked with a specific service is the person in charge of said service within the bank)

Functional Dependencies

ServicesID → *servicesName*, *ServicesID* → *ManagerID*

Since all the LHS are the set key, it is in 3NF

Client Relation

Client(client_id, firstName, lastName, city, province, dob, join_date, standing, email, phone, category, branch_id)

(Assumption: one client has only one “bank contacting email” and one “bank contacting phone”)

Functional Dependencies

client_id → *firstName*, *client_id* → *lastName*, *client_id* → *city*, *client_id* → *province*, *client_id* → *dob*, *client_id* → *join_date*, *client_id* → *standing*, *client_id* → *email*, *client_id* → *phone*, *client_id* → *category*, *client_id* → *branch_id*;

Since the LHS are all the key, this schema in BCNF, thus also in 3NF.

Account Relation

Account(account_id, client_id, account_type, chargePlan_id, balance, credit_limit, interest_rate, lvl, transactionLeft)

Functional Dependencies

$account_id \rightarrow client_id$, $account_id \rightarrow account_type$, $account_id \rightarrow charegPlan$,
 $account_id \rightarrow balance$, $account_id \rightarrow credit_limit$, $account_id \rightarrow interest_rate$,
 $account_id \rightarrow lvi$, $account_id \rightarrow transactionLeft$

Since the LHS are all the key, this schema is in BCNF, thus also in 3NF.

Charge Plan Relation

ChargePlan(chargePlan_id, option_name, draw_limit, charge_value);

Functional Dependencies

$chargePlan_id \rightarrow option_name$, $chargePlan_id \rightarrow draw_limit$, $chargePlan_id \rightarrow charge_value$

Since the LHS are all the key, this schema is in BCNF, thus also in 3NF.

Transaction Relation

Transaction(tid, account1_id, account2_id, amount, dt)

Functional Dependencies

FDs $tid \rightarrow account1_id$, $tid \rightarrow account2_id$, $tid \rightarrow amount$, $tid \rightarrow dt$

Since the LHS are all key, this schema in BCNF, thus also in 3NF.

Bills Relation

Bills(bill_id, amount, account1_id, account2_id, recurring)

Functional Dependencies

bill_id → amount, bill_id → account1_id, bill_id → account2_id, bill_id → recurring

Since the LHS are all key, this schema is in BCNF, thus also in 3NF.

The design of this relation is further explained in upcoming sections.

Employee Login Relation

EmployeeLogin(employee_id, psw)

The relation is in BCNF because the key determines the other attribute.

Client Login Relation

ClientLogin(client_id, psw)

The relation is in BCNF because the key determines the other attribute.

Discussion

This section will split the project up into a number of parts and will evaluate the associated assumptions that were made during this project. The parts are as such: global logic of the bank, client pages and administrative pages.

Global Assumptions

First up is the global logic of the bank. In order to identify the main branch or the head office, we created a flag in the Branch table that is set to true only for the head office. The president of the bank is considered to be the manager of the head office. The managers of the branches are also considered employees of the bank and have an entry in the Employee table.

Branch

In order to keep track of the banks earnings, clients and accounts are created for each branch. The setup makes it possible to send any fees that a client incurs to the account of his branch. A transaction of the fee is also created. The transaction includes a date and time which allows the calculation of the profits of each branch for a window in time. Thus, this approach to processing fees is conducive to an easy compilation of annual profits.

Bills

Bills are between a sender and receiver account. Importantly to note, bills are not processed immediately. The bills get processed at a consistent frequency by the bank and the bank takes a fee on a successful payment. A crontab job is setup on the server to run monthly. This periodic PHP script looks at all the bills and performs the transfer of money only if the sender account has sufficient funds. If the bill is recurrent, the bill will be left in the database. In the case that the account had insufficient funds two things may happen. If the bill is not recurrent, then the bill will remain unpaid and in the Bills table. In the situation where it is recurrent, then the server will duplicate the

information in the bill and add it to the Bills table (with the recurrent tag set to false). The logic behind this is that if you failed to pay a bill this month, you will owe the receiver two payments the next month. Of note, clients can setup bills for themselves and administrators have the power to create and modify the bills of all clients.

Clients

For the clients, they have to log in from the home page. Upon successfully entering their client id and their password, they are redirected to the client hub. This hub lists all their information to them. The hub is their launching point to perform all the actions they need.

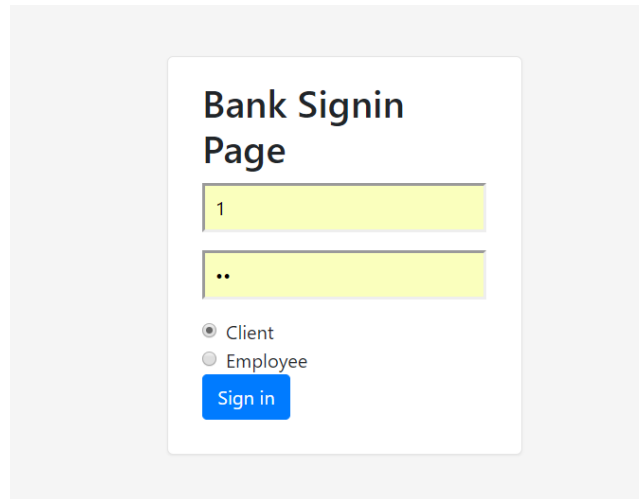
A mockup of a 'Bank Signin Page'. The page has a white background with a light gray border. At the top, the title 'Bank Signin Page' is displayed in a bold, dark font. Below the title are two yellow input fields. The first field contains the number '1', and the second field contains two dots '..'. Under the input fields, there are two radio button options: 'Client' (which is selected) and 'Employee'. At the bottom of the form is a blue button with the text 'Sign in' in white.

Illustration 1: Log in Page

At the top of the page, the clients may see their personal information which includes their name, their address, their date of birth and other fields. Under their personal information, there is a table with all their accounts. In this table, the Ids , the levels and the current balances of their accounts are listed. From this table, clients are able to dig deeper into the state of their specific accounts.

Client Hub

Client Info

Client ID: 1
Associated with Branch # 2
Name: Ricky Martin
Location: Montreal, Quebec
Date of Birth: 1980-02-22
Joined on 2003-01-12
Standing is *not good yet*
Email: bno@gmail.com
Phone: 514-222-3456

Accounts Info

Select	Account Number	Level	Balance
<input checked="" type="radio"/>	1	Personal	9184.00\$
<input type="radio"/>	2	Business	5514.42\$
<input type="radio"/>	18	Business	1000.00\$
<input type="radio"/>	19	Personal	10000.00\$

[See Account](#)

Transfer Money

From: To: Amount:

☒ ID ☐ Email ☐ Phone

Bills

View Bills

Bill ID	From	To	Amount	Recurring
3	1	4	257.00\$	Yes
6	2	8	67.00\$	Yes

Setup Bills

Sender Account ID: Recipient Account ID: Bill Amount:

Sender Account ID: Recipient Account ID: Bill Amount:

Sender Account ID: Recipient Account ID: Bill Amount:

☐ Bill(s) Recurring

[Submit New Bills](#)

Illustration 2: Client Hub

They simply need to select which account they want more information about and then click the 'See Account' button. Once pressed, they will go to their specific account page. On their account's page, they will see their past transactions and the bills that are linked to this particular account. The transactions and the bills are chronologically ordered in their respective tables.

Account Details				
Account ID: 2				
Account Type: chequing				
Balance: 5514.42				
Credit Limit: 20000.00				
Interests: 0.03				
Level: business				
Transactions History				
Transaction ID	Sender Account ID	Recipient Account ID	Amount	Date
62	2	12	-2.00	2018-12-01 07:40:55
61	2	8	-67.00	2018-12-01 07:40:55
1	1	2	100.00	2018-11-29 13:46:09
7	4	2	700.00	2018-11-29 13:46:09
2	3	2	300.00	2018-11-29 13:46:09
3	4	2	200.00	2018-11-29 13:46:09

Illustration 3: Specific Account Page

The quick sending money functionality of the bank allows user to send money to other clients, given either the client ID of the other party, or by giving their unique email or unique phone number.

Administrators

The administrators, authorized employees, can gain super user powers into the database by entering their employee id and password on the login page and selecting the Employee radio button. This redirects them to the admin hub. Similarly to the client hub, the admin hub contains all the links for them to perform their desired actions. In short, it is from there that they can get access to pages that modify every single table within the database. It also allows them to search the content of the tables.

Bank Administration

[Account Admin](#)[Branch Admin](#)[Charge Plan Admin](#)[Client Admin](#)[Employee Admin](#)[Service Admin](#)[Transaction Admin](#)[Bill Admin](#)[Client Log in Admin](#)[Employee Log in](#)[Schedule Admin](#)[Admin](#)

Force Monthly Script

[Exert Pressure](#)

Calculate Profits

☒ One Branch☐ All branch in a city☐ All branches

Enter Period Term:

Search terms:

Starting date:

End date:

[CalculateProfit](#)

Illustration 4: Admin Hub

The Admin Hub has a section that is used to force the process the bills and monthly fees. The script is the same one discussed in the Bills section. The lower part of the page contains a query to calculate the profits of either a branch, a city or of the whole bank. The input that it takes is the start and end time period on which the calculation occurs on.

Abstractions

In regards to the backend of the bank's website, classes were created to reflect the main tables in the database. A few examples of the classes are for bills, clients, branches and others. This form of programming decreased code reuse and streamlined the work between all the team members. Some people

could work on certain parts and only deal the methods without having to implement all the logic associated with their actions.

Security

Given that login page is used both by clients and administrators, a security measure has been put in place to limit the exposition of the clients to the higher privilege pages. To access admin pages, session information is verified and only up meeting certain criterias can a user view the admin pages. The user must have an employee ID that is registered in the EmployeeLogin table. This table records the id of employees and their password to gain access to the website. Similarly, only a client may view client and account information. The information is logged in the session information and verification is performed at every page so as not to leak any personal information. The choice of not putting the passwords with the Employee and Client table was deliberate as it is not good practice to merge such sensitive information into the same table. In addition, the passwords should not be put into a clear text form. However, given the focus of the project being on the implementation of a small database and not on the security aspects of it we did not store hashed forms of the passwords.

Contributions

Joel

- ER diagram
- Part of 3NF proof
- Created classes
 - Branch
 - Employee
 - Accounts
 - DomainLogic
 - Scripts
- Creates examples to insert into database

Feng

- Other part of 3NF proof
- Creates examples to insert into database
- Admin side
 - Pages to be able to modify all the tables within the database
 - Query, insertions, deletions and updates

Yassin

- Scripts
 - Database creation
 - Insertion value into database
- Classes
 - Bills
 - Client
- Login page and session control
- Client hub
- Integration of parts of project (code and report)

Alireza

- Page for specific accounts
- User guide
- Part of the creation of new transactions
- Part of the creation of new bills