COP 5725: Database Management Systems

Project Deliverable 2

Group 25

Shangde Gao - gao.shangde@ufl.edu
Srija Gurijala - srijagurijala@ufl.edu
Dimitrios Melissourgos - dmelissourgos@ufl.edu
Andrei Sura - asura@ufl.edu
Mukul Yadav - mchand.yadav@ufl.edu

Contents

Czech Bank Financial Data Analysis and Demographics

User Interface (UI) Design

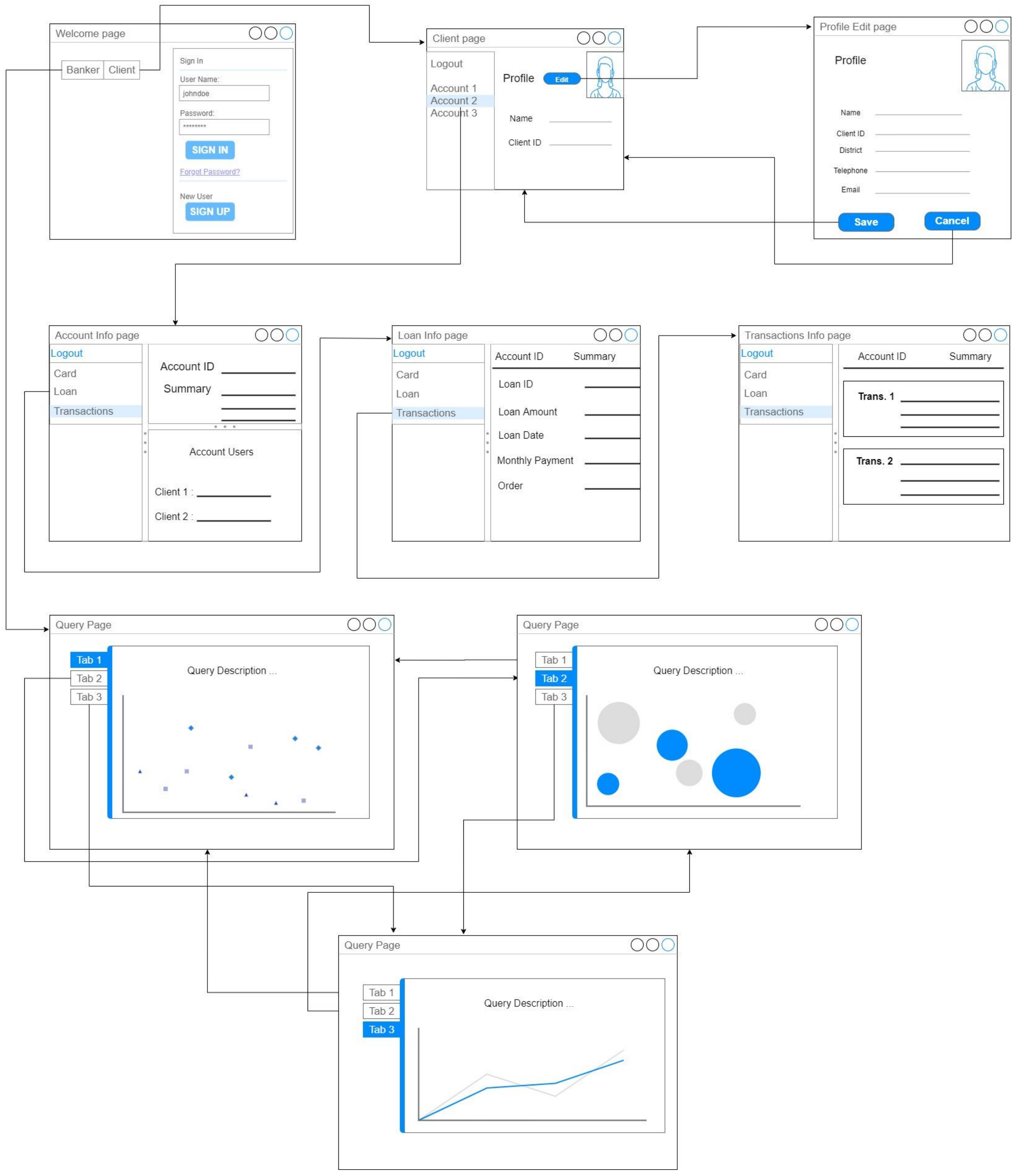
As explained in the first phase of our project, we intend to implement a website (implemented locally on Wamp Server), which offers a number of banking data services to its users. Webpage users can be either the bank's customers (*Client*) or the bank's employees (*Banker*). Each of these two user groups will have different levels of access to the data and different services offered to them by the website. The bankers will need to register with an account in order to be able to use the system. The clients are already inside the database of the system, since they are customers of the Czech bank. The structure of the website is presented in the flowchart of Figure 1 below.

Firstly, the website user will land on the welcome page, which will prompt them to select their user group, either Banker or Client and subsequently input their username and password. The sign in button will work differently for each group. In the case of a Banker after the sign in the user will be directed to the query page. This will be the webpage that implements most of the Complex Trend Database Query functionality discussed in the previous deliverable. The queries available to the Banker will be placed at the left as tabs and each tab will contain a complex trend query. Since the queries will usually have filters for various attributes, the filters will be placed on the top of the page and the resulting graph will be placed below them. The Bankers will have access to all the data of the database.

In the case of a Client, we will restrict the data that the user can interact with to only the fields that are directly related to that user. After sign in, the Client will be directed to the client page, which will show them their accounts with the Czech bank. It will also display a logout button (which will take them back to the welcome page), their personal information and a button that can take them to the profile edit page. In that page the client can change some of their information, but not the basic fields, such as name and client_id.

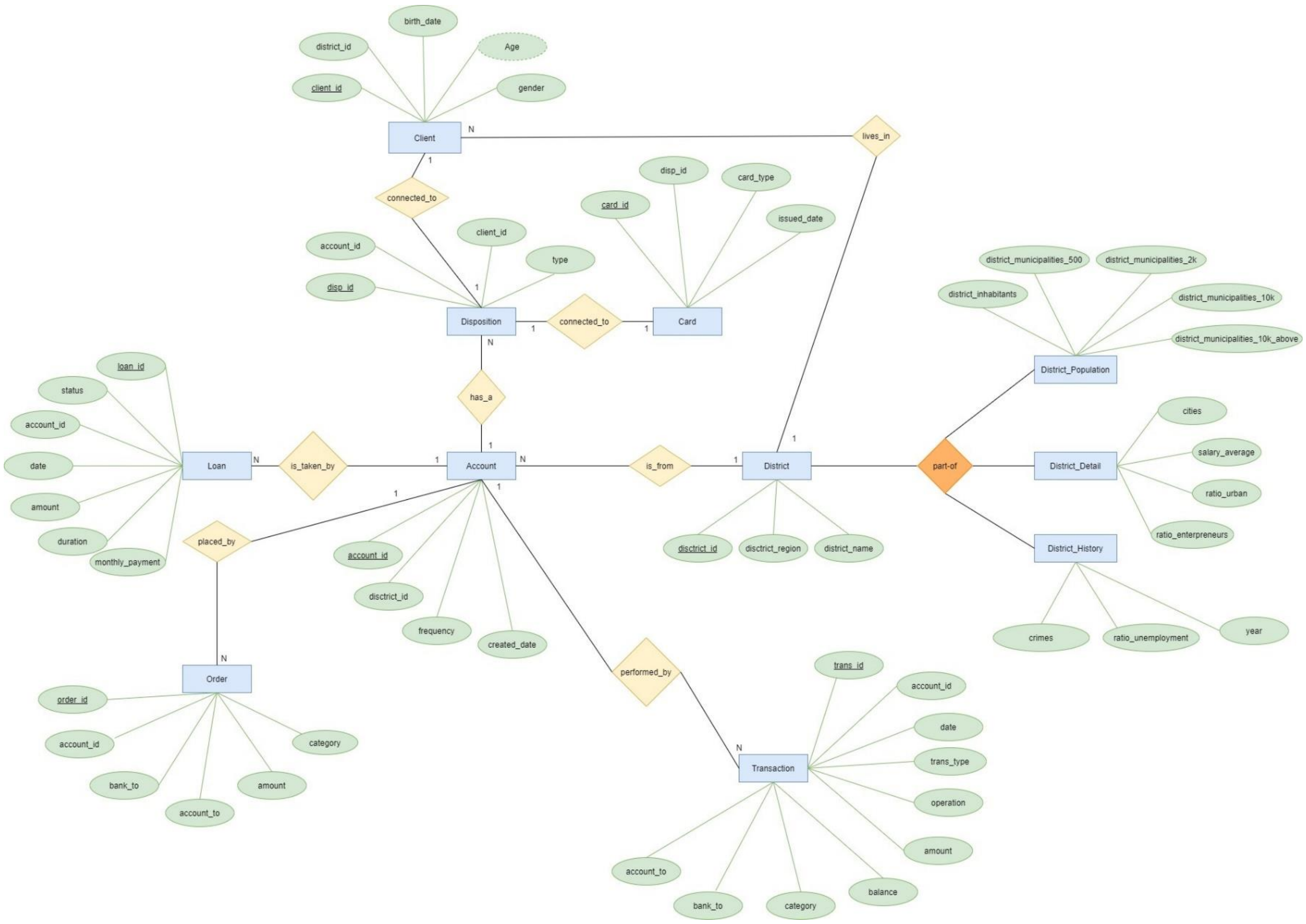
If the client clicks on one of their accounts, they will be directed to the account info page. In that page they will be able to see loans, cards, orders and transactions of that account in tabs on the left. Clicking on the card tab, the client will be directed to the credit card page, where they can see their credit card type and the date of issue. If the user clicks on the loan tab, they will be directed to the loan info page, where they can see the loan_id, the amount, the monthly payment, etc. In the transactions tab, the user can see their latest transactions of that account. Finally, the order tab will present the latest money orders (payments) along with their type. However, as mentioned above the Client will be restricted to these simple search queries, since they don't have access to other clients' data, for security reasons.

Figure 1: Website Flowchart



Conceptual Database Design

Figure 2: ER Diagram



Design Motivation

Entities: Client, Disposition, Card, Loan, Account, Order, District and Transaction.

Client: We provide the basic profile information – birth date, Client ID(the primary key), gender and district ID. This information is very crucial, because this basic information is required for complex queries such as the number of loans a particular client has taken and some trend-related information such as how many clients in the age gap of 20-30 are opting for loans. Here age is a derived attribute. The district_ID connects the clients with the demographic entity District, since our application is intended for a banker to understand the various aspects of his clients like the area they are living in, their history with the bank etc. It also connected to the card entity and the account entity through the disposition entity.

Disposition: The disposition entity has 4 attributes namely, type, account_id, client_id and the disp_id, which is the primary key of the attribute. This entity plays an important role in our application. As the name suggests, it connects the client entity, the account entity and the card entity. Also, the type attribute of this entity tells us whether the client is the owner of the account or the dispondent. It is the connecting link between the card entity, the client entity and the account entity.

Card: The functionality of this entity is to help analyse the number of cards each account has and the type of card that they own. This helps us to understand which type of card is being more preferred by the customers. This also helps us to track the spending of each client (owner/dispondent) on each card that they own. The attributes of this entity are date_issued, type, disp_id, card_id. It is connected to the account entity and the client entity through the disposition entity.

Loan: The loan entity gives the user information such as which account has taken a loan, the amount of the loan and the date of the loan granted. It also provides information about the loan being under review or rejected. It is connected to the account entity through the account_id key.

Account: The account attribute plays a key role in the ER model. The account_id connects the loan amount being taken by the client, it tells how many people belong to that account, with the help of the transactions and the card details, we would be able to find the spending trend of each client of the bank. It is the connection link between many entities such as the client, the card, the loan, the order, the transaction and the district entities, through their respective keys.

District: The district entity gives us demographic information, such as the number of people taking loans in that area, the spending of people from a particular area. With all this information we will be able to calculate the unemployment rate and how *well-off* the locality was. It could help the bankers setup new branches or ATMs in cases where they has a huge number of customers from that area. The district entity is further divided as district_population, district_detail and district_history. Each of these gives further information about the district. It is connected to the client and account entities through the district_id key.

Order: The order entity provides information about the amounts the origin and destination of money orders. It also shows the type of the order in the type attribute. It is connected to the account attribute through the account_id key.

Transaction: The transaction entity aims at providing useful information such as the times of the year when people spend most of their spending's. It also gives the user a comprehensive understanding of their spending amounts. This spending could be linked with the amount of loan taken by the account holder and help in deciding whether this account holder was capable enough to be granted a new loan, if needed. It is connected to the account entity through the account_id key.

With our website we wish to show a banker the trends that the present and past customers have been using with respect to the loans being taken at the same time as their spending. We wish to make this an easy and robust system to understand the demographics of different localities where they have clients and possibly identify their prospective clients. We also believe that this website would help the bank setup new branches in potential areas.

Important Relationships:

A client could be an owner of the account or the despondent of the account which in turn is linked to the account and card. This way we understand the number and type of cards each account could be linked with. Each loan is taken by an account which could have a single user or multiple users. The account also has a district to which it is related. This district has the information of each district and is partitioned to smaller entities, even though it exists as a single table in our database. Every client is from a district. An order could be placed by an account. Every account includes transactions.

Cardinalities:

- The client lives in 1 district, while a district can have many clients: 1-n
- The client_id corresponds to 1 disp_id and vice-versa: 1-1
- The card_id corresponds to 1 disp_id and vice-versa: 1-1
- A disp_id corresponds to 1 account, but an account can have many dispositions: 1-n
- An account can take many loans, but a loan can only be taken by a single account (loan_id): 1-n
- An account can place many orders, but an order can only be placed by a single account: 1-n
- An account can perform many transactions, but a transaction can be performed by a single account: 1-n
- An account belongs to 1 district, but a district can have many accounts: 1-n

Software Requirements

As mentioned in the previous deliverable, the software stack will consist of a web-based UI that provides navigation controls to browse between the pages which present new knowledge we derive from our relational data model (Figure 3).

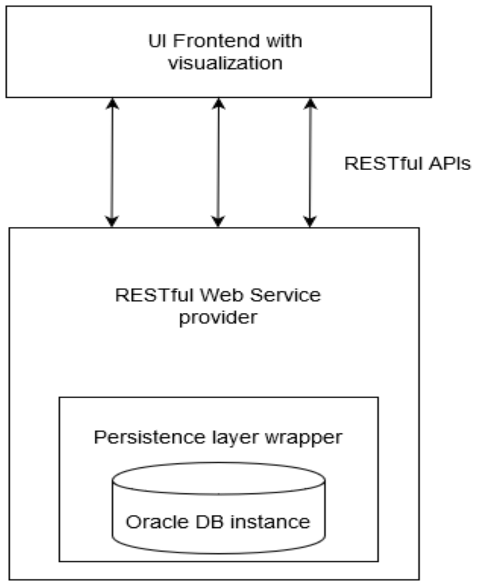
We are planning to use the following technologies for the project implementation:

- Drivers: provided by Oracle for accessing the proprietary database
- Python SqlAlchemy: Object Relational Mapper https://docs.sqlalchemy.org/en/13/dialects/oracle.html
- Javascript: Implementing the visualizations

The frontend will stream data from backend layer through REST APIs that translates UI query into a select used to fetch data from the Oracle DB instance.

Frontend: ReactJS framework with a visualization toolkit such as d3.js, chartjs, react-vis, or highcharts. The decision on which library to use will be made as soon as we start to experiment with the backend.
Backend: Flask/Spring/NodeJS based RESTful APIs for returning data to the frontend.
DB: CISE hosted Oracle DB instance as persistence layer.

Figure 3: Full Stack


UI Frontend with visualization
RESTful APIs
RESTful Web Service provider
Persistence layer wrapper
Oracle DB instance

Data Source

The dataset to be used for drawing aforementioned inferences is hosted at data.world website. All of our data can be found in the links below.

- http://lisp.vse.cz/pkdd99/berka.htm
- https://data.world/lpetrocelli/czech-financial-dataset-real-anonymized-transactions/workspace/intro
- https://www.researchgate.net/post/Is_there_any_public_database_for_financial_transactions_or_at_least_a_synthetic_generated_data_set