Advanced Databases

Project Phase 1

Report
Academic Year: 2024/2025

Authors: *Diogo Barreta 64560, Muhammed Jaabir Mohamed Zifri 64912, Josè Dalla Torre 64672*

Date: December 1, 2024

Dataset Description

The dataset studied was the **"Financial Transactions Dataset: Analytics"**, obtained from Kaggle. This data set contains data related to transaction records, customer information, and card data from a bank during the 2010s. The dataset comprises three CSV files:

- `cards_data.csv`: Includes information about bank accounts. The file has 6146 rows and 13 columns:
  - Columns: `id`, `client_id`, `card_brand`, `card_type`, `card_number`, `expires`, `cvv`, `has_chip`, `num_cards_issued`, `credit_limit`, `acct_open_date`, `year_pin_last_changed`, and `card_on_dark_web`.

- `users_data.csv`: Contains information about clients. This file has 2000 rows and 14 columns:
  - Columns: `id`, `current_age`, `retirement_age`, `birth_year`, `birth_month`, `gender`, `address`, `latitude`, `longitude`, `per_capita_income`, `yearly_income`, `total_debt`, `credit_score`, and `num_cards_issued`.

- `transactions_data.csv`: Contains all transactions processed by the bank, including transaction features. This file has 13.3m rows and 12 columns:
  - Columns: `id`, `date`, `client_id`, `card_id`, `amount`, `use_chip`, `merchant_id`, `merchant_state`, and `zip`.

Due to the large size of `transactions_data.csv`, we worked with a subset of 50000 rows for analysis. By associating the `id` value in `users_data.csv` with `client_id` in the other files and textttid value in `cards_data.csv` with `client_id`, we established a relationship between the three datasets.

Data Cleaning and Preprocessing

2.1 Handling Missing Values

- First, we used a countplot to analyze the presence of missing values.
- The missing values in the numerical fields were replaced with 0.
- Categorical fields were imputed with the mode or 'Unknown'.

2.2 Data Type Conversions

Before the database population, we decided the following criteria:

- All columns that had amounts with the `$` sign were converted to int by removing the sign.
- Columns that had 'YES' & 'NO' were converted to the corresponding boolean value.
- All the columns values must have the same panda datatype.

Relational Schema

We decided to maintain the same structure since, after the data cleaning, the data set has all the property suitable to be a relational schema. We present our diagrams next.

- The user primary key is **id**. It has a (1,N) relationship with the others identity.
- The cards primary key is **id** with the user_id as a foreign key. It has a (1,1) relationship with user and (1,N) with transactions.
- The transaction primary key is **id** with the user_id and card_id as a foreign keys. It has a (1,1) relationship with the user and (1,1) with cards.

id	*client_id*	card_brand	card_type	card_number	expires	cvv	has_chip	nums_cards_issued	credit_limit	acct_open_date	year_pin_last_changed	card_on_dark_web
4524	825	Visa	Debit	4344676511950444	12/2022	623	YES	2	24295	09/2002	2008	No
2731	825	Visa	Debit	4956965974959986	12/2020	393	YES	2	21968	04/2014	2014	No
3701	825	Visa	Debit	4582313478255491	02/2024	719	YES	2	46414	07/2003	2004	No

id	current_age	retirement_age	birth_year	birth_month	gender	address	latitude	longitude	per_capita_income	yearly_income	total_debt	credit_score	num_credit_cards
825	53	66	1966	11	Female	462 Rose Lane	34.15	-117.76	29278	59696	127613	787	5
1746	53	68	1966	12	Female	3606 Federal Boulevard	40.76	-73.74	37891	77254	191349	701	5
1718	81	67	1938	11	Female	766 Third Drive	34.02	-117.89	22681	33483	33483	698	5

id	*client_id*	*card_id*	date	amount	use_chip	merchant_id	merchant_city	merchant_state	zip	mcc	errors
7475327	1556	2972	2010-01-01 00:01:00	-77.00	Swipe Transaction	59935	Beulah	ND	58523.0	5499	
7475328	561	4575	2010-01-01 00:02:00	14.57	Swipe Transaction	67570	Bettendorf	IA	52722.0	5311	
7475329	1129	102	2010-01-01 00:02:00	80.00	Swipe Transaction	27092	Vista	CA	92084.0	4829	

Figure 1: Relational Schema Diagram

ER Diagram

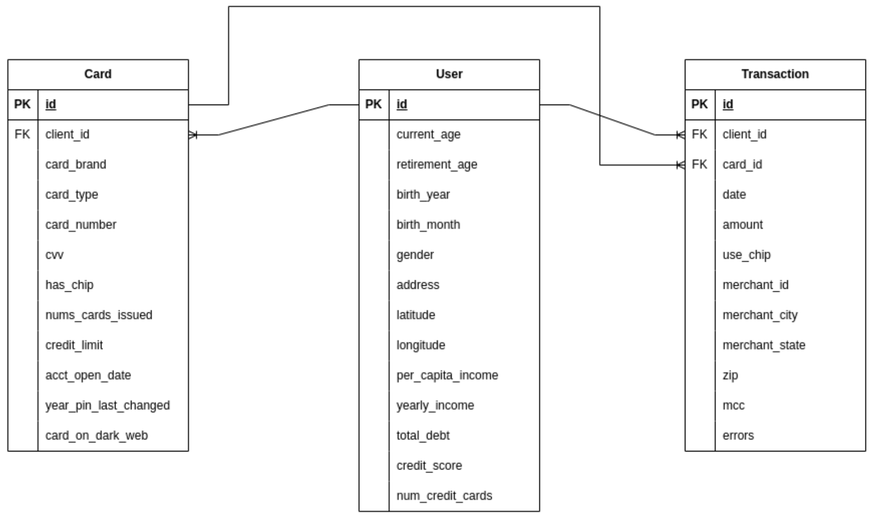

Figure 2: Entity-Relationship Diagram

Database Setup and Upload Challenges

5.1 MySQL Implementation

- The schema for MySQL was created based on an Entity-Relationship model.
- Data were uploaded using Python's `mysql.connector` as seen in class.

5.2 MongoDB Implementation

- The data were uploaded to collections using Python's `pymongo`.
- Data were denormalized to optimize query performance.

Queries and Results

6.1 Simple Queries

1. Retrieve users with a credit score greater than 750.
2. Find transactions with non-null error messages.

6.2 Complex Queries

1. Calculate the total transaction amount by state for female users.
2. Identify cards flagged as compromised (`card_on_dark_web`) used in transactions above $1,000.

6.3 Performance Evaluation

As expected MongoDB has a better performance on almost everything except for the third query.
The results are provided in the notebook for further reading.

Discussion

7.1 Observations

- MongoDB outperformed MySQL for queries that require a large number of aggregation.
- No results were found in the last query. This is not due to an error of the query, but instead because there were no cards that fulfilled the conditions.

7.2 Limitations

- One of the complex queries was slower for MongoDB.

Conclusion and Next Steps

Phase 1 highlights the strengths and weaknesses of both database systems. Future steps include implementing indexing and query optimizations in Phase 2 to further enhance performance.

References

[1] Victor, C. *Transactions Fraud Dataset*. Kaggle. Available at: https://www.kaggle.com/datasets/computingvictor/transactions-fraud-datasets.