

**CSE 414 DATABASE**

**SPRING 2024**

**HOMEWORK 1**

**MERT GÜRŞİMŞİR**

**1901042646**

**DESIGN**

For the building a NoSQL database system for banking operations, I have used MongoDB database technology.

For the banking operations I have 4 collections:

* customers
* accounts
* transactions
* branches

I am keeping accounts’ id values that customers has as an array in the customers collection. Also I am keeping associated customer’s id value in the accounts collection to easily access customer, to know who owns the account. Each customer can have several accounts but each account has only 1 user. I am also keeping branches’ id values in the accounts collection to know which branch each account belongs to. Different accounts can have same branch but an account can have only 1 branch. I am keeping accounts’ id values in the transactions to know that transaction belongs to which account. Accounts can have more than 1 transactions but transactions can have only 1 account.

Advantage of keeping the id values is it reduces the data duplication. We can access the to other collections by using the id values.

In order to visualize the paragraph, I am representing my schema design below:

A blue squares with white text

Description automatically generated

These schemes provide flexibility and scalability. For example, if a new requirement arises in the future, we can easily add new fields or change the existing structure.

Creating the database:



Creating collections:

A screen shot of a computer program

Description automatically generated

**BRANCHES**

Inserting Branches:

A computer screen shot of a computer code

Description automatically generated

I have used insertMany to insert 2 branches at once. I have given fields and values for 2 branches.

**CUSTOMERS**

Inserting a customer:

A computer screen shot of a code

Description automatically generated

Here, I am inserting 1 customer with specified fields. I have not given any id value for the customer because it is automatically assigned.

name is an object that keeps first and last values that specify the first name and last name.

email, phone, address fields are all string.

accounts field has an array to keep accounts id values that user has. Initially the array is empty because user doesn’t have any account at the beginning.

Inserting many customers:

A screenshot of a computer program

Description automatically generatedA computer screen shot of a code

Description automatically generated

**ACCOUNTS**

Inserting a new account for user Mert (for WestSide branch):

A screen shot of a computer code

Description automatically generated

Now since a new account is added to customer with ID “662d89e7b2bbeb64b81b0e9e”, we  
 need to add this account to that customer’s accounts field.

A screen shot of a computer

Description automatically generated

Now customer looks like this:

A screenshot of a computer code

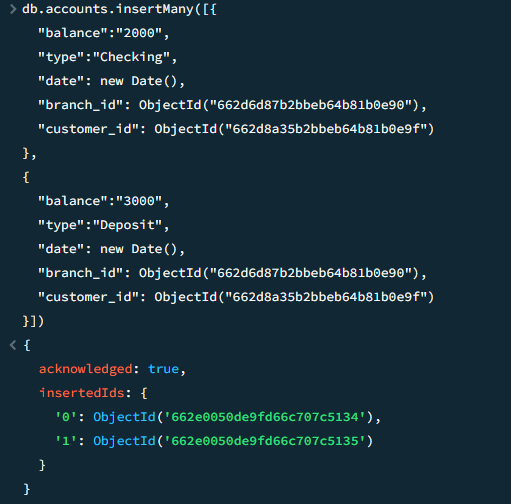
Description automatically generated

I am keeping accounts as array to be flexible. So when a new account is going to be added   
 to the customer, its id is going to be simply pushed to that array.

Insert 2 accounts for user Haruki Murakami (for WestSide branch):

A computer screen shot of a program code

Description automatically generated



Updating Haruki’s accounts field:

A computer screen with white text

Description automatically generated

Haruki now looks like this:

A screen shot of a computer code

Description automatically generated

Insert an account for user Carl Sagan (for WestSide branch):

A computer screen shot of a code

Description automatically generated

A computer screen shot of a code

Description automatically generated

Updating Carl’s accounts field:

A screen shot of a computer

Description automatically generated

Carl now looks like this:

A screen shot of a computer code

Description automatically generated

Inserting 3 accounts for user Richard Feynman (for EastSide branch):

A computer screen shot of a code

Description automatically generated

A screen shot of a computer code

Description automatically generated

Updating Richard’s accounts field:

A computer code on a dark background

Description automatically generated

Richard now looks like this:

A computer screen shot of numbers and letters

Description automatically generated

Inserting 2 accounts for user Nilgün Marmara (for EastSide branch):

A computer screen shot of a program code

Description automatically generated

A computer screen shot of a code

Description automatically generated

Updating Nilgün’s accounts field:

A computer screen shot of a code

Description automatically generated

A screenshot of a computer code

Description automatically generated Nilgün now looks like this:

**TRANSACTIONS**

Now user Mert is making transactions:

A computer screen shot of a computer code

Description automatically generated

User Haruki making transactions to his accounts:

A screenshot of a computer program

Description automatically generated

User Carl making transactions to his account:

A screenshot of a computer program

Description automatically generatedA computer screen shot of white text

Description automatically generated

User Richard making transactions to his accounts:

A computer screen shot of white text

Description automatically generatedA computer screen shot of a code

Description automatically generatedA computer screen shot of a computer code

Description automatically generated

User Nilgün making transactions to her accounts:

A computer screen with white text

Description automatically generated

A computer screen with white text

Description automatically generated

I am summing up all the accounts and transactions below:

* **MERT GÜRŞİMŞİR (1 account, 3 transactions)**
  + Account
    - Transactions: +1000, -1000, +1000
* **HARUKI MURAKAMI (2 accounts, 5 transactions)**
  + Account 1
    - Transactions: +2000
  + Account 2
    - Transactions: +3000, -1000, -500, +1500
* **CARL SAGAN (1 account, 7 transactions)**
  + Account
    - Transactions: +2000, -1500, +500, +2000, +1000, -500, +500
* **RICHARD FEYNMAN (3 accounts, 6 transactions)**
  + Account 1
    - Transactions: +1000, +4000
  + Account 2
    - Transactions: +6000
  + Account 3
    - Transactions: +4000, -1000, +4000
* **NİLGÜN MARMARA (2 accounts, 3 transactions)**
  + Account 1
    - Transactions: +4000, +4000
  + Account 2
    - Transactions: +9000

**QUERIES**

1 🡪 Retrieve all transactions associated with a particular account

Haruki Murakami’s transactions associated with his 2nd account:

A screenshot of a computer program

Description automatically generated

2 🡪 Find the total balance of a customer across all accounts

Finding total balance of Richard Feynman across all of his accounts. His first account has 5000, second account has 6000, third account has 7000 so in total he has 18000:

A computer screen shot of a program code

Description automatically generated

Finding total balance of Haruki Murakami across all of his accounts. His first account has 2000, second account has 3000 so in total he has 5000:

A computer screen shot of a program code

Description automatically generated

With aggregate, we process data records and return computed results. match and group are stages in aggregation pipeline:

1. $match: Filters documents based on specified criteria similar to find() method. Here we are selecting only those accounts associated with the specified customer based on the customer\_id.
2. $group: Groups the documents based on specified criteria and applies expressions to the grouped data. Here we are grouping the accounts by customer\_id and calculating the total balance.

3 🡪 Identify customers with the highest number of transactions

In the current situation, Carl Sagan has the most number of transactions with 7 transactions.

Query:

A screen shot of a computer program

Description automatically generated

Result:

A computer screen shot of a code

Description automatically generated

Here we have 7 stages:

1. $lookup: Here we are joining the transactions collection with the accounts collection based on the account\_id.
2. $unwind: Here we are deconstructing the resulting array from the $lookup.
3. $lookup: This another lookup is for joining the accounts collection with the customers collection based on the customer’s id.
4. $unwind: This unwing is deconstructing the resulting array from the second $lookup.
5. $group: In this stage, we are grouping transactions by customer and calculating the total number of transactions for each customer.
6. $sort: Here we are sorting the customers based on their total number of transactions in descending order specified by -1.
7. $limit: This is for retrieving only the first result which will be the customer with the maximum number of transactions.

Richard Feynman has 6 transactions. I am adding 2 test transaction for his second account:

A screen shot of a computer code

Description automatically generated

Now making the same query I am getting the result:

A screen shot of a computer program

Description automatically generated

4 🡪 Get a list of branches along with the total number of accounts associated with each branch

A computer screen shot of a computer code

Description automatically generated

Mert’s 1 account, Haruki’s 2 accounts, Carl’s 1 account are associated with WestSide branch;

Richard’s 3 accounts, Nilgün’s 2 accounts are associated with EastSide branch.

Stages:

1. $lookup: To join accounts collection with the branches collection based on branch\_id.
2. $unwind: Deconstructing resulting array from $lookup.
3. $group: Grouping accounts by branch and calculating total number of accounts for each.