Contract Sleeving Service

OZAN TAÇALAN

Summary

I've implemented required service with Spring Boot. I also added more functionality to contract entity such as:

- transaction date
- trading material
- material amount
- exchange rate (random value in fixed width)
- total cost of trade (calculated via amount, exhange rate and material price)

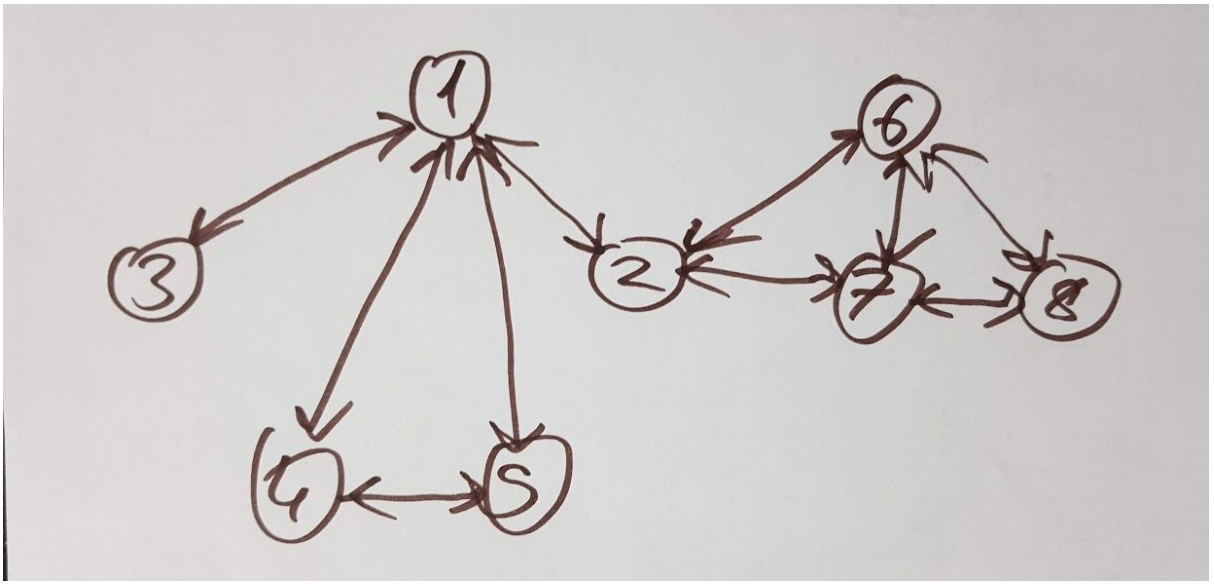
Entity

- Contract (fundamentals: companyA, anothercompany, cost(trade material, amount))
- Material (fundamentals: material name and price information)

Data Structure

This part was a little tricky for me. Because when I think about how to represent some database rows in memory, which they might have connections with each other, first and easiest thing that comes to my mind is using Graph data structure. I described some possible candidates to graph but I've used graph anyway. Because I think there is no way to be sure there is no connection between companies without checking each of them. It might some possible solutions to this and I'll talk about them below.

I inserted some materials and contracts into in-memory database. You can see contract relations in visualization below. Numbers describes companies, for example 4 means company_4.



When user specifies start and end nodes to find possible sleeves;

1. Collect all rows from database
2. Generate graph representation of these rows in memory
3. Use altered version of breadth first search algoritm to discover adjacencies.
4. Collect possible outcomes.

Other possible solutions:

- Some controlled environment where we are sure where to insert our contract (dividing our dataset), for example there might be “gold”, “silver” contracts/companies. Then we can calculate sleeves in smaller records.
- Some disk representation of past sleeve calculations. For example, if server calculates possible sleeves between company_1-company_7, generated graph can be stored in disk. When company_1/company_7 needs this calculation, it can read from disk. Of course disk representation must be triggered on some actions to be updated somehow.

I’m not sure about these alternate solutions but it was fun to think outside of box.

Rest Service

I’ve exposed three endpoint for this service where I also added some optional parameters to some of them. Request/responses are taken after importing some data to database.

- POST /contracts: You can insert new contract by stating two companies, trading material name and amount.

  Example request:

```
{
        "aCompany": "company_1",
        "anotherCompany": "company_9",
        "material": {
        "name": "gas"
        },
        "amount": 2000
}
```

  Example response:

```
{
        "data": [
        {
        "transactionDate": "2019-05-26T22:17:32.702+0000",
        "aCompany": "company_1",
        "anotherCompany": "company_9",
        "material": {
                "name": "gas",
                "price": 7.533161012695676
        },
        "amount": 2000,
        "rate": 1.1806944651299758,
        "cost": 17839.225406124846,
        "valid": true,
        "status": "active"
        }
        ],
        "response": "Success"
}
```

- GET /sleeves: You can see possible sleeve options between two companies by dynamically generated graph.
    - Optional parameter option: You can sort records by transaction date (asc/desc) or you may only select the best result. Default: desc

Example request:
localhost:8080/sleeves?aCompany=company_6&anotherCompany=company_8
Example response:


```
{
        "data": [
        {
        "sleevePath": [
                "company_6",
                "company_8"
        ],
        "totalCost": 159666.38800486413
        },
        {
        "sleevePath": [
                "company_6",
                "company_7",
                "company_8"
        ],
        "totalCost": 239889.43853677934
        },
        {
        "sleevePath": [
                "company_6",
                "company_2",
                "company_7",
                "company_8"
        ],
        "totalCost": 268673.00552404224
        }
        ],
        "response": "Success"
}
```

- GET /history: You may see a company's latest contracts.
    - Optional parameter limit: You may limit result list by stating a value between 1-20.

Example request:
localhost:8080/history?aCompany=company_4
Example response:


```
{
        "data": [
        {
        "transactionDate": "2019-05-20T15:47:52.690+0000",
        "aCompany": "company_5",
        "anotherCompany": "company_4",
        "material": {
                "name": "gas",
                "price": 8.77767526541273
        },
        "amount": 10000,
        "rate": 1.1156676776069685,
        "cost": 84727.44562653279,
        "valid": true,
        "status": "active"
        },
        {
        "transactionDate": "2019-01-01T15:47:52.690+0000",
        "aCompany": "company_4",
        "anotherCompany": "company_1",
        "material": {
                "name": "sun",
                "price": 9.995060565303193
        },
        "amount": 12000,
        "rate": 1.1065536009634844,
        "cost": 139096.80521805456,
        "valid": true,
        "status": "active"
        }
        ],
        "response": "Success"
}
```


Response Wrappers

I've wrote custom wrappers that wraps my List<Contract> or List<List<Contract>> list. They seem a little messy to me so I trimmed a little bit. SleeveWrapper consist of two parts: data and response. Data is anything related to user. Response states if everything went okay (Success) or some error message.