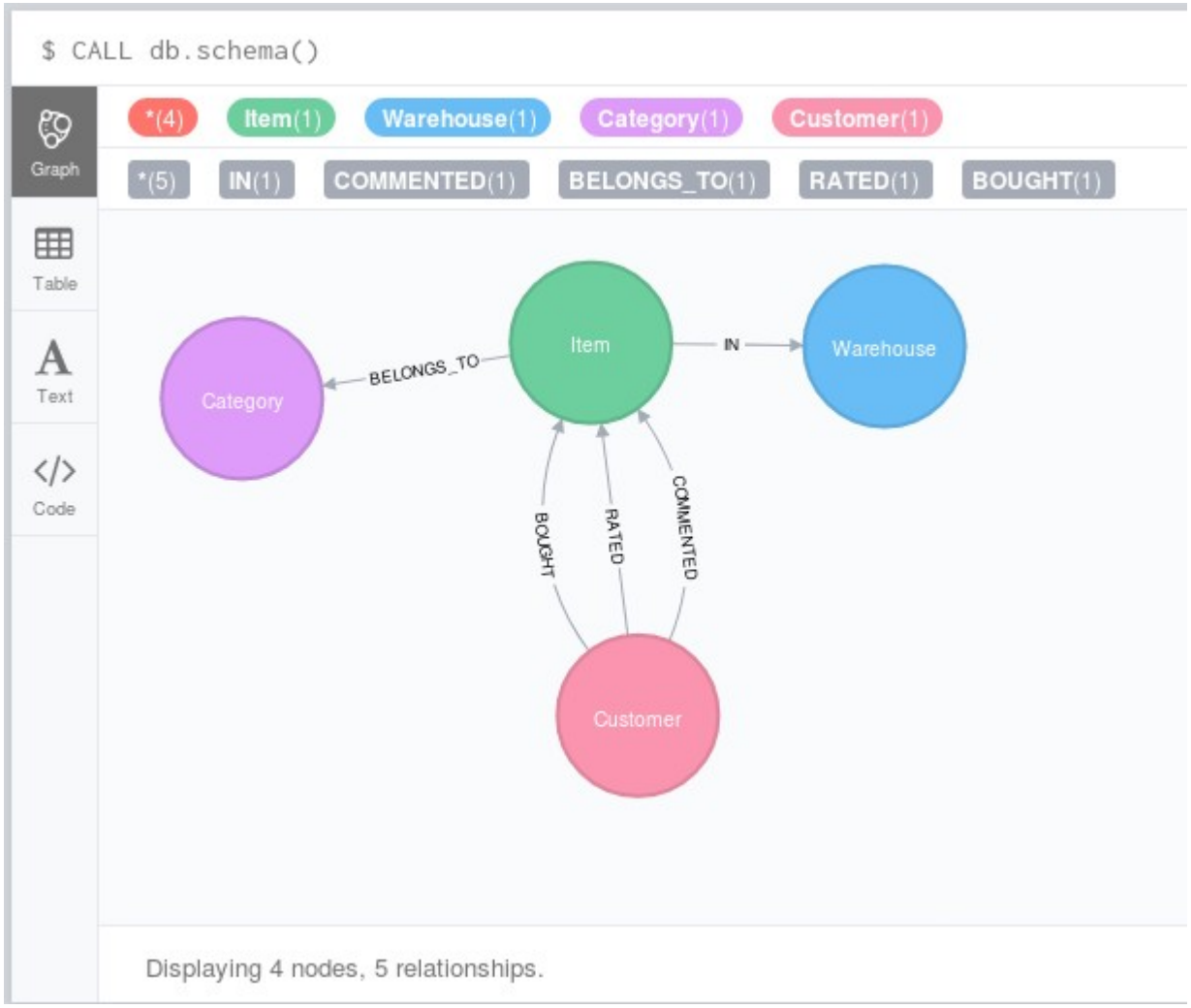
Sprawozdanie Neo4j

Model

Na potrzeby ćwiczenia stworzyłem prosty model bazy danych:



Kod aplikacji stworzyłem przy użyciu frameworka Spring Boot wraz z Spring Data Neo4j, OGM.

Funkcje do tworzenia obiektów:

```
public Item addItem(String productName, double price){
    Item item = new Item(productName, price);
    itemRepository.save(item);
    return item;
}
public Category addCategory(String name){
    Category category = new Category(name);
    categoryRepository.save(category);
    return category;
}
```

```java
public Warehouse addWarehouse(String symbol, String address, Long phoneNumber){
    Warehouse warehouse = new Warehouse(symbol, address, phoneNumber);
    warehouseRepository.save(warehouse);
    return warehouse;
}


public Customer addCustomer(String firstName, String lastName, int age, String
address){
    Customer customer = new Customer(firstName, lastName, age, address);
    customerRepository.save(customer);
    return customer;
}
```

Funkcje do tworzenia relacji:

```java
public Item addItemToWarehouse(Item item, Warehouse warehouse, int inStock){
    if (item == null) return null;
    item.addWarehouse(warehouse, inStock);
    itemRepository.save(item);
    return item;
}
public Item addItemToCategory(Item item, Category category){
    if (item == null || category == null) return item;
    item.addCategory(category);
    itemRepository.save(item);
    return item;
}
public Customer buyItem(Customer customer, Item item){
    if (customer == null || item == null) return customer;
    customer.buyItem(item);
    customerRepository.save(customer);
    return customer;
}
public Customer commentItem(Customer customer, Item item, String message){
    if (customer == null || item == null) return customer;
    customer.commentItem(item, message);
    customerRepository.save(customer);
    return customer;
}
public Customer rateItem(Customer customer, Item item, int rating){
    if (customer == null || item == null) return customer;
    customer.rateItem(item, rating);
    customerRepository.save(customer);
    return customer;
}
```

Populator danych

```
@Service
public class PopulateDatabase implements Command {
    @Autowired
    private DatabaseAccess db;
    DataFactory df = new DataFactory();
    @Override
    public void execute() {
        ArrayList<Category> categories = new ArrayList<>();
        ArrayList<Customer> customers = new ArrayList<>();
        ArrayList<Item> items = new ArrayList<>();
        ArrayList<Warehouse> warehouses = new ArrayList<>();
        for (int i = 0; i<30; i++){
            warehouses.add(db.addWarehouse(df.getRandomChars(2),
df.getAddress(),
                    Long.parseLong(df.getNumberText(9))));
            categories.add(db.addCategory(df.getRandomWord()));
            items.add(db.addItem(df.getRandomWord(),
                    (double) df.getNumberBetween(100, 10000) / 100));
            customers.add(db.addCustomer(df.getFirstName(), df.getLastName(),
                    df.getNumberBetween(18, 100), df.getAddress()));
        }
        for (int i = 0; i<50; i++){
            db.buyItem(df.getItem(customers), df.getItem(items));
            db.addItemToWarehouse(df.getItem(items),
                    df.getItem(warehouses), df.getNumberBetween(30, 1000));
            db.addItemToCategory(df.getItem(items), df.getItem(categories));
        }
        for (int i = 0; i<10; i++){
            Customer c = df.getItem(customers);
            List<Item> itemsBought = c.getItemsBought()
                    .stream()
                    .map(Bought::getItem)
                    .collect(Collectors.toList());
            if (!itemsBought.isEmpty()) {
                db.commentItem(c, df.getItem(itemsBought), df.getRandomText(50,
150));
                db.rateItem(c, df.getItem(itemsBought), df.getNumberBetween(1,
10));
            }
        }
    }
}
```

Funkcja do pobrania wszystkich relacji dla węzła

```
public Iterable<Object> getRelationships(Object e1){
    if (e1 == null) return new ArrayList<>();
    Result result = entityRepository.getRelations(e1);
    LinkedList<Object> rels = new LinkedList<>();
    result.forEach(map -> rels.add(map.getOrDefault("r", null)));
    return rels;
}

@Query("MATCH (e)-[r]-(b) WHERE " +
        "id(e)={e} RETURN r, e, b")
Result getRelations(@Param("e") Object e);

@SpringBootApplication
@EnableNeo4jRepositories
public class Main {
    private final static Logger log = (Logger)
LoggerFactory.getLogger(Main.class);
    public static void main(String[] args){
        SpringApplication.run(Main.class, args);
    }
    @Bean
    CommandLineRunner demo(DatabaseAccess db,
                           AddCustomer addCustomer, AddCategory addCategory,
                           AddItem addItem, AddWarehouse addWarehouse,
                           AddItemToCategory addItemToCategory,
                           AddItemToWarehouse addItemToWarehouse,
                           BuyItem buyItem, RateItem rateItem,
                           CommentItem commentItem,
                           PopulateDatabase populateDatabase,
                           GetShortestPath getShortestPath,
                           GetAllRelationships getAllRelationships,
                           EntityRepository entityRepository,
                           CustomerRepository customerRepository,
                           ItemRepository itemRepository
                           ){
        return args -> {
            db.drop();
            populateDatabase.execute();
//            getShortestPath.execute();
//            getAllRelationships.execute();
            ArrayList<Customer> customers = new ArrayList<>();
            customerRepository.findAll().forEach(customer -> {
                if (!customer.getItemsBought().isEmpty())
customers.add(customer);
            });
            //
            // Shortest path
            //
            System.out.println(db.getShortestPath(customers.get(3),
customers.get(23)).describe());
            //
            // Relationships
            //
            db.getRelationships(customers.get(23)).forEach(System.out::println);
            System.out.println("Finishing");
        };
    }
}
```

Powyższe wywołanie skutkuje wypisaniem na ekran, na przykład takich danych:
ID: 129 BrentHampton Age: 19 Address: 1133 Clarice Parkway ----[RATED]----> ID: 60 ProductName: caves
ID: 129 BrentHampton Age: 19 Address: 1133 Clarice Parkway ----[COMMENTED]----> ID: 60 ProductName: caves
ID: 129 BrentHampton Age: 19 Address: 1133 Clarice Parkway ----[BOUGHT]----> ID: 60 ProductName: caves

Funkcja do znalezienia ścieżki dla dwóch węzłów:

```
public Path getShortestPath(Object e1, Object e2){
    return e1 == null || e2 == null ? new Path() : new
Path(entityRepository.getShortestPath(e1, e2));
}

@Query("MATCH path=shortestPATH((n1)-[*]-(n2)) WHERE " +
        "id(n1)={n1} AND id(n2)={n2} " +
        "RETURN nodes(path) as npath, relationships(path) as relationships")
Result getShortestPath(@Param("n1") Object n1, @Param("n2") Object n2);
```

Uwaga! Ścieżka jest nieskierowana, aby można było znaleźć łatwo jakąkolwiek ścieżkę w moim modelu

Wywołanie tego samego maina, co w poprzednim punkcie, dla najkrótszej ścieżki wypisze na przykład:
ID: 41 KaitlynKerr Age: 53 Address: 1751 Fairall Boulevard ----[RATED]----> ID: 56 ProductName: computer
ID: 97 DeannaMcClain Age: 38 Address: 1174 Arch Boulevard ----[BOUGHT]----> ID: 56 ProductName: computer
ID: 97 DeannaMcClain Age: 38 Address: 1174 Arch Boulevard ----[BOUGHT]----> ID: 4 ProductName: asked
ID: 109 AliciaGrant Age: 54 Address: 696 Galena Lane ----[BOUGHT]----> ID: 4 ProductName: asked
ID: 109 AliciaGrant Age: 54 Address: 696 Galena Lane ----[RATED]----> ID: 60 ProductName: caves
ID: 129 BrentHampton Age: 19 Address: 1133 Clarice Parkway ----[RATED]----> ID: 60 ProductName: caves

Repozytorium:

Mój program dostępny jest w repozytorium pod adresem:

https://github.com/Elrohil44/neo4j

Aby uruchomić, należy wykonać polecenie:

> gradle bootRun