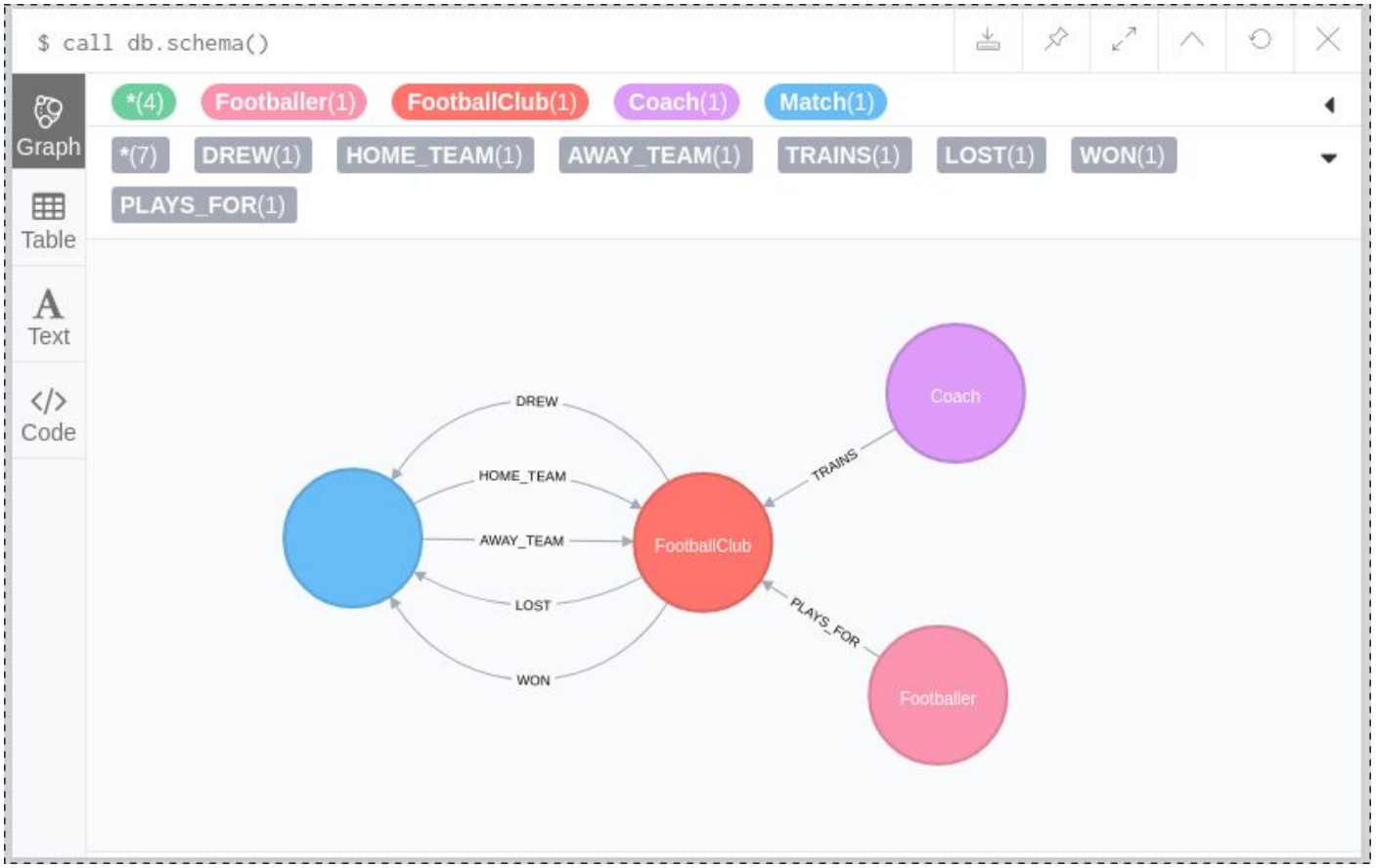
Dawid Siwko

Sprawozdanie Neo4j

1. (1pkt) Należy wymyślić, lub znaleźć prosty graf, który ma przynajmniej 2 węzły i 3 różne krawędzie (+ kilka atrybutów). Należy opisać lub pokazać model + atrybuty. Można użyć do tego np: CALL db.schema()

Graf będzie opisywał relacje w świecie piłki nożnej.
Węzły: FootballClub, Match, Coach, Footballer
Relacje:

- Footballer **PLAYS_FOR** FootballClub
- Coach **TRAINS** FootballClub
- FootballClub **WON / DRAW / LOST** Match
- Match **HOME_TEAM** FootballClub
- Match **AWAY_TEAM** FootballClub



2. (1pkt) Napisać funkcje do tworzenia poszczególnych obiektów i relacji w grafie. Proszę o wklejenie kodu/zdjęcia kodu

```java
@NodeEntity
public class FootballClub {
    @Id
    @GeneratedValue
    private Long FootballClubId;
    private String name;

    @Relationship(type = "TRAINS", direction = Relationship.INCOMING)
    private CoachContract coachContract;

    @Relationship(type = "PLAYS_FOR", direction = Relationship.INCOMING)
    private Set<FootballerContract> footballerContracts = new HashSet<>();

    @Relationship(type = "WON")
    private Set<Match> winnings = new HashSet<>();

    @Relationship(type = "DREW")
    private Set<Match> draws = new HashSet<>();

    @Relationship(type = "LOST")
    private Set<Match> losses = new HashSet<>();

    public FootballClub() {}

    public FootballClub(String name) {
        this.name = name;
    }

    public Long getFootballClubId() {
        return FootballClubId;
    }

    public String getName() {
        return name;
    }

    public CoachContract getCoachContract() {
        return coachContract;
    }

    public Set<FootballerContract> getFootballerContracts() {
        return footballerContracts;
```

```
    }

    public void setCoach(CoachContract coachContract) {
        this.coachContract = coachContract;
    }

    public void addFoorballer(FootballerContract footballerContract) {
        footballerContracts.add(footballerContract);
    }

    public Set<Match> getWinnings() {
        return winnings;
    }

    public Set<Match> getDraws() {
        return draws;
    }

    public Set<Match> getLosses() {
        return losses;
    }

    public void addWin(Match match) {
        winnings.add(match);
    }

    public void addDraw(Match match) {
        draws.add(match);
    }

    public void addLoss(Match match) {
        losses.add(match);
    }
}
```

```java
@NodeEntity
public class Match {
    @Id
    @GeneratedValue
    private Long matchId;

    @Relationship(type = "HOME_TEAM", direction = Relationship.UNDIRECTED)
    private FootballClub homeTeam;

    @Relationship(type = "AWAY_TEAM", direction = Relationship.UNDIRECTED)
    private FootballClub awayTeam;

    private LocalDate date;
    private Integer goalsHomeTeam;
    private Integer goalsAwayTeam;

    public Match() {}

    public Match(FootballClub homeTeam, FootballClub club2, LocalDate date) {
        this.homeTeam = homeTeam;
        this.awayTeam = club2;
        this.date = date;
    }

    public Match(FootballClub homeTeam, FootballClub awayTeam, LocalDate
date, Integer goalsHomeTeam, Integer goalsAwayTeam) {
        this.homeTeam = homeTeam;
        this.awayTeam = awayTeam;
        this.date = date;
        this.goalsHomeTeam = goalsHomeTeam;
        this.goalsAwayTeam = goalsAwayTeam;
        updateFootballClubs();
    }

    public Long getMatchId() {
        return matchId;
    }

    public FootballClub getHomeTeam() {
        return homeTeam;
    }

    public FootballClub getAwayTeam() {
        return awayTeam;
    }
```

```
    public LocalDate getDate() {
        return date;
    }

    public Integer getGoalsHomeTeam() {
        return goalsHomeTeam;
    }

    public Integer getGoalsAwayTeam() {
        return goalsAwayTeam;
    }

    public void setResult(Integer goalsHomeTeam, Integer goalsAwayTeam) {
        this.goalsHomeTeam = goalsHomeTeam;
        this.goalsAwayTeam = goalsAwayTeam;
        updateFootballClubs();
    }

    public Optional<FootballClub> getWinner() {
        if(goalsHomeTeam.equals(goalsAwayTeam))
            return Optional.empty();
        return (goalsHomeTeam > goalsAwayTeam ?
Optional.of(homeTeam):Optional.of(awayTeam));
    }

    private void updateFootballClubs() {
        if(goalsHomeTeam.equals(goalsAwayTeam)) {
            homeTeam.addDraw(this);
            awayTeam.addDraw(this);
        } else if(goalsHomeTeam.compareTo(goalsAwayTeam) > 0) {
            homeTeam.addWin(this);
            awayTeam.addLoss(this);
        } else {
            homeTeam.addLoss(this);
            awayTeam.addWin(this);
        }
    }

    @Override
    public String toString() {
        return homeTeam + " " + goalsHomeTeam + ":" + goalsAwayTeam + " " +
awayTeam;
    }
}
```

```java
@NodeEntity
public class Coach {
    @Id
    @GeneratedValue
    private Long coachId;
    private String name;

    public Coach() {}

    public Coach(String name) {
        this.name = name;
    }

    public Long getCoachId() {
        return coachId;
    }

    public String getName() {
        return name;
    }
}
```

```java
@NodeEntity
public class Footballer {
    @Id
    @GeneratedValue
    private Long footballerId;
    private String name;

    public Footballer() {}

    public Footballer(String name) {
        this.name = name;
    }

    public Long getFootballerId() {
        return footballerId;
    }

    public String getName() {
        return name;
    }
}
```

```java
@RelationshipEntity(type = "TRAINS")
public class CoachContract {
    @Id
    @GeneratedValue
    private Long contractId;
    @StartNode
    private Coach coach;
    @EndNode
    private FootballClub footballClub;
    private LocalDate startDate;
    private LocalDate endDate;

    public CoachContract() {}

    public CoachContract(Coach coach, FootballClub footballClub, LocalDate
startDate, LocalDate endDate) {
        this.coach = coach;
        this.footballClub = footballClub;
        this.startDate = startDate;
        this.endDate = endDate;
    }

    public Long getContractId() {
        return contractId;
    }

    public Coach getCoach() {
        return coach;
    }

    public FootballClub getFootballClub() {
        return footballClub;
    }

    public LocalDate getStartDate() {
        return startDate;
    }

    public LocalDate getEndDate() {
        return endDate;
    }

    public void setEndDate(LocalDate endDate) {
        this.endDate = endDate;
    }
```

```
    @Override
    public String toString() {
        return "Coach Contract: " + contractId;
    }
}
```

```
@RelationshipEntity(type = "PLAYS_FOR")
public class FootballerContract {
    @Id
    @GeneratedValue
    private Long contractId;

    @StartNode
    private Footballer footballer;

    @EndNode
    private FootballClub footballClub;

    private LocalDate startDate;
    private LocalDate endDate;

    public FootballerContract() {}

    public FootballerContract(Footballer footballer, FootballClub
footballClub, LocalDate startDate, LocalDate endDate) {
        this.footballer = footballer;
        this.footballClub = footballClub;
        this.startDate = startDate;
        this.endDate = endDate;
    }

    public Long getContractId() {
        return contractId;
    }

    public Footballer getFootballer() {
        return footballer;
    }

    public FootballClub getFootballClub() {
        return footballClub;
    }

    public LocalDate getStartDate() {
        return startDate;
```

```
    }

    public LocalDate getEndDate() {
        return endDate;
    }

    public void setEndDate(LocalDate endDate) {
        this.endDate = endDate;
    }

    @Override
    public String toString() {
        return "Footballer Contract: " + contractId;
    }
}
```

```
@Configuration
@ComponentScan({"repository", "model", "service"})
@EnableNeo4jRepositories("repository")
@EnableTransactionManagement
public class PersistenceContext {
    @Bean
    public org.neo4j.ogm.config.Configuration configuration() {
        return  new org.neo4j.ogm.config.Configuration.Builder().build();
    }

    @Bean
    public SessionFactory sessionFactory() {
        return new SessionFactory(configuration(), "model");
    }

    @Bean
    public Neo4jTransactionManager transactionManager() {
        return new Neo4jTransactionManager(sessionFactory());
    }
}
```

```java
@Repository
public interface FootballClubRepository extends Neo4jRepository<FootballClub,
Long> {
    List<FootballClub> findByName(String name);
}
```

```java
@Repository
public interface MatchRepository extends Neo4jRepository<Match, Long> {
}
```

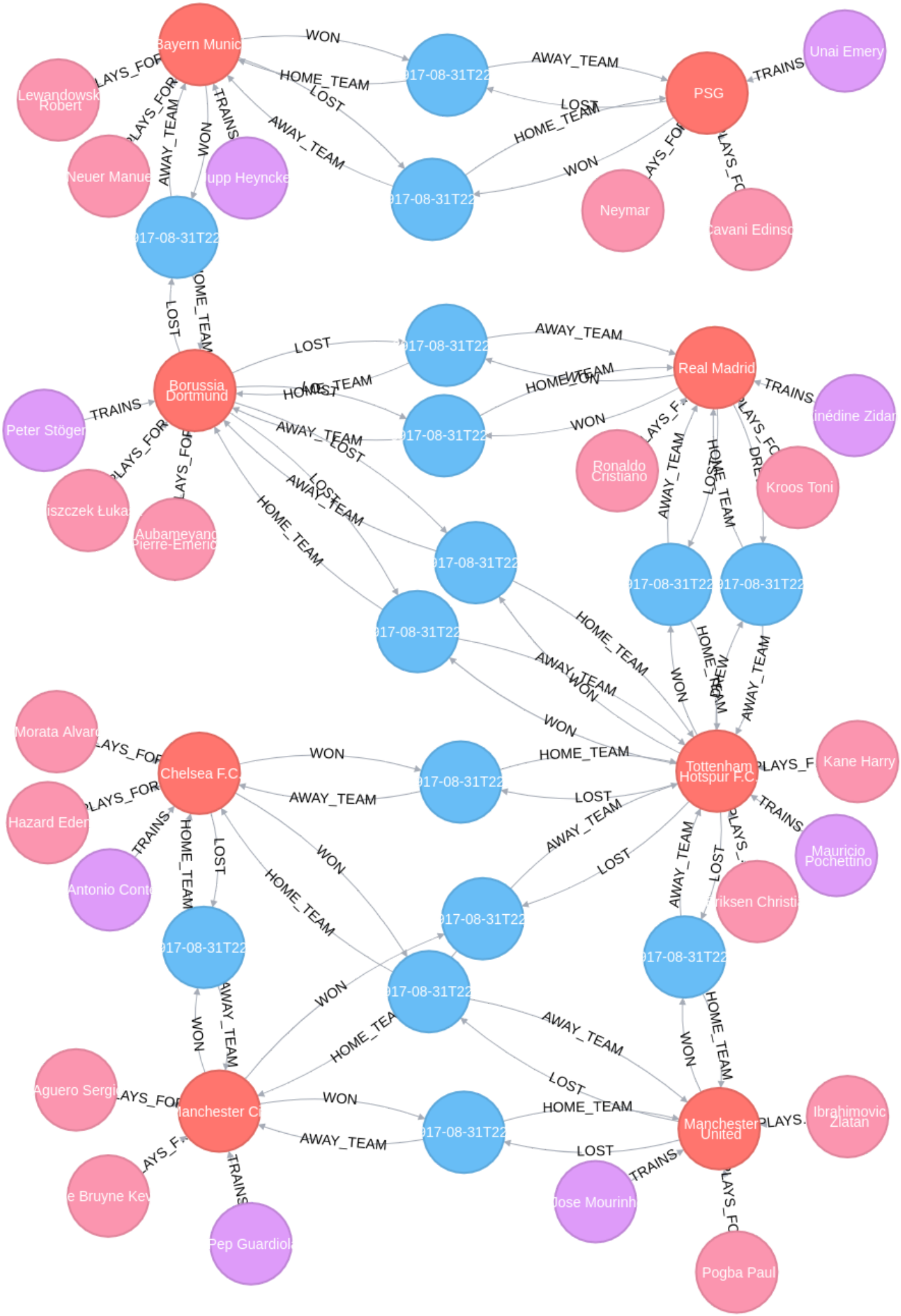
```java
@Repository
public interface CoachRepository extends Neo4jRepository<Coach, Long> {
    List<Coach> findByName(String name);
}
```

```java
@Repository
public interface FootballerRepository extends Neo4jRepository<Footballer,
Long> {
    List<Footballer> findByName(String name);
}
```

3. (1pkt) Napisać bardzo prosty populator danych. Może to być zwykły Main czy Unit-test. Może być bardzo prosty, ale tak, aby było po 5 różnych obiektów i 10 relacji każdego typu. Proszę o wklejenie kodu/zdjęcia.

```
@Service
public class DataPopulator {
    @Autowired
    private FootballClubRepository footballClubRepository;
    @Autowired
    private CoachRepository coachRepository;
    @Autowired
    private FootballerRepository footballerRepository;
    @Autowired
    private MatchRepository matchRepository;

    public void populateData() {
        String [] clubNames = {
                "Manchester City", "Manchester United", "Chelsea F.C.",
                "Tottenham Hotspur F.C.", "Bayern Munich", "Borussia
Dortmund",
                "FC Barcelona", "Real Madrid", "PSG"
        };
        String [] coachNames = {
                "Pep Guardiola", "Jose Mourinho", "Antonio Conte",
                "Mauricio Pochettino", "Jupp Heynckes", "Peter Stöger",
                "Ernesto Valverde", "Zinédine Zidane", "Unai Emery"};
        String [][] footballerNames = {
                {"Aguero Sergio", "De Bruyne Kevin"},
                {"Ibrahimovic Zlatan", "Pogba Paul"},
                {"Hazard Eden", "Morata Alvaro"},
                {"Kane Harry", "Eriksen Christian"},
                {"Lewandowski Robert", "Neuer Manuel"},
                {"Aubameyang Pierre-Emerick", "Piszczek Łukasz"},
                {"Messi Lionel", "Suarez Luis"},
                {"Ronaldo Cristiano", "Kroos Toni"},
                {"Neymar", "Cavani Edinson"}
        };
        String [][] matches = {
                {"Bayern Munich", "PSG", "3", "1"},
                {"PSG", "Bayern Munich", "3", "0"},
                {"Borussia Dortmund", "Tottenham Hotspur F.C.", "1", "2"},
                {"Tottenham Hotspur F.C.", "Borussia Dortmund", "3", "1"},
                {"Tottenham Hotspur F.C.", "Real Madrid", "3", "1"},
                {"Real Madrid", "Tottenham Hotspur F.C.", "1", "1"},
```

```
                {"Real Madrid", "Borussia Dortmund", "3", "2"},
                {"Borussia Dortmund", "Real Madrid", "1", "3"},
                {"Manchester United", "Manchester City", "1", "2"},
                {"Manchester United", "Tottenham Hotspur F.C.", "1", "0"},
                {"Chelsea F.C.", "Manchester United", "1", "0"},
                {"Manchester City", "Tottenham Hotspur F.C.", "4", "1"},
                {"Chelsea F.C.", "Manchester City", "0", "1"},
                {"Tottenham Hotspur F.C.", "Chelsea F.C.", "1", "2"},
                {"Borussia Dortmund", "Bayern Munich", "1", "3"}
        };

        Map<String, FootballClub> clubs = new HashMap<>();
        for(int i = 0; i < clubNames.length; i++) {
            FootballClub club = new FootballClub(clubNames[i]);
            clubs.put(clubNames[i], club);
            footballClubRepository.save(club);
            Coach coach = new Coach(coachNames[i]);
            coachRepository.save(coach);

            club.setCoach(
                    new CoachContract(coach, club,
                            LocalDate.of(2017, 8, 1),
                            LocalDate.of(2018,8,1)));

            for(String footballerName : footballerNames[i]) {
                Footballer footballer = new Footballer(footballerName);
                footballerRepository.save(footballer);
                club.addFoorballer(
                        new FootballerContract(
                                footballer,
                                club,
                                LocalDate.of(2017, 8, 1),
                                LocalDate.of(2018,8,1)));
            }
        }
        for(String[] matchData : matches) {
            Match match = new Match(
                    clubs.get(matchData[0]),
                    clubs.get(matchData[1]),
                    LocalDate.of(2017, 8, 1),
                    Integer.valueOf(matchData[2]),
                    Integer.valueOf(matchData[3]));
            matchRepository.save(match);
        }
    }
}
```


Bayern Munic
Lewandowsk Robert
Neuer Manue
upp Heyncke
PSG
Unai Emery
Neymar
avani Edinso
Borussia Dortmund
Peter Stöger
iszczek Łuka
Aubameyang Pierre-Emeric
Real Madrid
inédine Zidar
Ronaldo Cristiano
Kroos Toni
Tottenham Hotspur F.C.
Kane Harry
Mauricio Pochettino
riksen Christi
Chelsea F.C.
Morata Alvar
Hazard Eden
Antonio Conte
Manchester Ci
Aguero Sergi
e Bruyne Kev
Pep Guardiol
Manchester United
Ibrahimovic Zlatan
Jose Mourinh
Pogba Paul
917-08-31T22
WON
LOST
HOME_TEAM
AWAY_TEAM
PLAYS_FOR
TRAINS

4. (1pkt) Napisać funkcję do pobrania wszystkich relacji dla danego węzła. Proszę o wklejenie kodu/zdjęcia oraz o przygotowanie możliwości przetestowania tego (Main/UnitTest) wraz ze zdjęciem/kopią wyniku funkcji.

```
public void printAllRelations(Long id) {
        Session session = sessionFactory.openSession();
        Map<String, Long> bindings = new HashMap<>();
        bindings.put("id", id);

        String query =
                "MATCH (a)-[r]-(b) " +
                "WHERE ID(a) = $id " +
                "RETURN a, b, r, LABELS(a) AS aLabel, LABELS(b) AS bLabel,
TYPE(r) as rType";

        Result res = session.query(query, bindings);

        StringBuilder builder = new StringBuilder();
        res.forEach(relation -> {
            builder.append(relation.get("a").toString())
                    .append("\t->\t")
                    .append(((String[])relation.get("aLabel"))[0])
                    .append("\t->\t")
                    .append(relation.get("r").toString())
                    .append(" \t->\t")
                    .append(relation.get("rType").toString())
                    .append(" \t->\t")
                    .append(((String[])relation.get("bLabel"))[0])
                    .append(" \t->\t")
                    .append(relation.get("b").toString())
                    .append("\n");
        });

        System.out.println(builder);
    }
```

```
RelationsService relationsService = context.getBean(RelationsService.class);
System.out.println("Bayern relations:");
FootballClubRepository clubRepository =
context.getBean(FootballClubRepository.class);
FootballClub bayern = clubRepository.findByName("Bayern Munich").get(0);
relationsService.printAllRelations(bayern.getFootballClubId());
```

```
Bayern relations:
Bayern Munich   ->  FootballClub    ->  (63)-[AWAY_TEAM]->(34)  ->  AWAY_TEAM   ->  Match   ->  1:3
Bayern Munich   ->  FootballClub    ->  (34)-[WON]->(63)    ->  WON     ->  Match   ->  1:3
Bayern Munich   ->  FootballClub    ->  (34)-[LOST]->(2)    ->  LOST    ->  Match   ->  3:0
Bayern Munich   ->  FootballClub    ->  (2)-[AWAY_TEAM]->(34)   ->  AWAY_TEAM   ->  Match   ->  3:0
Bayern Munich   ->  FootballClub    ->  Footballer Contract: 7  ->  PLAYS_FOR   ->  Footballer  ->  Neuer Manuel
Bayern Munich   ->  FootballClub    ->  Footballer Contract: 6  ->  PLAYS_FOR   ->  Footballer  ->  Lewandowski Robert
Bayern Munich   ->  FootballClub    ->  (34)-[WON]->(54)    ->  WON     ->  Match   ->  3:1
Bayern Munich   ->  FootballClub    ->  Coach Contract: 2   ->  TRAINS  ->  Coach   ->  Jupp Heynckes
Bayern Munich   ->  FootballClub    ->  (54)-[HOME_TEAM]->(34)  ->  HOME_TEAM   ->  Match   ->  3:1
```

5. (2pkt) Napisać funkcje do znalezienia ścieżki dla danych dwóch węzłów. Proszę o wklejenie kodu/zdjęcia oraz o przygotowanie możliwości przetestowania tego (Main/UnitTest) wraz ze zdjęciem/kopią wyniku funkcji.

```
public void printShortestPath(Long idA, Long idB) {
        Session session = sessionFactory.openSession();
        Map<String, Long> bindings = new HashMap<>();
        bindings.put("idA", idA);
        bindings.put("idB", idB);

        String query =
                "MATCH path=shortestPath((a)-[*]-(b)) " +
                "WHERE ID(a) = $idA AND ID(b) = $idB " +
                "RETURN a, b, LABELS(a) AS aLabel, LABELS(b) AS bLabel,
NODES(path) AS npath, RELS(path) AS rpath LIMIT 1";

        Result res = session.query(query, bindings);

        StringBuilder builder = new StringBuilder();

        for (Map<String, Object> path : res) {
            ArrayList<Object> nodes = (ArrayList<Object>) path.get("npath");
            ArrayList<Object> rels = (ArrayList<Object>) path.get("rpath");

            for (int i = 0; i < nodes.size(); i++) {
                builder.append(nodes.get(i).toString());
                if(i < nodes.size() - 1)
                    builder.append("\t->\t");

                if (i < rels.size()) {

builder.append(rels.get(i).toString().replaceAll("[\\W\\d]",
"")).append("\t->\t").append(nodes.get(i+1).toString()).append("\n");
                }
            }
        }

        System.out.println(builder);
    }
```

```
System.out.println("Lewandowski->Hazard shortest path:");
FootballerRepository footballerRepository =
context.getBean(FootballerRepository.class);
Footballer lewy = footballerRepository.findByName("Lewandowski
Robert").get(0);
Footballer hazard = footballerRepository.findByName("Hazard Eden").get(0);
relationsService.printShortestPath(lewy.getFootballerId(),
hazard.getFootballerId());
```

```
Lewandowski->Hazard shortest path:
Lewandowski Robert  ->  FootballerContract  ->  Bayern Munich
Bayern Munich   ->  AWAY_TEAM   ->  1:3
1:3 ->  HOME_TEAM   ->  Borussia Dortmund
Borussia Dortmund   ->  LOST    ->  3:1
3:1 ->  WON ->  Tottenham Hotspur F.C.
Tottenham Hotspur F.C.  ->  LOST    ->  1:2
1:2 ->  AWAY_TEAM   ->  Chelsea F.C.
Chelsea F.C.    ->  FootballerContract  ->  Hazard Eden
Hazard Eden
```

```
public class Main {
    private static AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(PersistenceContext.class);
    public static void main(String [] args) {
        DataPopulator dataPopulator = context.getBean(DataPopulator.class);
        dataPopulator.populateData();
        RelationsService relationsService =
context.getBean(RelationsService.class);

        System.out.println("Bayern relations:");
        FootballClubRepository clubRepository =
context.getBean(FootballClubRepository.class);
        FootballClub bayern = clubRepository.findByName("Bayern
Munich").get(0);
        relationsService.printAllRelations(bayern.getFootballClubId());

        System.out.println("Lewandowski->Hazard shortest path:");
        FootballerRepository footballerRepository =
context.getBean(FootballerRepository.class);
        Footballer lewy = footballerRepository.findByName("Lewandowski
Robert").get(0);
        Footballer hazard = footballerRepository.findByName("Hazard
Eden").get(0);
        relationsService.printShortestPath(lewy.getFootballerId(),
hazard.getFootballerId());
    }
}
```