WP NoSQL und BigData
Aufgabenblatt 2
Team: Igor Arkhipov und Jannik Bruhns

4.
a)
Datenbankstruktur:
Beispiel aus plz.dat:
{ "_id" : "01001", "city" : "AGAWAM", "loc" : [-72.622739, 42.070206], "pop" : 15338, "state" : "MA" }

=>
Key (Postleitzahl als String): "01001"
Value (JSON): {"city" : "AGAWAM", "loc" : [-72.622739, 42.070206], "pop" : 15338, "state" : "MA"}

Import:

```
package com.nosql;

import org.json.JSONObject;
import redis.clients.jedis.Jedis;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Set;

public class Redis {
    private static Jedis jedis;

    public static void main(String[] args) throws IOException {
        jedis = new Jedis("localhost", 6379);
        System.out.println(jedis.ping());

        importJson();
    }

    public static void importJson() throws IOException {
        FileReader freader = new FileReader("./res/plz.data");
        BufferedReader breader = new BufferedReader(freader);
        String dataline = breader.readLine();

        while (dataline != null) {
            JSONObject jsonObject = new JSONObject(dataline);
            String key = (String) jsonObject.get("_id");
            jsonObject.remove(key);
            String value = jsonObject.toString();
            jedis.set(key, value);
            dataline = breader.readLine();
        }
        breader.close();
    }
}
```

b)

```
public class Redis {
    private static Jedis jedis;

    public static void main(String[] args) throws IOException {
```

```
        jedis = new Jedis("localhost", 6379);
        System.out.println(jedis.ping());

        System.out.println(getValue("01001"));
    }

    public static String getValue(String key) {
        String value = jedis.get(key);
        if (value != "") {
            JSONObject jsonObject = new JSONObject(value);
            String city = jsonObject.get("city").toString();
            String state = jsonObject.get("state").toString();

            return "City > " + city + " and State > " + state;
        }
        return "Keine zutreffende Eingabe!";
    }
}

c)
public class Redis {
    private static Jedis jedis;

    public static void main(String[] args) throws IOException {
        jedis = new Jedis("localhost", 6379);
        System.out.println(jedis.ping());

        System.out.println(findKey("TUMTUM")); //4c
        System.out.println(findKey("HAMBURG")); //4c
    }

    public static String findKey(String city) {
        Set<String> keys = jedis.keys("*");
        List<String> answer = new ArrayList<String>();

        for (String key : keys) {
            String value = jedis.get(key);
            JSONObject jsonObject = new JSONObject(value);
            if (city.equals(jsonObject.get("city").toString())) {
                answer.add(key);
            }
        }

        Collections.sort(answer);

        if (answer.isEmpty()) {
            return "Keine zutreffende Eingabe!";
        }
        return answer.toString();
    }
}

Ergebnisse:

City > AGAWAM and State > MA
[99034]
[07419, 14075, 19526, 51640, 54411, 55339, 62045, 71339, 71646]

5.
a)
Create(AI:Modul { name: 'AI' }),
(AD:Modul { name: 'AD' }),
(AF:Modul { name: 'AF' }),
(BS:Modul { name: 'BS' }),
```

```
(BW1:Modul { name: 'BW1' }),
(BW2:Modul { name: 'BW2' }),
(DB:Modul { name: 'DB' }),
(GI:Modul { name: 'GI' }),
(GKA:Modul { name: 'GKA' }),
(IS:Modul { name: 'IS' }),
(LB:Modul { name: 'LB' }),
(MG:Modul { name: 'MG' }),
(PR1:Modul { name: 'PR1' }),
(PR2:Modul { name: 'PR2' }),
(RN:Modul { name: 'RN' }),
(RMP:Modul { name: 'RMP' }),
(SE1:Modul { name: 'SE1' }),
(SE2:Modul { name: 'SE2' }),
(wpdbd:Modul { name: 'wp-dbd' }),
(wphci:Modul { name: 'wp-hci' }),
(wpnosql:Modul { name: 'wp-nosql' }),
(GW1:Modul { name: 'GW1' }),
(GW2:Modul { name: 'GW2' }),
(GW3:Modul { name: 'GW3' }),
(Projekt:Modul { name: 'Projekt' }),
(Seminar:Modul { name: 'Seminar' }),
((wpnosql) -[:uses]-> (DB)),
((wpnosql) -[:uses]-> (GKA)),
((wpdbd) -[:uses]-> (DB)),
((AI) -[:uses]-> (SE1)),
((AI) -[:uses]-> (SE2)),
((AD) -[:uses]-> (PR1)),
((AD) -[:uses]-> (PR2)),
((AD) -[:uses]-> (GI)),
((AF) -[:uses]-> (GI)),
((BS) -[:uses]-> (GI)),
((BW2) -[:uses]-> (BW1)),
((BW2) -[:uses]-> (DB)),
((DB) -[:uses]-> (PR1)),
((GKA) -[:uses]-> (PR1)),
((GKA) -[:uses]-> (GI)),
((GKA) -[:uses]-> (LB)),
((GKA) -[:uses]-> (AF)),
((IS) -[:uses]-> (PR1)),
((IS) -[:uses]-> (PR2)),
((IS) -[:uses]-> (DB)),
((IS) -[:uses]-> (AD)),
((IS) -[:uses]-> (AF)),
((IS) -[:uses]-> (SE1)),
((LB) -[:uses]-> (GI)),
((LB) -[:uses]-> (MG)),
((PR2) -[:uses]-> (PR1)),
((RN) -[:uses]-> (BS)),
((RMP) -[:uses]-> (GI)),
((SE1) -[:uses]-> (PR1)),
((SE1) -[:uses]-> (DB)),
((SE2) -[:uses]-> (SE1)),
((SE2) -[:uses]-> (DB)),
((SE2) -[:uses]-> (PR1)),
((SE2) -[:uses]-> (PR2))

b)
#
# Abfrage: Welche Module sind für das NoSQL/Big Data-Modul nützlich?

MATCH ((wpnosql) -[:uses]-> (X)) RETURN X.name

#
```

Abfrage: Welche Module wurden bisher im Studium nicht wieder genutzt?

```
MATCH (X) WHERE NOT (X) <-[:uses]- () return X.name
```

6.
a)
Die DB wurde gespeichert und alte (lokale) überschrieben.

b)
„ConceptNet is a semantic network based on the information in the OMCS database. ConceptNet is expressed as a directed graph whose nodes are concepts, and whose edges
are assertions of common sense about these concepts. Concepts represent sets of closely
related natural language phrases, which could be noun phrases, verb phrases, adjective
phrases, or clauses.“

c)

```
MATCH (a),(b) WHERE (b)-[:IsA]->(a) AND b.id="/c/en/baseball" RETURN a.id
```

```
/c/en/game
/c/en/sport
/c/en/person
/c/en/organization
/c/en/athlete
/c/en/popular_sport
/c/en/band
/c/en/activity
/c/en/baseball_player
/c/en/hobby
/c/en/music_group
/c/en/not_play_professionally
/c/en/good_before_all_player
/c/en/speech
/c/en/firewood
/c/en/perl
/c/en/c++
/c/en/organisation
/c/en/gridiron_football_player

Started streaming 19 records after 3 ms and completed after 12466 ms.
```