Huibo Zhao
hz2480
COMS4111 SEC003
HW5
Dec.14th.2017

Part1:

Sql:

```
select appearances.playerID,appearances.teamID,appearances.yearID from appearances join
    ((select yearID, teamID from appearances where playerid = "napolmi01") as a)
    where appearances.yearID = a.yearID and appearances.teamID = a.teamID and
        appearances.yearID > 2010 and appearances.yearID < 2017 and appearances.playerid != "napolmi01"
    order by appearances.playerID desc;
```

100% 21:13

Result Grid | Filter Rows: Search | Export:

playerID	teamID	yearID
youngmi02	TEX	2011
youngmi02	TEX	2012
wrighst01	BOS	2013
wrighst01	BOS	2014
wrighst01	BOS	2015
workmbr01	BOS	2013
workmbr01	BOS	2014
wilsocj01	TEX	2011
wilsobo02	TEX	2015
wilsoal01	BOS	2013
wilsoal01	BOS	2014
weeksje01	BOS	2014
weeksje01	BOS	2015
webstal01	BOS	2013

mastersmall 137 | appearancessmall 138 | Result 139

Neo4j:

```
$ match(n1:Player {player_id: 'napolmi01'})-[r1:APPEARANCE]->(t:Team)<-
  [r2:APPEARANCE]-(n2:Player) where r1.year = r2.year return
  n2.player_id,t.team_id,r1.year order by n2.player_id desc
```

```
$ match(n1:Player {player_id: 'napolmi01'})-[r1:APPEARANCE]->(t:Team)<-[r2:AP…
```

n2.player_id	t.team_id	r1.year
"youngmi02"	"TEX"	"2012"
"youngmi02"	"TEX"	"2011"
"wrighst01"	"BOS"	"2015"
"wrighst01"	"BOS"	"2014"
"wrighst01"	"BOS"	"2013"
"workmbr01"	"BOS"	"2014"
"workmbr01"	"BOS"	"2013"
"wilsocj01"	"TEX"	"2011"
"wilsobo02"	"TEX"	"2015"
"wilsoal01"	"BOS"	"2014"
"wilsoal01"	"BOS"	"2013"
"weeksje01"	"BOS"	"2015"
"weeksje01"	"BOS"	"2014"
"webstal01"	"BOS"	"2014"
"webstal01"	"BOS"	"2013"
"villabr02"	"BOS"	"2013"

Part 2:

Initially, we have nothing stored in Redis.

```
127.0.0.1:6379> keys *
(empty list or set)
127.0.0.1:6379>
```

Then, we call find_by_id("napolmi01")

```
public static void main(String[] args) {
    System.out.println(find_by_id("napolmi01"));
}
```

```
orld.java   hw5func1.java
Messages   jGRASP Messages   Run I/O   Interactions
  ----jGRASP: operation complete.

  ----jGRASP exec: java hw5.hw5func1
Connection to server sucessfully
Will load data from MySQL to Redis..
Successfully add to redis!
{nameLast=Napoli, nameFirst=Mike, birthYear=1981, birthCountry=USA, playerID=napolmi01}

  ----jGRASP: operation complete.
```

Checking Redis again, we could see that the data is loaded into Redis

```
127.0.0.1:6379> keys *
 1) "players:napolmi01"
127.0.0.1:6379>
```

```
127.0.0.1:6379> hgetall players:napolmi01
 1) "nameFirst"
 2) "Mike"
 3) "playerID"
 4) "napolmi01"
 5) "birthCountry"
 6) "USA"
 7) "birthYear"
 8) "1981"
 9) "nameLast"
10) "Napoli"
127.0.0.1:6379>
```

Calling find_by_id function again, it would find data from redis directly.

```
public static void main(String[] args) {
    System.out.println(find_by_id("napolmi01"));
```

```
orld.java | hw5func1.java
Messages | jGRASP Messages | Run I/O | Interactions
----jGRASP: operation complete.

----jGRASP exec: java hw5.hw5func1
Connection to server sucessfully
Found data in Redis
{birthCountry=USA, nameLast=Napoli, nameFirst=Mike, birthYear=1981, playerID=napolmi01}

----jGRASP: operation complete.
```

The above test examines all four functions. The codes are attached below. Please examine. I use a map format instead of JSON, which I got approved by our professor on Piazza.

coms4111 sec003 hw5

Huibo Zhao hz2480

December 16, 2017

Coding

```
package hw5;

import redis.clients.jedis.Jedis;



import java.util.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
import java.io.StringWriter;


import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

import redis.clients.jedis.BinaryJedis;
import org.json.*;
import javax.json.*;

public class hw5func1 {

    private static java.sql.Connection getConnection() {
        java.sql.Connection conn = null;
```

```
    try {
        // You must set the schema, user ID and password for your local date
        conn = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/fina
            "user=root&password=z123456");
    } catch (SQLException ex) {
        // handle any errors
        System.out.println("SQLException: " + ex.getMessage());
        System.out.println("SQLState: " + ex.getSQLState());
        System.out.println("VendorError: " + ex.getErrorCode());

        conn = null;
    }

    return conn;
}

private static Map<String,String> find_mysql_by_id(String id) {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
    Map<String,String> map = new HashMap<String,String>();
    try {
        conn = getConnection();
        stmt = conn.createStatement();
        rs = stmt.executeQuery("SELECT * FROM Master WHERE playerID='" + id
        rs.first();

        String playerID, nameLast, nameFirst, birthYear, birthCountry;

        playerID = rs.getString("playerID");
        nameLast = rs.getString("nameLast");
        nameFirst = rs.getString("nameFirst");
        birthYear = rs.getString("birthYear");
        birthCountry = rs.getString("birthCountry");
        //System.out.println("playerID = " + playerID + ", last name = " + n
        //     ", first name = " + nameFirst + ", birth year = " + birthYear
        map.put("playerID",playerID);
        map.put("nameLast",nameLast);
        map.put("nameFirst",nameFirst);
        map.put("birthYear",birthYear);
```

```
        map.put("birthCountry",birthCountry);

    } catch (SQLException ex) {
        // handle any errors
        System.out.println("SQLException: " + ex.getMessage());
        System.out.println("SQLState: " + ex.getSQLState());
        System.out.println("VendorError: " + ex.getErrorCode());
    } finally {
        // it is a good idea to release
        // resources in a finally{} block
        // in reverse-order of their creation
        // if they are no-longer needed

        if (rs != null) {
            try {
                rs.close();
            } catch (SQLException sqlEx) {} // ignore

            rs = null;
        }

        if (stmt != null) {
            try {
                stmt.close();
            } catch (SQLException sqlEx) {} // ignore

            stmt = null;
        }
    }
    return map;
}

private static Map<String,String> find_redis_by_id(String id) {
    Map < String, String > result;

    //Connecting to Redis server on localhost
    Jedis jedis = new Jedis("localhost");
    System.out.println("Connection to server sucessfully");
    result = jedis.hgetAll("players:" + id);
    jedis.close();
    return result;
```

```
    }

    private static void add_to_redis(String id, Map<String,String> data) {
        Jedis jedis = new Jedis("localhost");
        String result = jedis.hmset("players:"+id,data);
        if (result.substring(0,2).equals("OK"))
           System.out.println("Successfully add to redis!");

    }


    private static Map<String,String> find_by_id(String id) {
        Map<String, String> result;

        result = find_redis_by_id(id);

        if(result.isEmpty()) {
           System.out.println("Will load data from MySQL to Redis..");
           result = find_mysql_by_id(id);
           add_to_redis(id,find_mysql_by_id(id));
        } else {
           System.out.println("Found data in Redis");
        }

        return result;


    }

    public static void main(String[] args) {
        System.out.println(find_by_id("napolmi01"));
    }

}
```