



Politechnika Rzeszowska

Wydział Elektrotechniki i Informatyki

Katedra Informatyki i Automatyki

# **Bazy danych**

**Laboratorium – zadanie domowe**

Proxy Cache'ujące

Wykonał: Przysaś Artur

II EF-DI;L05

## Spis treści

Problematyka zadania .....	3
Redis.....	3
Baza danych .....	4
ORM .....	5
Program główny .....	6
Działanie .....	7

## Problematyka zadania

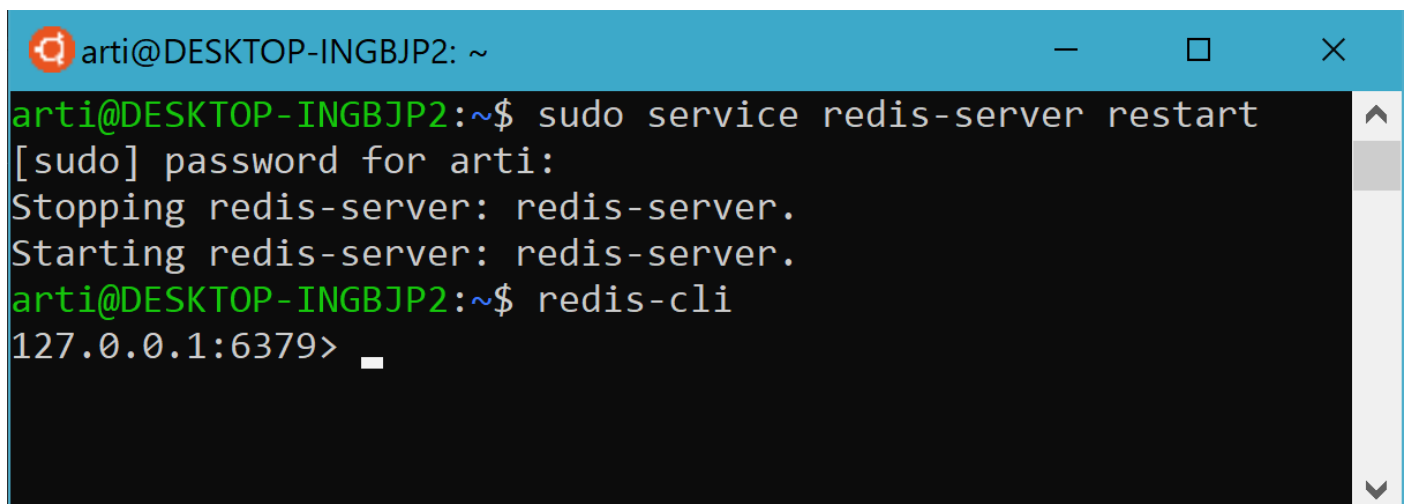
Celem zadania było napisać proxy Cache'ujące dzięki któremu można odczytać dane z bazy danych. W zaprojektowanym systemie znajduje się jednak proxy w postaci Redisa. Gdy dane są zawarte w proxy, pobierane są one w pierwszej kolejności z niego.

Napisany kod jest parametryzowany przez:

- dopuszczalny czas nieświeżości danych
- adres Redisa
- adres bazy danych

## Redis

W celu wykonania ćwiczenia potrzeba było zgłębić wiedzę na temat tworzenia serwera Redisa. Program ten wspiera jedynie systemy Linux, oraz MacOS. Wykonujący ćwiczenie na co dzień korzysta z systemu Windows 10. Z tego powodu posłużono się podsystemem Linuksa dla Windowsa. Dzięki temu konfiguracja Redisa przebiegała jak na każdej maszynie z zainstalowanym Ubuntu (czyli wybraną dystrybucją Linuksa).

A screenshot of a terminal window with a blue title bar. The title bar contains the text 'arti@DESKTOP-INGBJP2: ~' and standard window control buttons (minimize, maximize, close). The terminal text shows the user 'arti' running 'sudo service redis-server restart'. It prompts for a password, then shows 'Stopping redis-server: redis-server.' and 'Starting redis-server: redis-server.'. Next, the user runs 'redis-cli', and the prompt changes to '127.0.0.1:6379>'.

```
arti@DESKTOP-INGBJP2: ~  
arti@DESKTOP-INGBJP2:~$ sudo service redis-server restart  
[sudo] password for arti:  
Stopping redis-server: redis-server.  
Starting redis-server: redis-server.  
arti@DESKTOP-INGBJP2:~$ redis-cli  
127.0.0.1:6379> _
```

Rys. 1. Działający Redis

## Baza danych

Do wykonania ćwiczenia wykorzystano bazę danych „Chinook.db” znaną z zajęć laboratoryjnych. Została ona wykonana w systemie SQLite. Napisany program umożliwia korzystanie z tabeli „albums” tej bazy.

	AlbumId	Title	ArtistId
	Filter	Filter	Filter
1	1	For Those Abo...	1
2	2	Balls to the Wall	2
3	3	Restless and ...	2
4	4	Let There Be R...	1
5	5	Big Ones	3
6	6	Jagged Little Pill	4
7	7	Facelift	5
8	8	Warner 25 Anos	6
9	9	Plays Metallica...	7
10	10	Audioslave	8
11	11	Out Of Exile	8
12	12	BackBeat Soun...	9
13	13	The Best Of Bil...	10
14	14	Alcohol Fueled...	11
15	15	Alcohol Fueled...	11
16	16	Black Sabbath	12
17	17	Black Sabbath ...	12
18	18	Body Count	13

Rys. 2. Fragment tabelu albums

# ORM

Mapowanie obiektowo-relacyjne to technika konwertowania danych pomiędzy niekompatybilnymi typami systemów, przy użyciu obiektowych języków programowania. Zastosowanie ORMu umożliwiło działania na bazie danych z poziomu programu pisanego w języku Java.

Wybrany do wykonania ćwiczenia ORM'em jest ORMLite. Jest to bardzo popularne i dobrze udokumentowane rozwiązanie, często wykorzystywane w programach wykonywanych w języku Java.

Klasa albums odwzorowująca tabelę w bazie danych:

```
1. import com.j256.ormlite.field.DatabaseField;
2. import com.j256.ormlite.table.DatabaseTable;
3.
4. @DatabaseTable(tableName = "albums")
5. public class albums {
6.     public static final String ID_FIELD_NAME = "AlbumId";
7.     public static final String TITLE_FIELD_NAME = "Title";
8.     public static final String ARTISTID_FIELD_NAME = "ArtistId";
9.
10.    @DatabaseField(columnName = ID_FIELD_NAME, canBeNull = false)
11.    private int AlbumId;
12.
13.    @DatabaseField(columnName = TITLE_FIELD_NAME)
14.    private String Title;
15.
16.    @DatabaseField(columnName = ARTISTID_FIELD_NAME)
17.    private int ArtistId;
18.
19.    albums(){
20.
21.    }
22.
23.    public int getAlbumId(){
24.        return AlbumId;
25.    }
26.
27.    public void setAlbumId(){
28.        this.AlbumId = AlbumId;
29.    }
30.
31.
32.    public String getTitle(){
33.        return Title;
34.    }
35.    public void setTitle(){
36.        this.Title = Title;
37.    }
38.
39.
40.    public int getArtistId(){
41.        return ArtistId;
42.    }
43.
44.    public void setArtistId(){
45.        this.ArtistId = ArtistId;
46.    }
47.
48. }
```

# Program główny

Kod odpowiedzialny za główną funkcjonalność programu

```
1. import com.google.gson.Gson;
2. import com.google.gson.reflect.TypeToken;
3. import com.j256.ormlite.dao.Dao;
4. import com.j256.ormlite.dao.DaoManager;
5. import com.j256.ormlite.dao.GenericRawResults;
6. import com.j256.ormlite.jdbc.JdbcConnectionSource;
7. import com.j256.ormlite.support.ConnectionSource;
8. import redis.clients.jedis.Jedis;
9.
10.
11. import java.io.IOException;
12. import java.lang.reflect.Type;
13. import java.sql.SQLException;
14. import java.util.Arrays;
15. import java.util.List;
16. import java.util.Scanner;
17.
18. public class redis_homework {
19.     public static void main(String[] args) throws SQLException, IOException {
20.         //Łączenie z redisem będącym na localhoście
21.         Jedis jedis = new Jedis("localhost");
22.         //Łączenie z bazą danych
23.         String url = "jdbc:sqlite:D:/chinook.db";
24.         ConnectionSource connectionSource = new JdbcConnectionSource(url);
25.         //Utworzenie DAO dla albums
26.         Dao<albums,Integer> daoAlbums = DaoManager.createDao(connectionSource, albums.class);
27.
28.         Gson gson = new Gson();
29.         while(true) {
30.             Scanner scanner = new Scanner(System.in);
31.             System.out.println("q: wyłączenie programu; *query* from albums *rest of query*: zapytanie do
tabeli albums");
32.
33.             System.out.println("____WPISZ KOMENDĘ____");
34.             String query = scanner.nextLine();
35.
36.             if (query.equals("q")) {
37.                 connectionSource.close();
38.                 break;
39.             }
40.             else {
41.                 //Stworzenie redisowego klucza dla query
42.                 String key = query.toUpperCase().replaceAll(" ", "");
43.
44.                 //Sprawdzanko, czy wyniki są w redisie
45.                 if (jedis.exists(key)){
46.                     showRedisResults(jedis,gson, key);
47.                 }
48.
49.                 else {
50.                     //Pobranie wyników z bazy danych
51.                     List<String[]> resultDB = getDBResults(daoAlbums,query);
52.                     String queryResult = gson.toJson(resultDB);
53.                     jedis.set(key, queryResult);
54.                     jedis.expire(key, 420); //dane wygasają po 420 sekundach
55.                 }
56.             }
57.         }
58.     }
59.     //klasa pokazująca wyniki jeśli są one w redisie
60.     private static void showRedisResults(Jedis jedis, Gson gson, String key){
61.         Type theList = new TypeToken<List<String[]>>().getType();
62.         String fromRedis = jedis.get(key);
63.         List<String[]> resultRedis = gson.fromJson(fromRedis, theList);
64.         resultRedis.forEach(arr -> System.out.println(Arrays.toString(arr)));
65.         System.out.println("_____" + "\n" + "Redis");
66.     }
```

```

67.     //klasa ktora pobiera dane z bazy i je wyswietla
68.     private static List<String[]> getDBResults(Dao<albums,
Integer> daoAlbums, String query) throws SQLException{
69.         GenericRawResults<String[]> rawResults = daoAlbums.queryRaw(query);
70.         List<String[]> resultDB = rawResults.getResults();
71.         resultDB.forEach(ar -> System.out.println(Arrays.toString(ar)));
72.         System.out.println("_____ " + "\n" + "DB");
73.         return resultDB;
74.     }
75. }

```

## Działanie

### Pierwsze wywołanie zapytania:

```

_____ WPISZ KOMENDE _____
select * from albums where ArtistId = 1
[1, For Those About To Rock We Salute You, 1]
[4, Let There Be Rock, 1]

_____
DB

```

Wynik został pobrany z bazy danych.

### Kolejne wywołanie tego samego zapytania:

```

_____ WPISZ KOMENDE _____
select * from albums where ArtistId = 1
[1, For Those About To Rock We Salute You, 1]
[4, Let There Be Rock, 1]

_____
Redis

```

Wynik został podany z Redisa.

### Po odczekaniu ponad 420 sekund:

```

_____ WPISZ KOMENDE _____
select * from albums where ArtistId = 1
[1, For Those About To Rock We Salute You, 1]
[4, Let There Be Rock, 1]

_____
DB

```

Zgodnie z założeniem dane w Redisie wygasły, zostały pobrane z bazy.

### Wpisywanie znaków różnej wielkości i różnej ilości spacji

```

_____ WPISZ KOMENDE _____
SeLeCt          * FROM albums WHere ArtistId          = 1
[1, For Those About To Rock We Salute You, 1]
[4, Let There Be Rock, 1]

_____
Redis

```

Program jest odporny na kaprysy użytkownika co do ilości spacji, czy wielkości liter – dane zostały pobrane z Redisa mimo komendy nie wyglądającej identycznie co wcześniej.