GW2-SRS TRANSFORM

powered by LaTeX

Daniel Lopez: **Transform algorithm**

October 23, 2022

2.0 TRANSFORM

2.1 Introduction

After completing the data extraction, it was needed to do a deep cleaning on the files. The extracted data was displayed on the HTML source as a JSON type, it as well contained lots of information that is entirely related to the statistics in-game. Whether is true that statistics showed player names, player accounts, DPS[1], etc. It also showed information that wasn't needed at all, where we could find EliteInsights information about it's version, release and EVTC[2] version as well.

Therefore, I only wanted to gather clear stats data that could help me in the further analysis. The data I decided to aim for was:

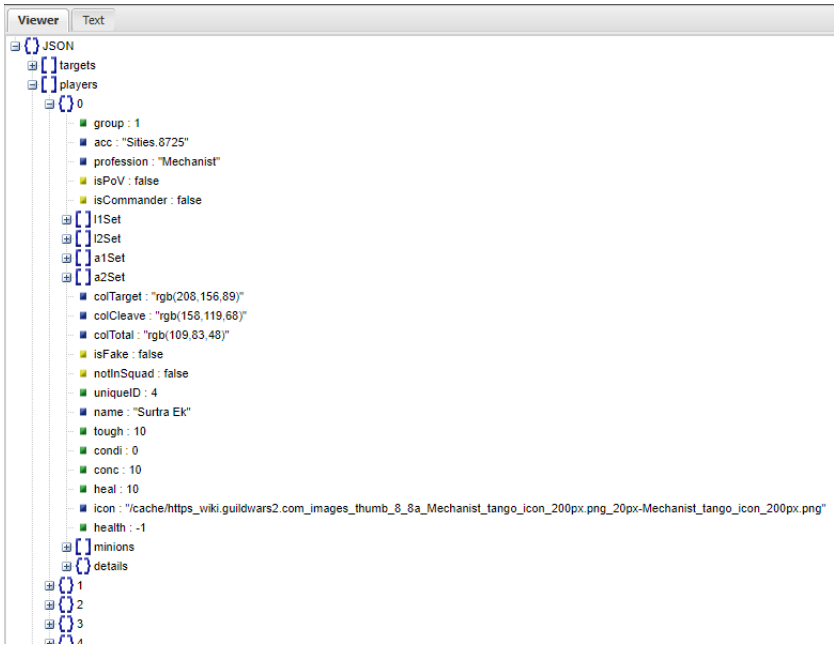
- Player name
- Player Account
- Player Profession/Class
- Player DPS Statistics

	Player's Name	Player's Account	Player's Class	Player's DPS
1	John_Doe	johndoe.9752	Catalyst	35867.43
2	...			
3				

[1]Damage Per Second
[2]Unique log filetype from ArcDPS app

2.2 In-depth data explanation

In order to understand the data, I had to read and investigate the JSON files lots of times. Due to the JSON files size, I decided to use a JSON Viewer[3] to help me with this task.



(a) JSON Viewer formatted schema

(b) Raw JSON view

[3]http://jsonviewer.stack.hu/

Using this method made the search of information extremely easy, for the most part, the essential player data like names, accounts and professions was pretty much finding each player and this data by it's index. I applied a zipped For loop to do the aggregation on SQL databases and on No-SQL databases I used a simple dictionary I created and then passed as a JSON dictionary.

```
for player in data['players']:
    player_group.append(player['group'])
    player_acc.append(player['acc'])
    player_names.append(player['name'])
    player_classes.append(player['profession'])
```

Listing 1: **Basic player data loop**

```
stats_dict = {
        'boss': target,
        'players':{
            'group': player_group,
            'account': player_acc,
            'names': player_names,
            'profession': player_classes,
            'phase_1_dps': player_dps1,
            'phase_2_dps': player_dps2,
            'phase_3_dps': player_dps3
        }
    }
```

Listing 2: **Custom Python stat dictionary**

```
for (name,acc,profession) in zip \\
(player_names,player_acc,player_classes):
```

Listing 3: **Zipped data for SQLite query**

It was important using a zipped For loop since the query need to be executed for every player. This is indeed not efficient if we look into the repetition of the same query for a simple operation but since there are only 10 players per boss fight, it ended up being rather useful.