"Artists & Bands": Shaping Derived Data By Constructing a Redis ⇔ MongoDB Data Pipeline

Avraham Leff

COM3580: Spring 2024

1 Assignment-Specific Packaging

The general packaging is unchanged from the basic "Homework Requirements" (see slides from first lecture and "Homework Policies for COM 3580" on Piazza).

This assignment's "DIR" **must be named** *ArtistsAndBands*. Your report **must be named** *$DIR/ArtistsAndBands.pdf*.

2 Motivation

Most of the assignments in this course involve a single data-store or data-processing system. The purpose of this assignment is to get some experience "reshaping" raw data into something more useful, and to do so by constructing a *Raw data ⇒ Redis ⇒ MongoDB* data pipeline.

Specifically: you will

1. Download a moderately large data-set about artists and bands.
2. Extract information from this data-set, "reshape" the data, and store it in Redis. You may want to view the Redis data as an intermediate form of the data that will eventually be stored in MongoDB. Alternatively: the Redis data should be considered to have independent

value, with its relative flexibility facilitating exploratory data analysis.

3. Reshape the Redis data, storing the second form in MongoDB.

You will demonstrate the value of the data transformations by formulating queries on both the Redis and MongoDB data-sets.

3 The Data-Set

Download the data-set (`group_membership.tsv`). The data-set is a tsv file, whose lines are in the following format.

1. name
2. id
3. member (*artist name*)
4. group (*band name*)
5. role (one or more comma-separated fields)
6. start
7. end

Example:

```
/m/02hrk87      Paul McCartney   The Beatles      Bass
      1958     1970
```

Note: the data-set is incomplete so that not all fields are populated for any given line of data. For example, I'm pretty sure that the `name` field isn't populated at all. However: there will always be n tab-separated fields; if a field value is empty, that means there is no information about the corresponding key for that row.

As you can see, the data are organized on a "per-artist/per band" basis. The artist may therefore appear multiple times, depending on the number of bands she was a member of. In its current format, the data-set has some limitations with respect to extracting facts about an artist's career.

Also: for many purposes, we want data to be organized on a "per-band" basis, and this data is simply not organized properly for "band" queries.

4 Phases

Because I will not be executing your code (note to self: in this iteration), you'll demonstrate successful completion of each of the following three phases by documenting the required results in your writeup file.

4.1 Phase 0

In this phase, you'll implement the specified pipeline in a **single** program written in your language of choice.

You will need to devise a schema for the data that you'll store in Redis. You'll need to devise another schema for the data that you'll store in MongoDB. As you design these schema, keep in mind that you'll eventually be formulating queries against both the Redis data ("Phase 1") and the MongoDB data ("Phase 2").

The pipeline:

1. Reads the file, line-by-line
2. Extracts the *member*, *group*, and *role* fields.

   You are only interested in rows that contain "useful" data: i.e., at a minimum, contain "non-empty" *member* and *group* data
3. Transform the raw data and store it in Redis. This can be done on a line-by-line basis, or after you've consumed the entire data-set.
4. Transform the Redis data and store it in MongoDB.

   I found that some key values triggered a "key too large to index" error from MongoDB. Don't panic: research the details of this message, and it's OK to ignore that document if it's giving you too many problems. (There are alternative solutions as well).

4.1.1 Redis Schema

1. Create associations from “artist name” $\Rightarrow$ “band name”
2. Create associations from “band name” $\Rightarrow$ artists who are members of the band
3. Create associations from “artist & band” $\Rightarrow$ all roles assumed by the specified artist in the specified band

You decide how to implement these associations.

4.1.2 MongoDB Schema

You must devise a “per-band” schema that integrates all artists associated with a band, and each artist’s roles in that band, into a single document.

5 Requirements

Your writeup file must contain the following sections in the specified order.

5.1 Phase 0

Invoke your program (e.g., `app.js`) with the `time` command to document program performance. The screenshot for this phase must contain the output listed below.

1. The program should emit, every $10,000$ of processed input, a line such as *so far: processed* $20,000$ *lines*.
2. At program completion: emit *“Total lines processed: x”*
3. At program completion: emit *“Useful lines processed: x”* (see definition above).
4. When the program completes, the screenshot should include the output of the `time` command.

At the end of this phase, your pipeline program has terminated, and you're ready to translate natural-language queries into queries against the Redis and MongoDB data-stores.

5.1.1 Grading note

Much of your grade will be based on the correctness of the implementation's queries. However: points may be deducted if the implementation has excessively slow duration in this phase. Guidance: my implementation takes less than three seconds.

5.2 Phase 1

Translate the following "natural-language" queries into the corresponding Redis API calls for your implementation. Your screenshots must show the Redis query (as invoked via `redis-cli`) and the Redis response.

1. *Which artists were members of "The Beatles"?*
2. *Which bands was Paul McCartney a member of?*
3. *Which roles did "Cale Harper" play in the band "Horses About Men"?*

5.3 Phase 2

Translate the following "natural-language" queries into the corresponding MongoDB shell calls for your Phase 2 implementation. Your screenshots must show the MongoDB query and the shell's response.

Return only the information that the natural language query asks for: **no more, and no less**!

You may use MongoDB's MQL or the aggregation framework to formulate your queries. All output must be in "pretty" format.

1. *Display a sample band document.*
2. *How many distinct bands in the data set?*
3. *How many artists in the band "Men at Work"?*

4. How many bands did *Paul McCartney* play in?
5. *Find all information about artist John Cooper in the band Skillet.*

   Note: your query can only return information about this artist: i.e., may not return information about any other band member.
6. *List the names of every band that is associated with more than 50 artists, along with the number of artists in that band.*
7. *List the names of artists with more than ten roles, along with the number of roles, sorted by ascending number of roles.*