Flat files vs. Databases

Study the differences between using flat files and databases when searching for data.

In this part of the project you will compare flat files and databases when it comes to organizing and querying data.

Note: For this checkpoint, assign the tag with Key: `Project` and Value: `3.1` for all resources

To get started, launch an m1.small Instance of the AMI – ami-fab52b93. In the directory `/home/ubuntu/songs/`, you will find a comma-separated file `million_songs_metadata.csv`. This is all of the track metadata from the Million Song Dataset.

The file has the following columns:

Col #	Name	MySQL type
1	track_id	varchar(19)
2	title	text
3	song_id	text
4	release	text
5	artist_id	text
6	artist_mbid	text
7	artist_name	text
8	duration	double
9	artist_familiarity	double
10	artist_hotttnesss	double
11	year	int(11)

Table 1: MySQL Data Schema for `million_songs_metadata.csv`

Note: `RELEASE` is a keyword in MySQL, so whenever you reference it, you will have to use the **backtick-escaped** form `` `release` `` instead.

Part 1: Operations on Flat Files

Recall the use of `grep` in Project 1.

1. Use the following to find the records that contain 'The Beatles', while understanding the difference between the 2 commands below:

   ```
   grep -P 'The Beatles' million_songs_metadata.csv
   grep -i -P 'The Beatles' million_songs_metadata.csv
   ```

   You will find that it returns rows that include that string in any column, but what if you wanted just the ones that match the **`artist_name`** field?

2. You can use **`awk`** to find the records with **`artist_name`** field containing 'The Beatles':

   ```
   awk ' BEGIN {FS = ","} ; {if ($7 ~ /The Beatles/) { print; }}'
   million_songs_metadata.csv
   ```

   (**`$7`** denotes the 7th column and **`FS=","`** sets the operator to a comma)

3. Make a note of the number of matches (use `| wc -l`) for steps 1 and 2 above for the checkpoint below.

4. Now, what if we wanted to know something more complicated, like "How many songs are from Michael Jackson, but only from the '80s ?"

   ```
   awk ' BEGIN {FS = ","} ; {if (tolower($7) ~ /michael jackson/ && $11 >=
   1980 && $11 < 1990) { print; }}' million_songs_metadata.csv
   ```

   (Column #11 is the year)

5. Write an **`awk`** program that computes the average song length (using the **`duration`** field) over the million songs. This program should be executed using a single UNIX line. Paste the command into the checkpoint below, when asked.

You can see that as your questions become more complex, it is more difficult to write a single line of code to answer the question. For these reasons (among many others) we want to move this data to a database.

Part 2: Working with MySQL

We will now use MySQL (an open source relational database) to load and process these records.

1. MySQL should be already installed and running on the AMI supplied for this project. To start a MySQL CLI client and connect to the running MySQL server on your instance, use the following command:

   ```
   mysql -u root -pdb15319root song_db
   ```

   In the command above, the username is **`root`** and the password is **`db15319root`**. The database that you will be using is **`song_db`**, which has been already created for you within the AMI.

   Before loading our records into the database, we need to describe the database and its fields in a schema, which encapsulates the structure and relationship of the data. For data in **`million_songs_metadata.csv`**, we have provided its schema in Table 1. Then we need to create a table according to the schema. We have provided a **`~/songs/create_tables.sql`** file for you to get started.

   After creating the table (named **`songs`**), you can use the following command to verify the schema of that table:

   ```
   DESCRIBE songs;
   ```

   We now have to load the database using the **`million_songs_metadata.csv`** file. Find a command that lets you do this operation. You can use either a SQL command from within the MySQL CLI client, or use the **`mysqlimport`** utility to load the records from the CSV file into the database. Please note down the command you have used and paste it into the checkpoint below, when asked.

   To verify whether the data has been successfully loaded into the database, you can list the first 10 records of the table using the **SELECT** command:

```
SELECT * FROM songs LIMIT 10;
```

Note: The MySQL installed in the AMI is 5.5. Make sure you search through the documentation for the correct version when looking for command references.

2. The equivalent SQL command of the Beatles awk command from the previous part is:

```
SELECT * FROM songs WHERE artist_name LIKE '%The Beatles%';
```

And similarly the Michael Jackson query looks like:

```
SELECT * FROM songs WHERE artist_name LIKE '%michael jackson%' AND year >=
1980 AND year < 1990 ;
```

MySQL will output the number of records returned, as well as the time for each query. If you only want the number of matches, use **COUNT(*)** instead of *****.

3. In the following steps, we are going to compare the MySQL query response times before creating the index with after creating the index.

```
SELECT AVG(duration) FROM songs;
SELECT * FROM songs WHERE duration=(SELECT MAX(duration) FROM songs);
SELECT * FROM songs ORDER BY duration desc limit 5;
SELECT * FROM songs WHERE duration=(SELECT min(duration) FROM songs);
SELECT * FROM songs ORDER BY duration asc limit 5;
SELECT COUNT(*) FROM songs WHERE duration > (SELECT AVG(duration) FROM
songs) ;
SELECT COUNT(*) FROM songs WHERE duration <= (SELECT AVG(duration) FROM
songs) ;
```

4. Create indices for the table. We provided a file **~/songs/create_indices.sql** which you can run. This might take a while. After that has finished, exit MySQL and restart it:

```
sudo service mysql restart
```

Then relaunch mysql client, and re-run the SQL queries from Part #7, noting the new times ("after index") for the quiz.

5. Sometimes SQL queries are more complicated than their grep/awk counterparts. Write SQL queries that will return the same results as both grep commands from part #1 in flat files above. Note each query for the quiz. (Hint: The SQL **LIKE** operator is case insensitive by default).
6. Remember to Terminate the Instance when you are done. Complete the following checkpoint quiz when ready:

Useful References:

- GNU Reference for AWK
- UNIX man page for grep
- MySQL 5.5 Reference

checkpoint

File vs. Database