ResearchCoders' For Programmers Series
Grep versus FlatSQL versus MySQL: Queries using UNIX tools vs. a DBMS
V.1

Mohammed Q. Hussain
mqh@reseachcoders.dev

December 2018

This document is one of "For Programmers" series, a part of ResearchCoders project. It explains the ideas of [1] for programmers to help them implement them. Please visit our website for more information: http://www.reseachcoders.dev

1 Introduction

This work [1] presents *FlatSQL* which is an SQL-like language that intended to be used with text files (flat files). In this way, the user get user interface that is similar to database management systems (DBMS) through FlatSQL while the advantages of text files remains, such as the simple and common format, and the diversity of the tools that work with text files.

2 FlatSQL

The syntax of FlatSQL is similar to SQL with some differences. The queries in FlatSQL can be in two formats. The first format is:

```
SELECT [select_clause]
FROM [from_clause]
```

While the second format is:

```
SELECT [select_clause]
FROM [from_clause]
WHERE [where_clause]
```

The files are considered tables in FlatSQL, and the content of a file determines the available columns of the table (file), by default, the separator between the columns of a file is any whitespace, but the user is free to change that as we will see later. The columns of a table can be referred to by their order number, or the user can give them an alias. This way of referring is valid in both select and where clauses.

2.1 Select Clause

Given that the user is reading from two files *foo* and *bar*, the following select clause is valid:

```
SELECT a, foo.b, bar.5
```

This clause is going to show the data of column *a* which is an alias for some column in *foo* or *bar* (or even third unmentioned file), then the data of column *b* of table *foo* will be shown, and finally the data of the fifth column of *bar* table is shown.

Also, the user has the ability to determine the separator between the columns when the output of the query shows. The default separator is new line, but the user can use the keyword ***delim*** to change this behaviour. For example, the following select clause shows the columns separated by a colon:

```
SELECT a, foo.b, bar.5 delim ":"
```

2.2 From Clause

As we have mentioned previously, the files are the tables in FlatSQL, so the name of tables in from clause should be filenames. As in SQL, multiple tables can be mentioned in from clause. By default, the separator between file's columns is any whitespace, but the user is free to change this behaviour by using the keyword ***with*** which accepts regular expressions between slashes as an input. The following example changes the separator to colon and uses the famous file of Linux systems */etc/passwd* which has a well-known format.

```
FROM /etc/passwd with /:/
```

Also, aliases can be introduced for both tables and the columns of the table after the keyword ***as***. The following from clause gets the data from three files data.csv, foo and /etc/group. The columns of data.csv are separated by a comma and *d* is used as alias for this file. The file foo has no alias, but two aliases for its columns are defined, the alias of the first column of foo is bar and the second is qux. Finally, the file /etc/group [1] got the alias *g*, its columns are separated by colon, and its first four columns got the aliases group, c, gid and memb. So, in the select clause we can write something like *d*.1, *foo.bar* and *g.gid*.

```
FROM    data.csv with "," as d,
        foo as (bar qux),
        /etc/group with ":" as g (group x gid memb)
```

[1] Which is a well-known file in Linux systems.

2.3 Where Clause

The columns in where clause can be referred in the same way in select clause. Either by their order number, or by a given alias. In FlatSQL only conjunction (AND operator) is can be used between the different conditions in where clause. The available comparison operators are:

Operator	Meaning
=	Equals
<>	Doesn't Equal
<, <=, >, >=	Less that, less than or equal, greater than, greater than or equal
like, not like	As in SQL
rlike, not rlike	Regular Expressions matching

References

[1] Konrad Lorincz, Kevin Redwine, and Jesse Tov. Grep versus flatsql versus mysql, 2003.