MySQL Command Line Tool: Unleash the Power of mycli

</> thevaluable.dev/mysql-command-line-tool-mycli

29 listopada 2018

29 Nov 2018 , updated 1 Jun 2019

#Tools #Mouseless

I had a mission, for many years.

This quest led me to do innumerable Google searches, trying new tools to finally give up each time. I was feeling like a knight fighting the same dragon again and again, and the dragon was always winning. Where was the happy ending?

As Perceval wanted to find the Holy Grail, I wanted to find a good interface for MySQL. I was using MySQL Workbench, like everybody. I wasn't satisfied, almost like everybody. The software was slow and buggy, the auto completion was working totally randomly. I was searching something similar but simpler and more reliable.

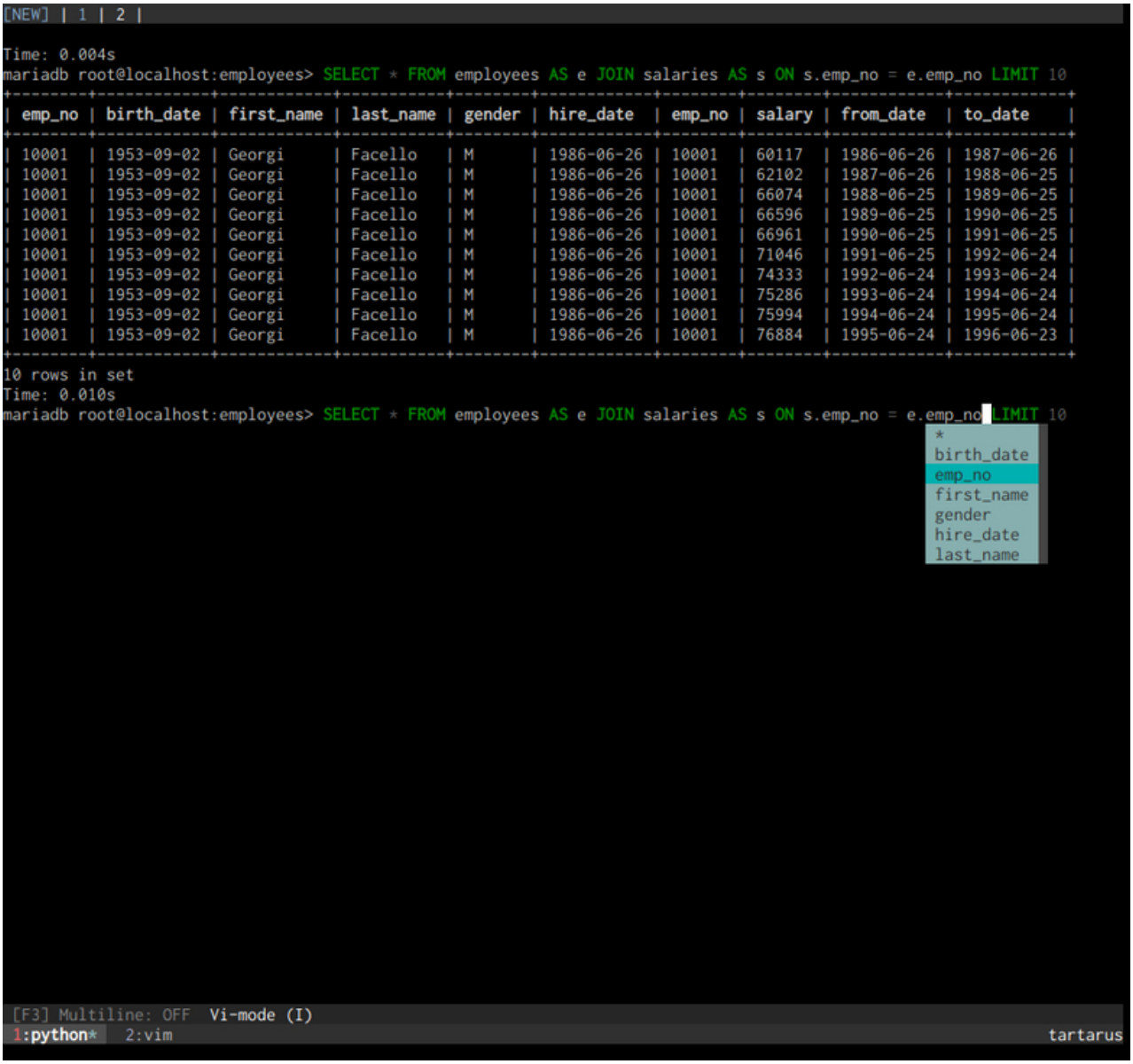
I was young, full of innocence. I was lost in the GUI world, I was searching something with button to click and pop up to close.

How fool I was! When former colleagues showed me the path to Vim and ultimately to the benediction of a full terminal oriented system, it became clear I wanted to use a powerful CLI for MySQL. I tried the one shipped with MySQL of course, but I found it too limited.

Then, in the corner of the Internet, where all hope were gone, I saw the Light.

I found mycli.

The Best MySQL Tool?



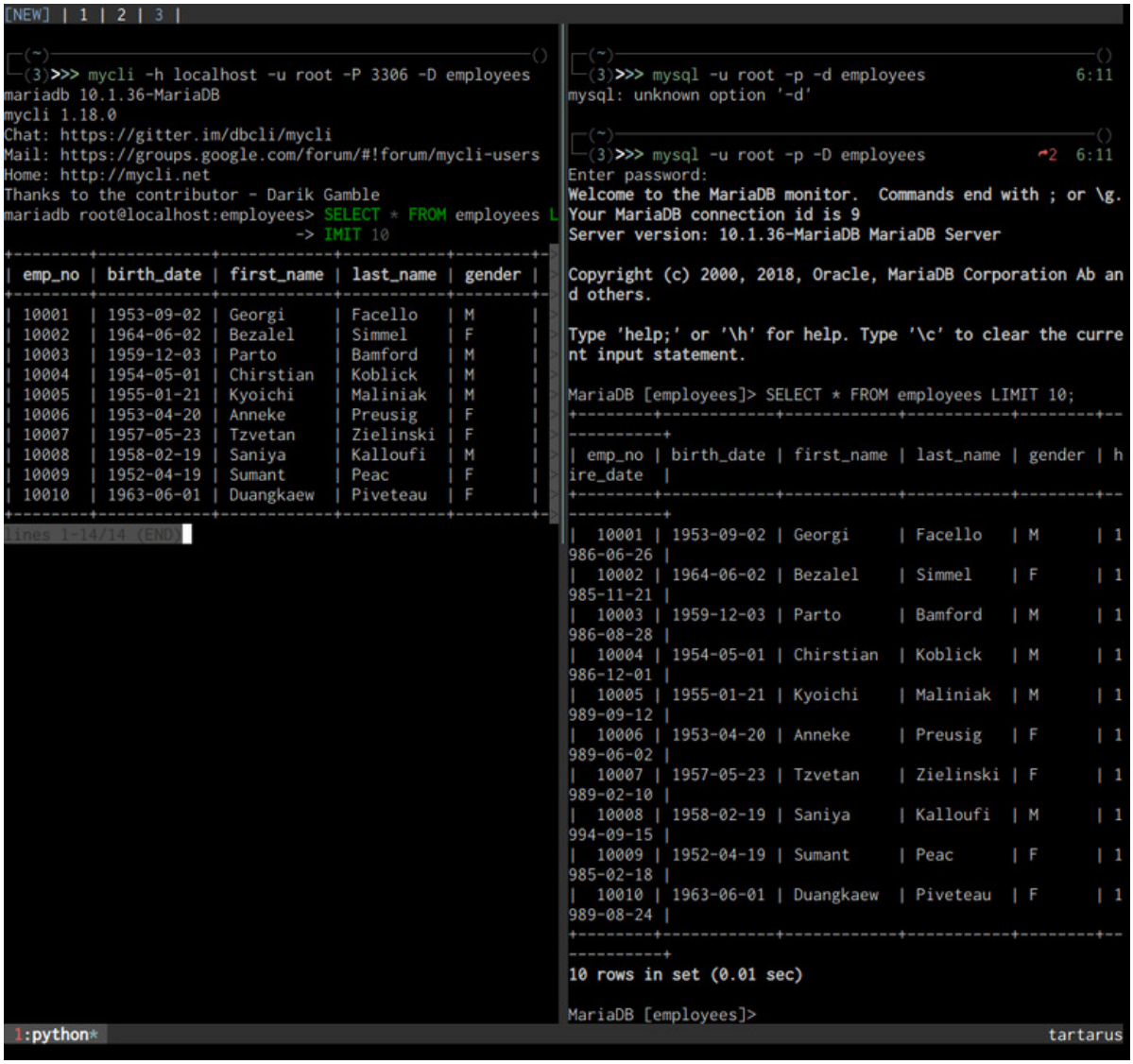
Let's understand each other: mycli is not perfect, but it's still better than anything I tried so far.

mycli vs MySQL CLI

mycli has a lot of useful features the good old MySQL CLI doesn't have. Here some examples:

- mycli provide a smart auto completion. It can even auto complete aliases. This feature alone led me to praise mycli till the end of time.
- You don't need to end each query with `;` (semicolon). It sounds silly but it's really a pain to me with MySQL CLI.

- You can save your favorite queries in mycli's config file via a handy snippet system.



On the right mycli, on the left MySQL CLI

mycli vs MySQL Workbench

At last, I've got rid of the unstable beast which is MySQL Workbench!

Why so much hate?

- mycli is stable. MySQL Workbench is not.
- MySQL Workbench has an attractive interface but mycli is way faster to use. It's the eternal discussion GUI vs TUI. No hesitation for me: I take the shell.
- I don't see anything MySQL Workbench can do that mycli can't. If you see some functionality missing in mycli, I would be curious to know it. The comment section is (surprisingly!) at the end of this article.
- mycli can format the output of your query in many different formats.
- MySQL Workbench can write for your some painful queries to remember (adding foreign keys / indexes for example). However, you can use snippets in mycli for the same benefit and result.

- The config file of mycli can be stored on Github (it's part of my dotfiles) or on a private cloud like Nextcloud. This is highly practical to have everywhere the exact same configuration for mycli (and everything else like Vim)!

mycli Installation and Database Connection

Since mycli is a command line tool, every command described here will have to be typed in a terminal.

Installing mycli

You can install mycli like any other software. The only requirement is Python 2.7 or 3.4+.

- Arch Linux (using the AUR with yay): `yay -S mycli`
- Debian / Ubuntu: `sudo apt-get install mycli`
- macOS: `brew install mycli`
- Fedora: `sudo dnf install mycli`
- Everywhere, using the Python package: `sudo pip install mycli`

I personally use Arch Linux and I install it via the AUR. It allows me to update easily mycli along with every application installed, thanks to yay and the command `yay -Syu`.

mycli Config file

Note that mycli use a config file which is usually located:

- On Linux: `~/.myclirc`
- On Windows: `C:\Users\<username>\.myclirc`

If you don't find the file, you can still use the command line `find $HOME -name .myclirc` on Linux to locate it easily. If it doesn't work, try to replace `$HOME` with your root folder `\`.

I will come back to this configuration file later.

Connection to MySQL databases

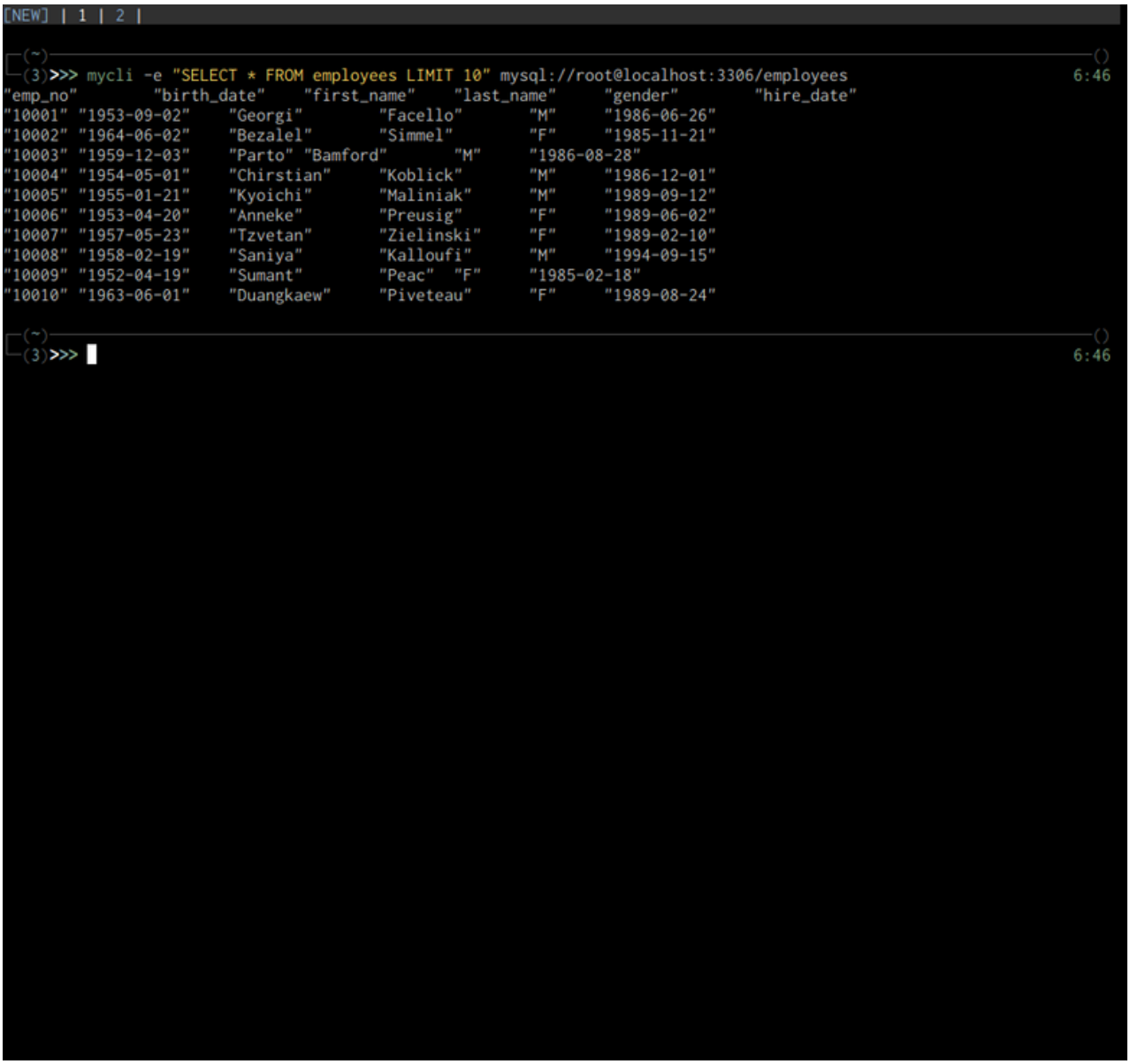
Local connection

Here are two examples which will connect mycli to a database `employees` hosted on `localhost` with the user `root` using the port `3306`:

- `mycli -h localhost -u root -D employees -P 3306`
- `mycli mysql://root@localhost:3306/employees`

You can even execute queries while staying in your comfy shell by typing:

```
mycli -e "SELECT * FROM employees LIMIT 10"
mysql://root@localhost:3306/employees
```



As you can see, the output format is not the prettiest. You can add the option `-t` to display a table.

Commands to Connect Quickly to MySQL Databases

Instead of typing each time the connection credentials to connect to a database, you can create DSN aliases.

To do so, you need to modify your config file `.mycli` under the section `[alias_dsn]`.

For example:

```
[alias_dsn]
employees = mysql://root@localhost:3306/employees
```

Then you just need to type `mycli -d employees` to connect to your `employees` database.

The command `mycli --list-dsn` will display the list of DSN aliases you configured.

Connecting to Remote Database via SSH

It's really easy to connect on a remote database via SSH using openssl.

First, you need to open a tunnel to your remote server: `ssh -Nf <user>@<host> -L 3310:localhost:3306`

Then you can connect to your database simply by using: `mycli -h localhost -u <database> -P 3310`

Here's a concrete example:

1. `ssh -Nf cooluser@192.168.0.1 -L 3310:localhost:3306`
2. `mycli -h localhost -u cooluser -D employees -P 3310`
3. Enter the password for the user `cooluser`
4. You're connected to the database `employees` !

Obviously the user `cooluser` in our example needs to have permissions on the database `employees` .

Basic Usages and Commands

Congratulation! You should now be connected to your database with mycli! As you can see the prompt will display the username, the host and the database you are in.

You can of course execute queries from there. Try it out and see by yourself the amazing auto completion mycli offers by typing something like: `SELECT * FROM employees WHERE id = 1`

Normal SQL stuff. Notice that you don't need to end your query with the annoying semicolon.

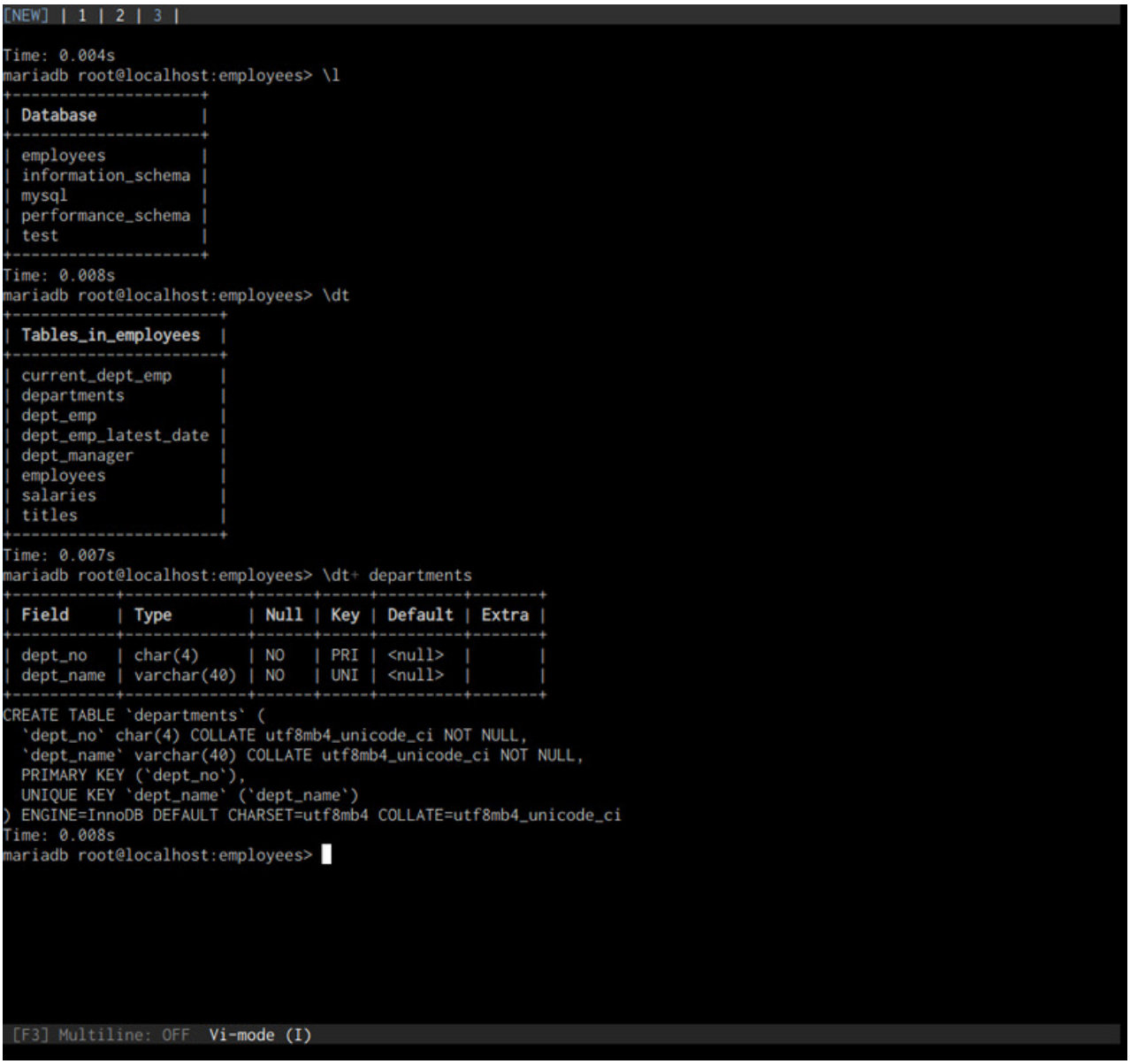
You can as well execute some special commands, all beginning with `\` .

Listing and Using Databases

- `\l` - List your databases.
- `\u` - Use a different database.

Displaying the List of Tables

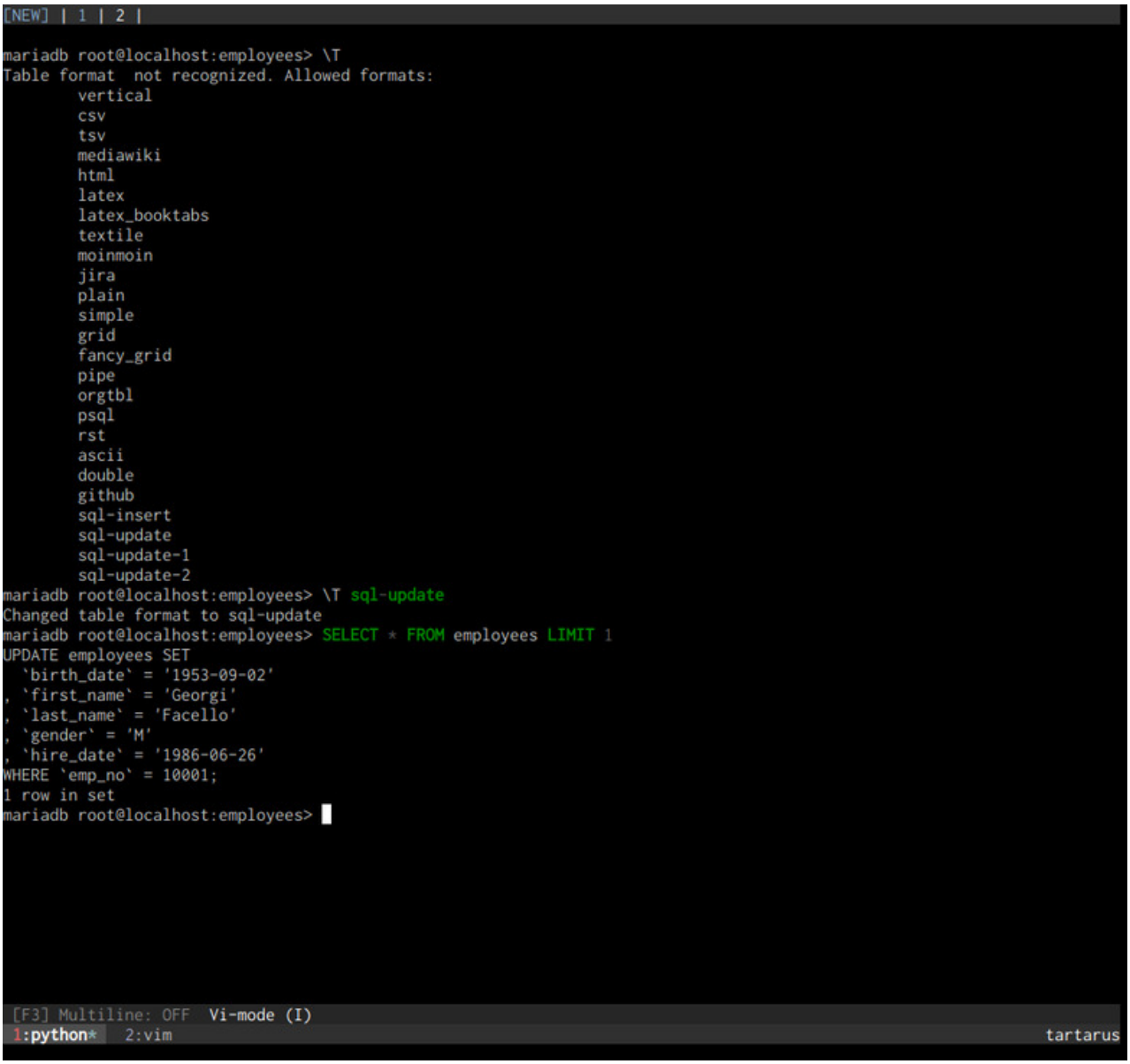
- `\dt` - List tables in the current database.
- `\dt+` - Show the created statement used to create the table. Useful to display the structure of the table.



Output Formatting

One of the most useful functionality of mycli is the ability to output data in different format:

- `<sql_query> \G` - Display the query result vertically instead of horizontally. I find it very useful: scrolling horizontally is always a bit of a pain.
- `\T <format>` - Change the output with the format of your choice. Type `\T` to list all the format available. The default format used is `psql`.



You have the choice between a lot of output formats. JSON and YAML are missing though…

You can as well modify the default output of every result from horizontal to vertical only when the results is wider than your shell. You just need to modify the following line in mycli config file:

```
auto_vertical_output = True
```

I would advice to give it a try!

Remember that you can execute queries without using mycli prompt? You can format the output in `csv` as well:

```
mycli --csv -e "SELECT * FROM employees LIMIT 10"
mysql://root@localhost:3306/employees
```

This can be handy if you only want to output quickly some data.

Query Snippets: your Favorite Queries

mycli has a system of favorite queries, basically snippets you can use with placeholders.

Here are the commands associated with these favorite queries:

- `\f` - List all favorite queries
- `\f <name>` - Invoke a specific favorite query
- `\fs <name> <query>` - Save a favorite query
- `\fd <name>` - Delete a favorite query

When you create a favorite query, mycli will write it in your mycli config file, under the section `[favorite_queries]`.

Here are the ones I have in mine:

```
# ADD INDEX
# $1 table name
# $2 index name
ikey = '''ALTER TABLE `$1`
ADD INDEX `IDX_$1_$2` (`$2` ASC)'''

# ADD FOREIGN KEY
# $1 table name
# $2 index name
# $3 reference table name name
fkey = '''ALTER TABLE `$1`
ADD CONSTRAINT `FK_$1_$2`
  FOREIGN KEY (`$2`)
  REFERENCES `$3` (`$2`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION'''
```

For example, if I type `\f ikey salaries to_date`, this query will be executed:

```
ALTER TABLE `salaries` ADD INDEX `IDX_salaries_to_date` (`to_date` ASC)
```

In other words, it will create an index on the column `to_date` of the table `salaries`. Super handy!

Working with Files

The System Command

First thing first, if you want to know what files are in the folder you are, directly from mycli, you can use the command:

```
system ls
```

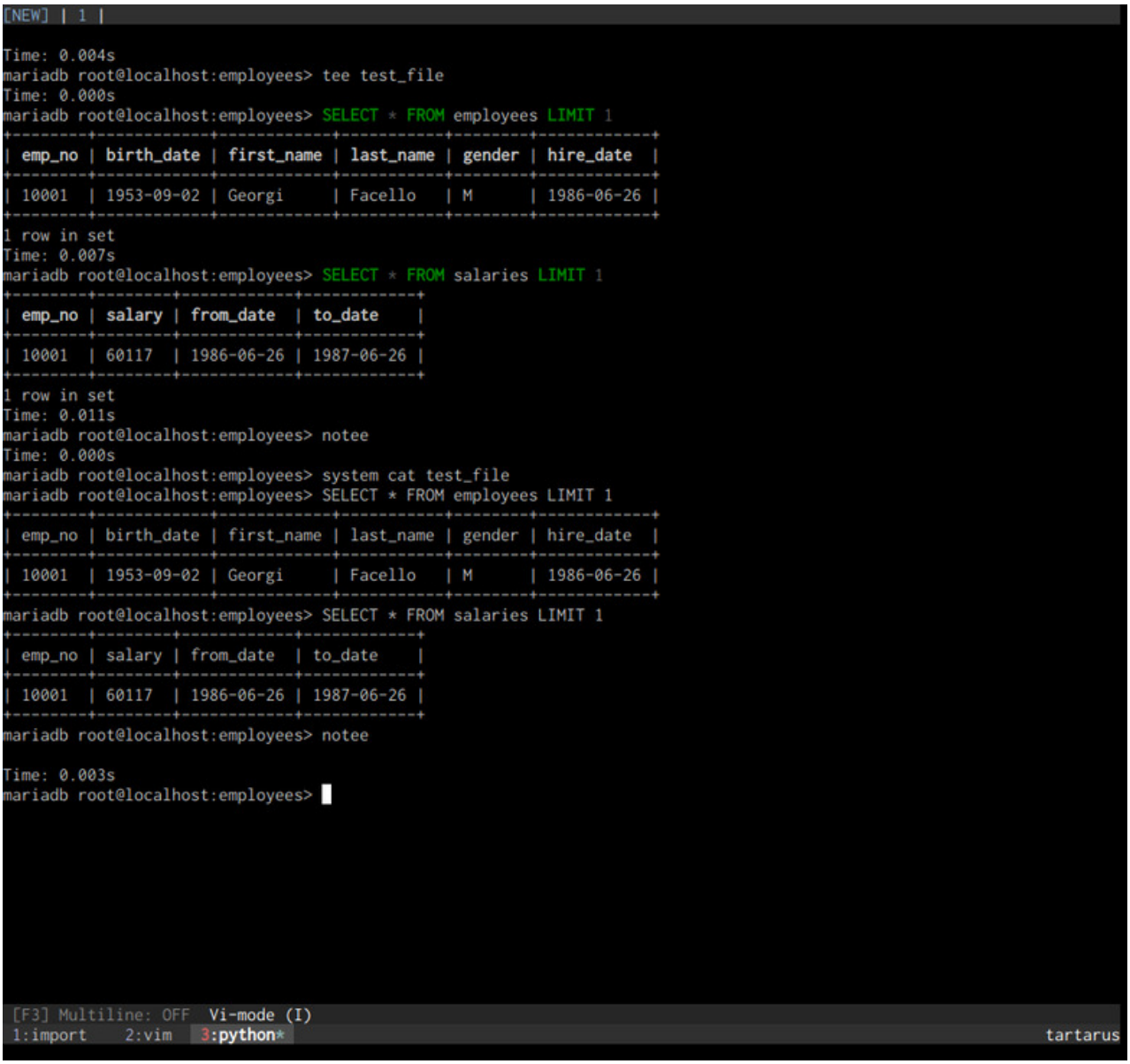
The command `system` can be used with any other shell command. It can be very useful if you need to do some `cat` or `grep` on sql files for example. If you want to clear your terminal, just type `system clear`.

Importing a SQL file

If you need to execute a SQL script, you can simply use the command `\. filename`.

Output To a File

- `tee [-o] filename` - output everything into a file. Queries, results, everything displayed on your terminal will be written. If the file `filename` doesn't exist, it will be created. To overwrite an already existing file, you can use the `-o` option.
- `notee` - stop writing the output to a file.
- `\o [-o] filename` - output the next result only into a file. Again, the `-o` option is to override a file already existing. However, it seems that this command only work once per session...



MySQL Command Line Happiness: Last Tips

The Auto Completion is Too Smart? Make it Dumb!

Playing around with mycli will show you the handy smart auto completion. If you're not satisfied with the propositions, you can make the auto completion dumb by pressing `F2`.

It will propose you every keyword available whatever the context. If you want to come back to the sweet smart auto completion, press `F2` again.

Refreshing the Auto Completion

The command `\#` will refresh the auto completion in case your new databases / table / alias you just created are not in the result set.

Vim Mode

A convenient Vim mode is available to be able to navigate in and modify your queries. You can enable it by adding / modifying this line in your config file: `key_bindings = vi`.

You have the choice between `vi` or `emacs`

MySQL Command Lines in Vim (or Nano)

You can use the command `\e` at the end of any query to be able to edit it with your favorite editor, defined thanks to both your `$EDITOR` and `$VISUAL` environment variables.

I **love** this functionality. Being able to modify complex query in Vim is a dream coming true.

Backward History Search

Exactly like in your favorite shell, you can search a query you typed before by using the keystroke `CTRL + r`.

Difficult to Live Without mycli

I spend most of my time in a terminal for good reasons: it's fast, powerful and I don't need to always switch between my keyboard and my mouse. mycli is the last addition to my set of tool in order to never ever leave the terminal again.

A last thing: if you're using PostgreSQL instead of MySQL, you can use pgcli from the same author. You can see the complete list of all the tools he created to manage different database system via CLI here.