PostgresPro

Git and vim for PostgreSQL developers

Nikolay Shaplov

PostgresPro

General topic

If you want to write well, you should read a lot. And you'd better read best samples you can find.
PostgreSQL is one of the finest C-driven project one can get.
In this talk we will learn how to use git and vim to read and explore PostgreSQL code right in the console.

PostgresPro

Git

Git, fetching the source code

In order to get latest snapshot of PostgreSQL code you should run command:

```
$ git clone https://git.postgresql.org/git/postgresql.git
```

If that command does not work for you, may be you just need to install ‘git’ package

```
$ sudo apt-get install git
```

PostgresPro

Vim

PostgresPro

Why vim?

- Masterclass is when expert shows how he works. And I work with vim
- We will need complex navigation methods to browse the code, that is not provided by simple editors
- If you work with the console, you are much closer to system level than in modern IDE
- You may consider this as an excursion into history
- Vim is actually a professional tool providing very high speed of coding
- Most students find vim very useful as soon as they learn how to do basic things

PostgresPro

Vim myths

Vim myths

- There are two main modes in vim:
  - Beep
  - Mess everything up

Vim myths

- ~~There are two main modes in vim:~~
  - ~~Beep~~
  - ~~Mess everything up~~

No, this is one mode

Vim myths

- ~~There are two main modes in vim:~~
  - ~~Beep~~
  - ~~Mess everything up~~

No, this is one mode

- You can't quit vim

Vim myths

- ~~There are two main modes in vim:~~
  - ~~Beep~~
  - ~~Mess everything up~~

No, this is one mode

- ~~You can't quit vim~~

No. Vim is not Hotel California.

Vim myths

- ~~There are two main modes in vim:~~
  - ~~Beep~~
  - ~~Mess everything up~~

No, this is one mode

- ~~You can't quit vim~~

No. Vim is not Hotel California.

I may assure you, you will be expert in quitting vim when these talks are finished.

Vim myths

- ~~There are two main modes in vim:~~
  - ~~Beep~~
  - ~~Mess everything up~~

No, this is one mode

- ~~You can't quit vim~~

No. Vim is not Hotel California.

I may assure you, you will be expert in quitting vim when these talks are finished.

- One need MIDI Foot Controller Pedal to properly work with vim

Vim myths

- ~~There are two main modes in vim:~~
  - ~~Beep~~
  - ~~Mess everything up~~

No, this is one mode

- ~~You can't quit vim~~

No. Vim is not Hotel California.

I may assure you, you will be expert in quitting vim when these talks are finished.

- ~~One need MIDI Foot Controller Pedal to properly work with vim~~

No. MIDI Foot Pedal is for Emacs users only

Vim modes

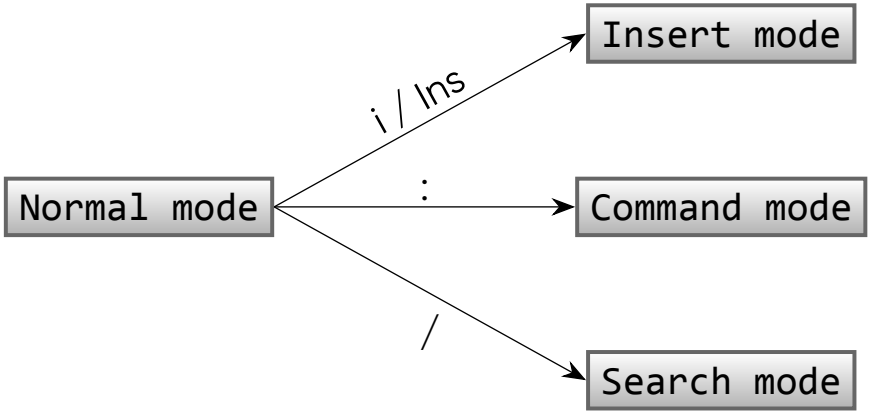
Insert mode

Normal mode

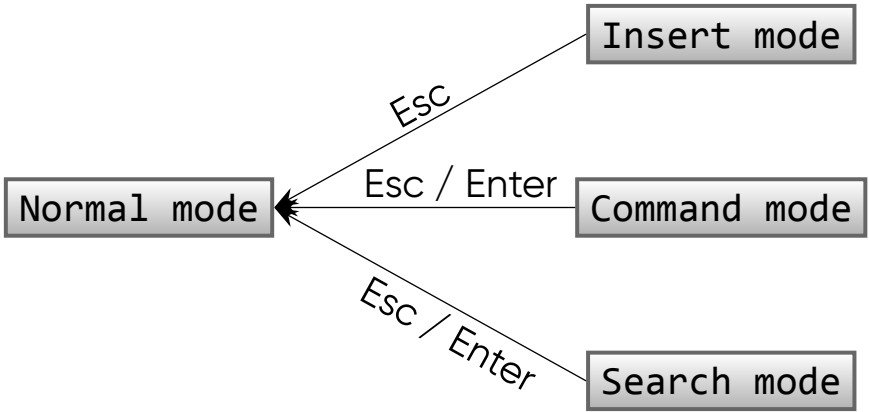
Command mode

Search mode

Vim modes



Vim modes


Insert mode
Esc
Esc / Enter
Normal mode
Command mode
Esc / Enter
Search mode

Commands of the command-line mode

- **:q** — quit vim
- **:q!** — quit without saving
- **:w** — save (**:wq** — save and quit)
- **:123** — go to line 123
- **:set number** — show line numbers
- **:set nonumber** — hide line numbers

Use **up** and **down** arrows to scroll through commands history

Tab is for autocompletion

Normal mode commands

- **dd** — delete line;
- **u** — undo;
- **Ctrl + r** — redo.

Copy and past commands

- **v** / **V** — text selection mode
- **d** — delete selected text into buffer
- **y** — copy (yank) selected text into buffer
- **p** / **P** — insert text from buffer in/after cursor position

Search mode commands

- /**abc** — search text for substring ‘abc’;
- / — search again.

Use **up** and **down** arrows to scroll through search history

You can also use regular expressions here.

PostgresPro

Navigating through code using tags in vim

Tags lists

Tags are an ancient way for navigating the code, for jumping from symbol's usage to it's definition.

In this technique you use some script for generating list of symbols defined in your project (tags) and your editor uses them for navigating through the code.

For C code of PostgreSQL tags are generated with 'ctags' program from 'exuberant-ctags' package.

Building tags list

To build tags list you should install 'exuberant-ctags' package:

```
$ sudo apt-get install exuberant-ctags
```

And then run `make_ctags` script provided by PostgreSQL developers from the root of PostgreSQL source tree

```
$ src/tools/make_ctags
```

You will get a bunch of `tags` files in each directory of source tree. You can copy it to your extension dir, so you can jump to postgres code function definitions while editing your code.

Using tags in vim

Let's open some file from PostgreSQL source code

```
$ vim src/backend/postmaster/postmaster.c
```

and find some symbol that most probably defined inside PosgreSQL code. I found `BACKEND_NUM_TYPES` on 145th line.

You can move cursor to that symbol, and press **Ctrl +]** to jump to the place where this symbol is defined.

In order to go back, press **Ctrl + t**

Hiding `tags` from git

`tags` files are not ignored by by git version control system. You can see that by running command

```
$ git status
```

Theoretically you cat add `tags` into `.girignore` file, but you will have to track this change manually, make sure it is not committed, and so on. Less known solution is to add `tags` into local `.git/info/exclude` file, that will do same trick.
Run again

```
$ git status
```

and you will see that `tags` files are now ignored.

Open file on certain position

If you, for some reason need to open file on certain position, for example you get an error message from a compiler, that specify certain line number, you can specify this line number as an extra option with "plus"prefix in front of it:

```
$ vim src/backend/access/common/reloptions.c +1234
```

This command will open `reloptions.c` file in vim, and move cursor to line number 1234.

greping for function definition

If you know function name (from compiler error message) and would like to find it's definition, most probably you will try to use **grep** utility. But you will not success, as multiple **tags** files will also contain this function name. You should use 'git grep' command instead, as it will ignore files that are not part of git repository

```
$ git grep ^tsvectorin
```

This command uses the convention that, in a C program, the function name is specified starting from the first character of the line. This allow to find only function definition, and ignore all other occurrences.

```
int
my_function(char * str, bool is_cool)
```

PostgresPro

Спасибо за внимание!