Table of Contents

SQL*Plus is a command line interface that provide access to your data via SQL statements. The basic SQL*Plus features were described in a previous document. This document is included for people who want to know how to use the SQL*Plus line editor. **It is not intuitive and you do not need to use it for class.**

1. The SQL buffer

SQL*Plus maintains a buffer called the SQL buffer.

- The SQL buffer contains the most recent SQL statement that you entered.
- SQL*Plus commands do not get stored in the SQL buffer.
- Executing the SQL statement does not empty the SQL buffer.
- You can re-execute the command in the SQL buffer with the / command.
- You can edit the SQL statement in the SQL buffer

2. Editing the SQL buffer using a text editor

You can give the edit command which loads the contents of the SQL buffer into your designated editor. The editor on the Unix system might be pico or vi. When you exit the editor, you load the file back into the SQL buffer. The default name of the file is afiedt.buf; if you are using this file simply to edit the **buffer**, you can keep that filename. You might see that name in your file system.

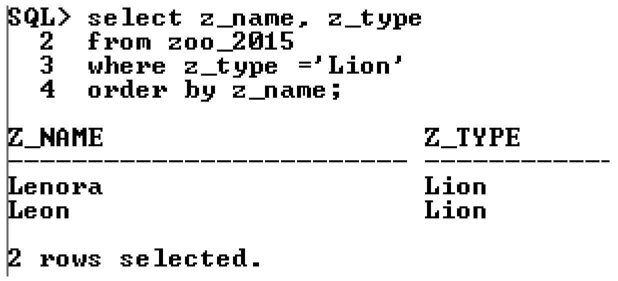
3. Editing the SQL buffer using the Line Editor

For small changes you might find the SQL*Plus line editor useful.

This first thing to be aware of is that this is a line editor and changes are made to the current line only. This is not the way we are used to in the windows world- or in most visual editors. As you add and delete lines from the SQL buffer, the lines get renumbered. The line numbers are supplied automatically and are not part of your SQL statement. Use the List command to redisplay the buffer. The current line is indicated by a * after the line number.

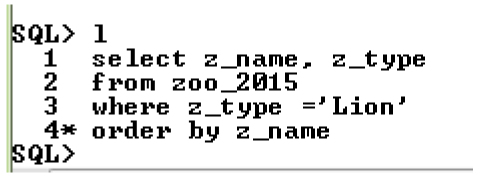
3.1. Walkthrough

First let's go through a simple walkthrough. Using the zoo_2015 table, you enter and execute the following sql statement.



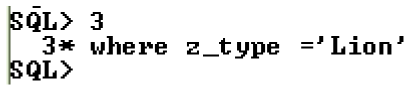
```
SQL> select z_name, z_type
  2  from zoo_2015
  3  where z_type ='Lion'
  4  order by z_name;

Z_NAME                     Z_TYPE
-------------------------- ------------
Lenora                     Lion
Leon                       Lion

2 rows selected.
```

You can now give the List command (abbreviation l) and see the content of the buffer



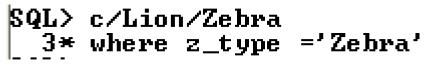
```
SQL> l
  1  select z_name, z_type
  2  from zoo_2015
  3  where z_type ='Lion'
  4* order by z_name
SQL>
```

Suppose I want to change the SQL statement in the buffer and select for zebras instead of lions. I need to edit line 3. I type a 3 and line 3 becomes the current line.



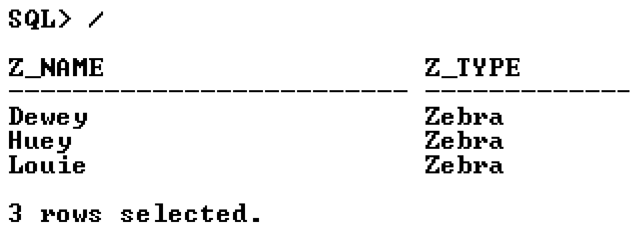
```
SQL> 3
  3* where z_type ='Lion'
SQL>
```

The following shows a change command. The / characters are used as delimiters and the command says to change the first occurrence of the pattern Lion to Zebra on the current line. We get feedback on the change.



```
SQL> c/Lion/Zebra
  3* where z_type ='Zebra'
```

Now use the slash command and we find the zebras.

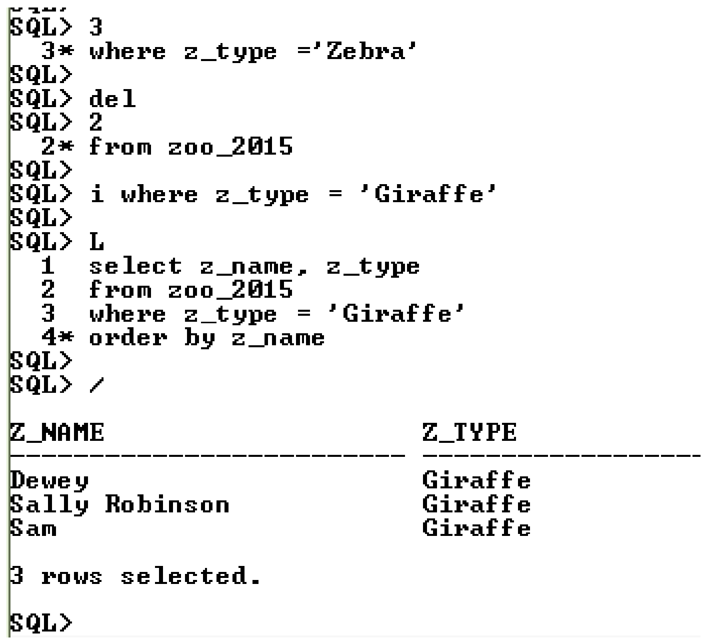


```
SQL> /

Z_NAME                    Z_TYPE
------------------------- -------------
Dewey                     Zebra
Huey                      Zebra
Louie                     Zebra

3 rows selected.
```

If you like this, you must be a Unix person at heart. Try a few other changes.

The del command deletes the current line

The i command lets you insert a new line after the current line



```
SQL> 3
  3* where z_type ='Zebra'
SQL>
SQL> del
SQL> 2
  2* from zoo_2015
SQL>
SQL> i where z_type = 'Giraffe'
SQL>
SQL> L
  1  select z_name, z_type
  2  from zoo_2015
  3  where z_type = 'Giraffe'
  4* order by z_name
SQL>
SQL> /

Z_NAME                    Z_TYPE
------------------------- ------------------
Dewey                     Giraffe
Sally Robinson            Giraffe
Sam                       Giraffe

3 rows selected.

SQL>
```

3.2. Rules for the SQL*Plus line editor

Remember this is an editor for the SQL*Plus client. If you are using another client, this does not apply.

The SQL*Plus line editor is editing the SQL buffer— it is only doing an edit of the last SQL query you entered and does not directly provide the ability to save this as a file.

You cannot go back up though a command history.

The semicolon (;) that you use to run a query is not stored in the buffer. Do not add a semicolon to the last line when you edit the buffer. After you have edited the buffer, you can list it with the List command and run it with either the slash command or the run command.

Below is a table of the SQL*Plus Line Editor commands. "m" and "n" refer to line numbers in the buffer; the asterisk (*) is used to indicate the current line; "text", "old" and "new" are text strings.

Errors often make the line with the error the current line. Use the list command and then select the line you want for the current line.

Editing Command	Abbrv	Function
`RUN`	`R`	display and execute the SQL statement in the buffer
`CLEAR BUFFER`	`CL BUFF`	delete all lines from the buffer
`LIST`	`L`	list all lines in the buffer; this also moves the current line pointer to the last line in the buffer
`LIST n`	`L n`	list line n in the buffer and makes that line the current line
`LIST m n`	`L m n`	list lines m through n in the buffer
`LIST *`	`L *`	list the current line in the buffer
`LIST LAST`	`L LAST`	display the last line and make that the current line
`LIST n`	`L n`	display line n through the last line and make that the current line
`n`		list line n and make it the current line; this does not work for a range of lines
`n text`		replaces line n with the indicated text to insert a line before the first line use 0 text

`CHANGE /old/new/`	`C/old/new/`
	
	`C/old/new`
		change text on the current line. The separator character can be anything except a letter or digit or space. Generally, you do not need the last separator character.
		Change affects only the first occurrence of that text in the current line.
		Change is not case-specific for the old text.
		In the change command, you can use ellipses in the old pattern.
		C/…old/new/ changes everything on the current line up to and including old with new.
		C/old…/new/ changes everything from old to the end of the line.
		C/old1…old2/new/ changes everything from old1 to the next occurrence of old2.

`CHANGE /text/`	`C/text/`	remove the indicated text from the current line
`APPEND text`	`A text`	append the indicated text to the current line; don't forget any blanks that you need; use one space after Append before your text.
`DEL`		delete the current line
`DEL n`		delete line n
`DEL m n`		delete lines m through n

`DEL  LAST`		delete the last line
`INPUT`	`I`	insert a series of line **after** the current line; terminate input with a blank line
`INPUT  text`	`I  text`	insert the text as a line; use this to insert a single line.