# Executing Complex Statements

If you need to execute a complex statement that itself contains other statements in the SQL Commander, such as a CREATE PROCEDURE statement, you need to help DbVisualizer figure out where the complex statement starts and ends. The reason is that DbVisualizer needs to send statements to the database for execution one by one.

We recommend that you use the Create Procedure dialog and Procedure Editor to work with procedures and similar objects, so only use the SQL Commander to run statements like this if these features do not work for you for some reason.


- Using Execute Buffer
- Using an SQL Block
- Using the @delimiter command


## Using Execute Buffer

The **SQL->Execute Buffer** operation sends the complete editor buffer for execution as one statement. No comments are removed and no parsing of individual statements based on any delimiters is made. You can use this feature if the complex statement is the only statement in the SQL Commander editor.

## Using an SQL Block

To tell DbVisualizer that a part of a script should be handled as a single statement, you can insert an SQL block begin identifier just before the block and an end identifier after the block. The delimiter must be the only text on the line. The default value for the **Begin Identifier** is `--/` and for the **End Identifier** i t is `/`.

Here is an example of an SQL block for Oracle:

```
--/
script to disable foreign keys

declare cursor tabs is select table_name, constraint_name
  from user_constraints where constraint_type = 'R' and owner = user;

begin
  for j in tabs loop
    execute immediate ('alter table '||j.table_name||' disable constraint'||j.constraint_name);
end loop;
end;
/
```

## Using the @delimiter command

With the **@delimiter** command you can temporarily change the statement delimiter DbVisualizer uses to separate the statements and send them one by one to the database. Use it before the complex statement, and after the statement if the script contains additional statements. Here's an example:

```
@delimiter ++;
CREATE OR REPLACE FUNCTION HELLO (p1 IN VARCHAR2) RETURN VARCHAR2
AS
BEGIN
 RETURN 'Hello ' || p1;
END;
++
@delimiter ;++
@call ${returnValue||(null)||String||noshow dir=out}$ = HELLO('World');
@echo returnValue = ${returnValue}$;
```

The first @delimiter command sets the delimiter to ++ so that the default `;` delimiter can be used within the function body in the CREATE statement. The ++ delimiter is then used to end the CREATE statement, and another **@delimiter** command sets the delimiter back to `;` for the remaining commands in the script.