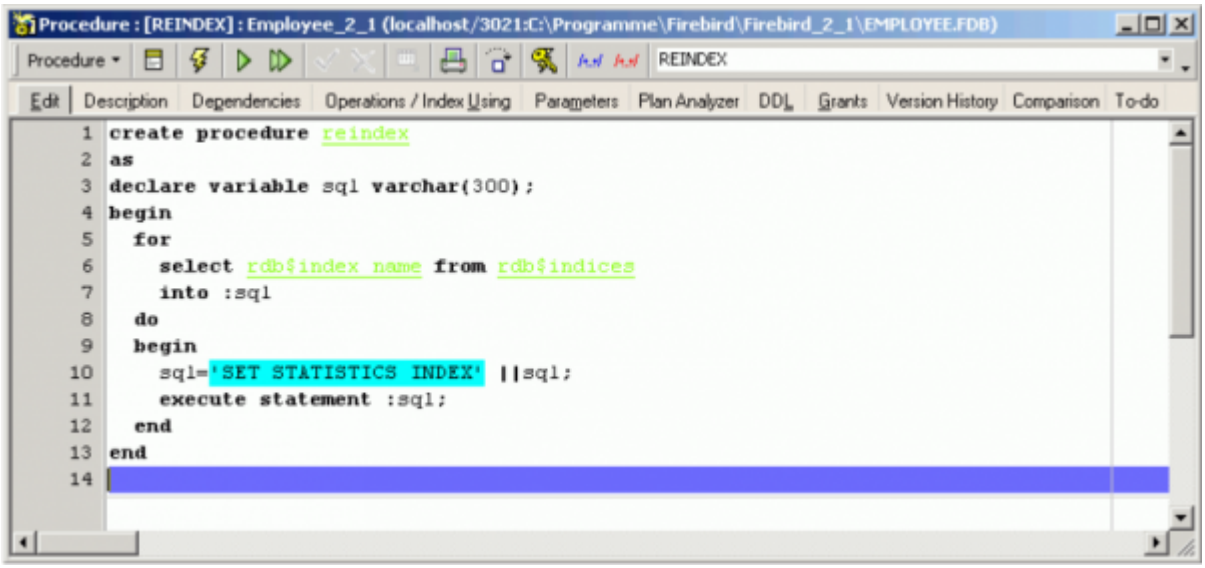
Firebird 2.0 Blocks

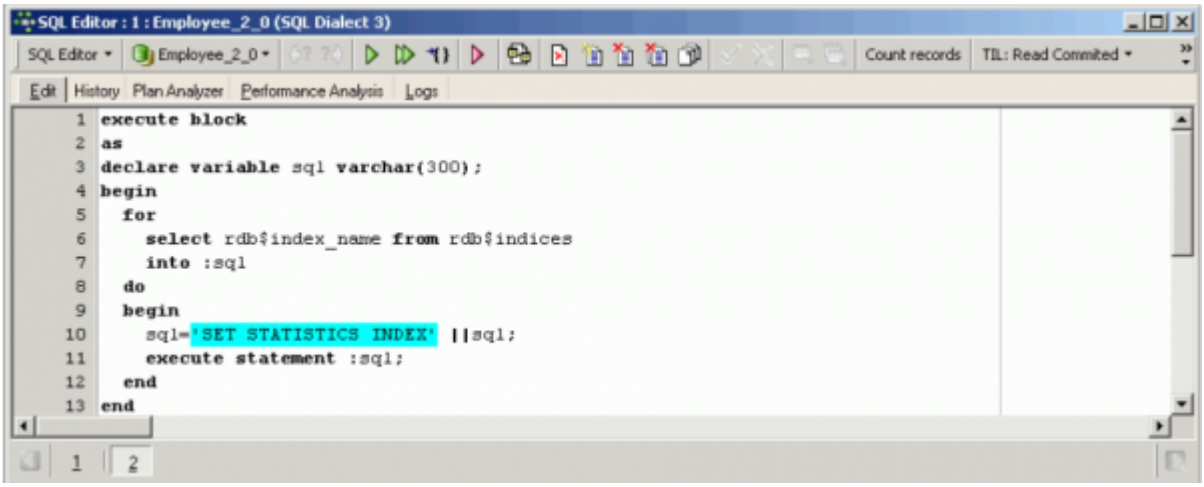
New to Firebird 2.0, Firebird's block implementation enables complex SQL operations in many application areas.

A block is a simple feature, using the new EXECUTE BLOCK syntax, which executes a block of PSQL code as if it were a stored procedure, optionally with input and output parameters and variable declarations. This allows the user to perform “on the fly” PSQL within a DSQL context. It performs a block of instructions on the server side, and can in fact be considered a virtual stored procedure.

To illustrate this, let's consider the following situation: you have a procedure, but you don't really want or need to store it in your database.:



You just want to create such a procedure on the fly and drop it afterwards. So make the following simple alterations:



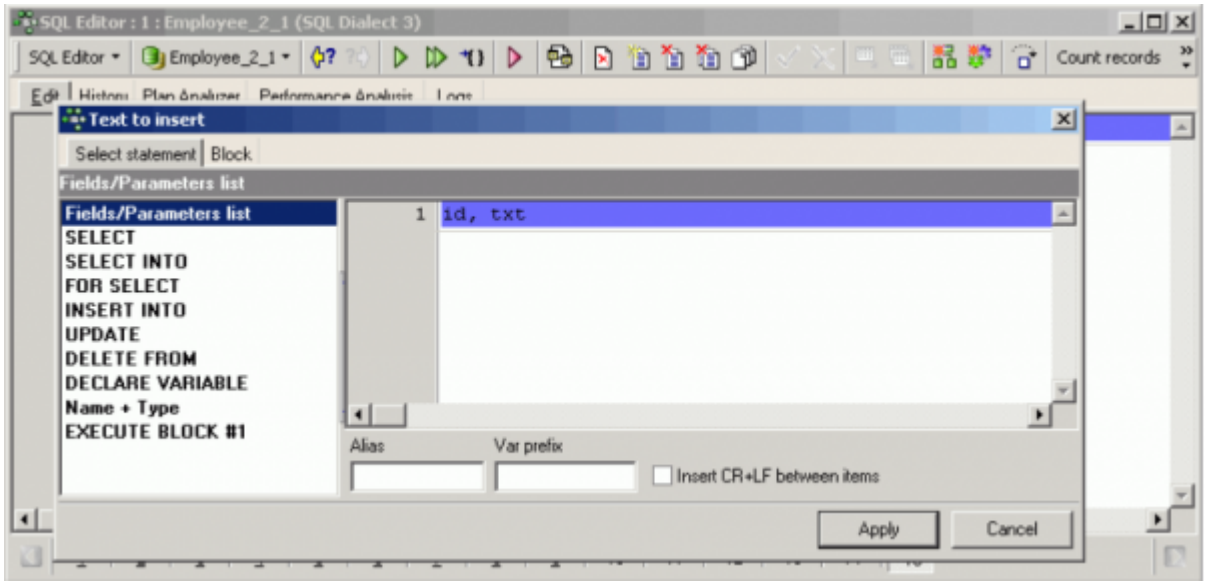
and it performs the same task, but as a dynamic block and not as a stored procedure.

The block transfers the source code from the client to the server, and executes it at the same speed as a stored procedure. The block is created and prepared when you start it, and deleted when you commit or roll back. The server will never use it again.

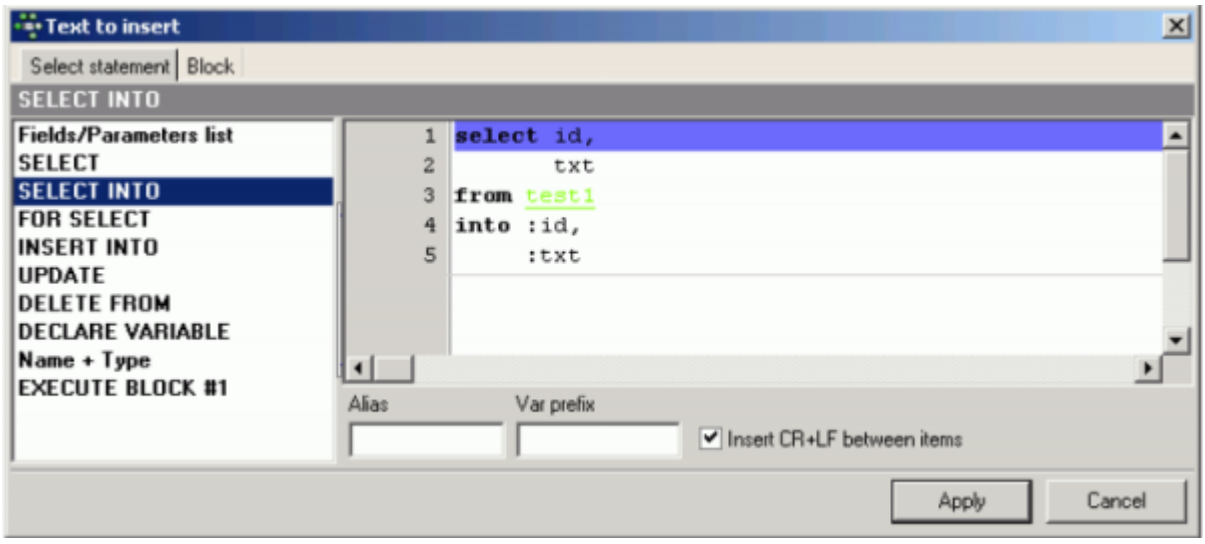
The major advantage of a block is when you are creating a variety of different but similar procedures from your client application, for example you have stored procedures for customer searching; in one stored procedure you are doing the customer search for the sales department, and in the other stored procedure you are doing the customer search for the invoice department. They have slightly different search criteria and want to see different columns in the result sets – this could be an interesting task, as the number of columns can be directly and dynamically created in a block.

EXECUTE BLOCK is not only a alternative to stored procedures; there are other uses, particularly for performance tasks.

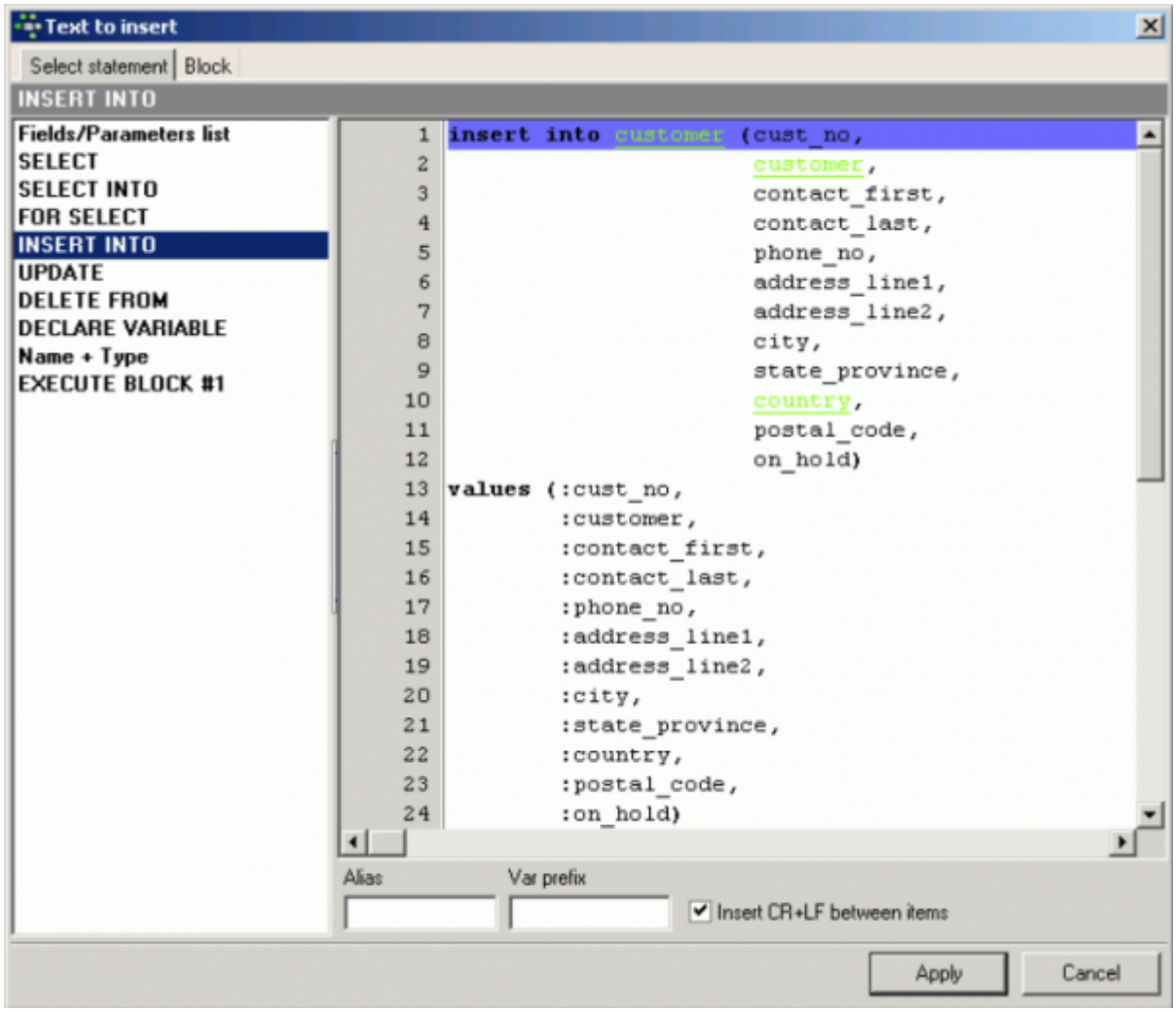
To illustrate this, for example, take a table TEST1, drag it from the DB Explorer into the SQL Editor. The Text to insert window opens offering a range of options:



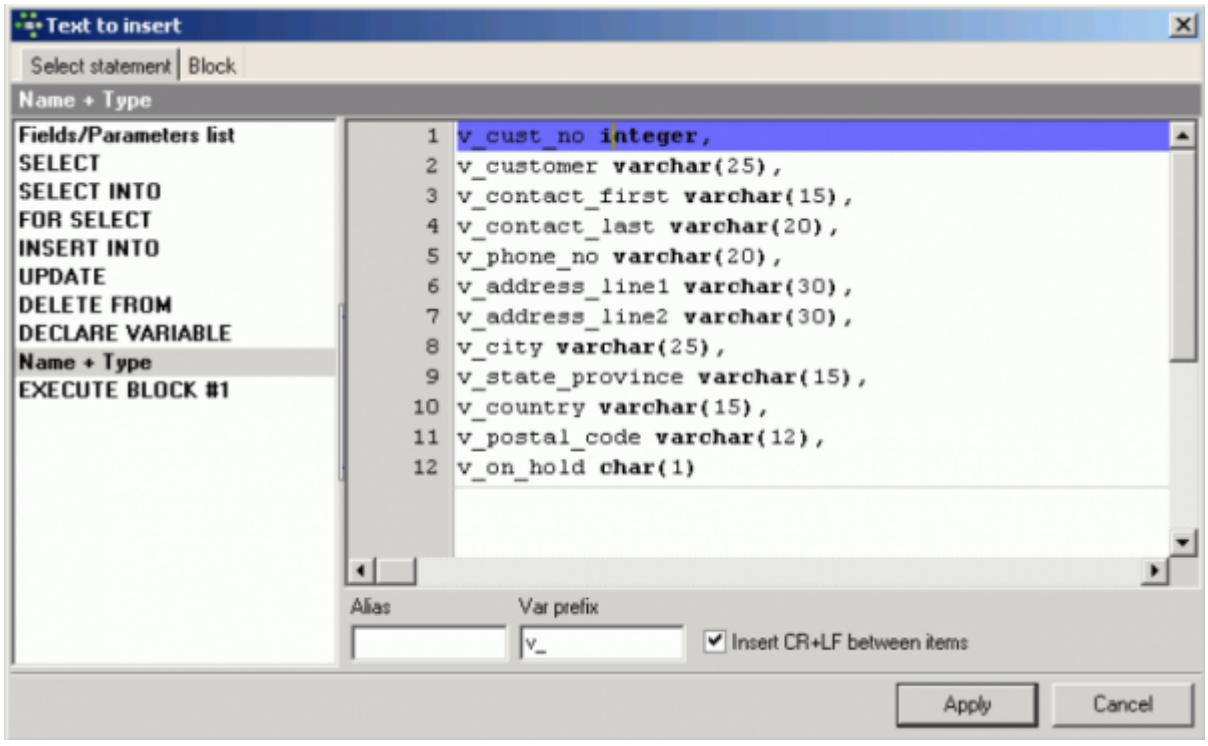
To prepare a SELECT INTO with carriage return and line feed, simply click on the SELECT INTO from the list on the left and check the *Insert CR+LF between items*. IBExpert then inserts the correct syntax:



In the case of this small table TEST1, this might not appear to be such an advantage, but if you take a look at a table with a larger number of fields (e.g. the EMPLOYEE CUSTOMER table), you will see how much it helps to have the field names and parameters already inserted into the standard syntax:

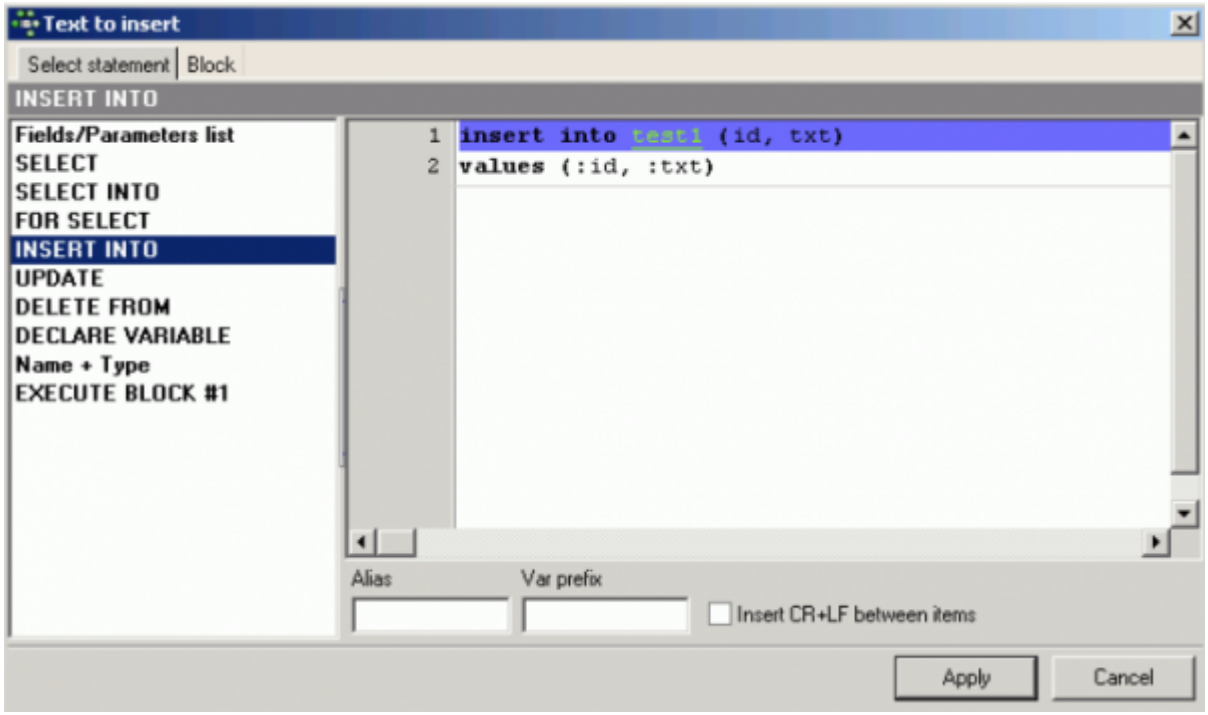


To ascertain the datatype definitions or to declare variables, simply click on the *Name + Type* in the left-hand list. Variable prefixes can be inserted (for example: v_) in the field *Var prefix* below, to offer you an instant full list of variables for all fields in the table.

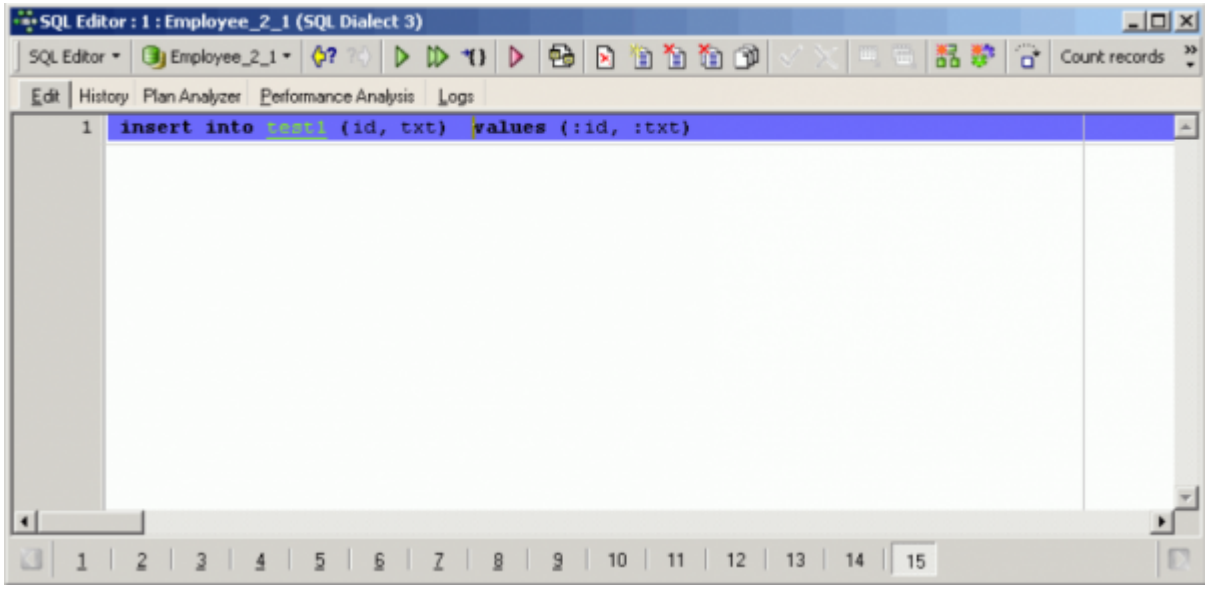


Firebird 2.1 also introduced the possibility to use domains for procedures, procedure parameters and so on. (Please refer to Using domains in procedures and the Firebird 2.1 Release Notes chapter, Procedural SQL for details and examples).

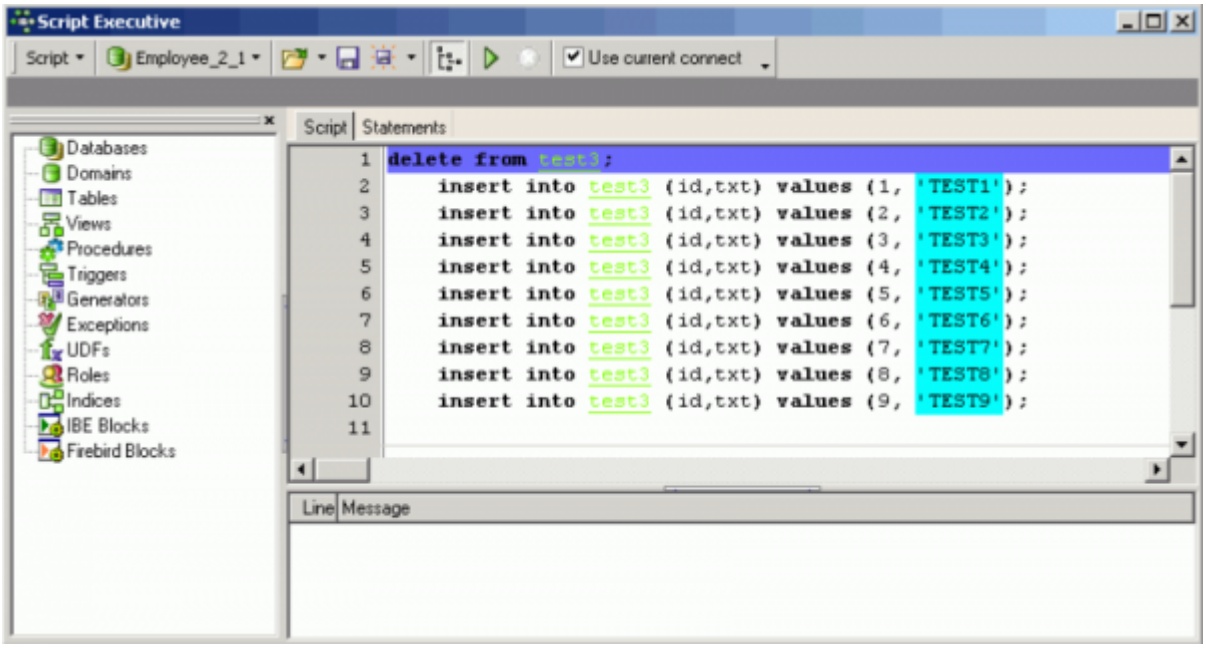
To continue with the implementation of the TEST1 table: an INSERT INTO statement is specified, without carriage return and line feed or a variable prefix:



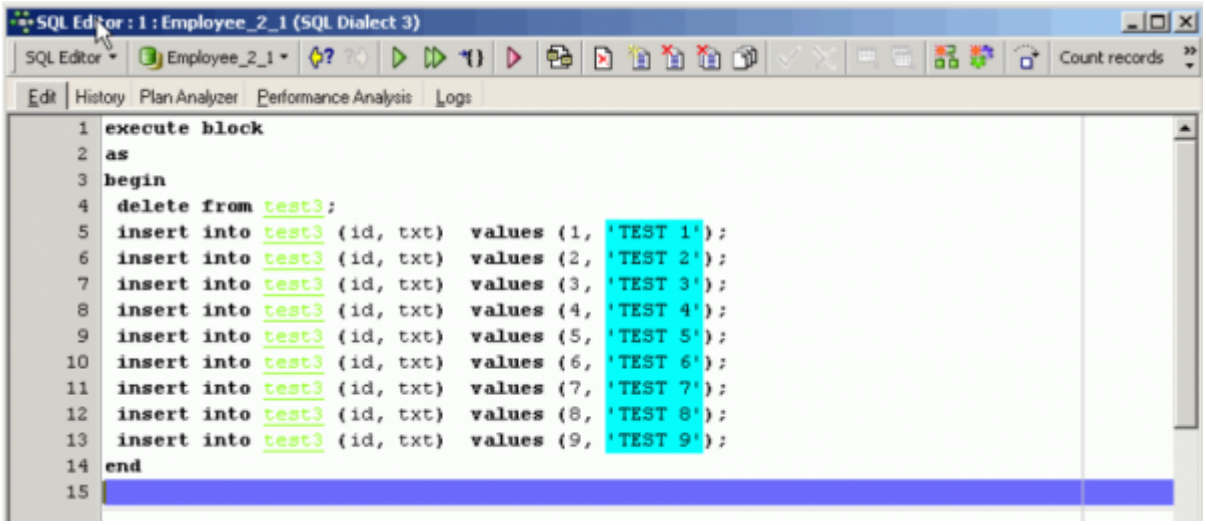
When it's ready simply apply and the INSERT INTO command is already formulated in the SQL Editor or Script Executive:



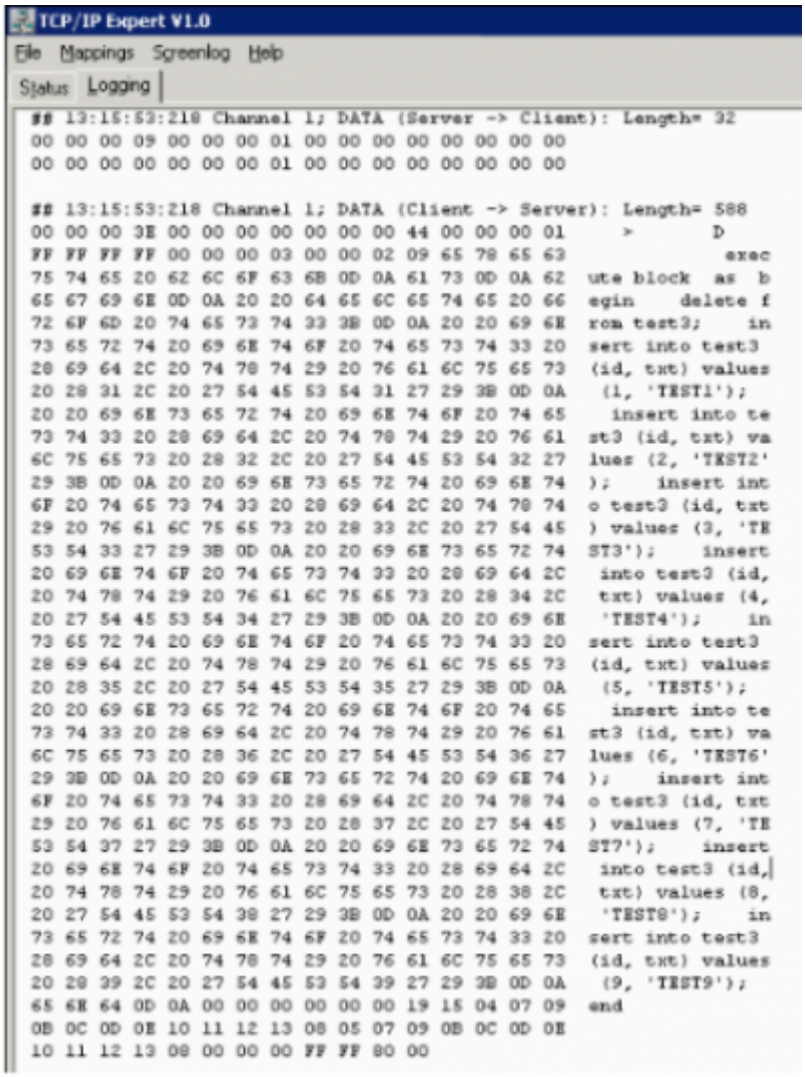
Now to illustrate one of the main advantages of Firebird blocks, some operations are added, one by one:



Add the beginning and closing clauses, to turn these statements into a block:



The Firebird server now processes all operations in one go, and you can see that all operations have been sent as one package to the server.



Especially when you need to insert or update a large amount of data, you can write your application in such a way as this, storing all the insert/update statements as a TString list or similar, writing EXECUTE BLOCK in front of it, concluding with an END, and executing it as a single statement.

Firebird 2.0 blocks can also be debugged directly in the SQL EditorSQL Editor (or alternatively in the Block Editor) using the Block Debugger.

There is a limit to the amount of source code that can be transferred in a single package, it may not be larger than 32 Kb. In the case of larger data packets, it is necessary to split them into multiple packages, but this is usually still more efficient that sending each command individually.

Transactions cannot be controlled from inside a block because the block is always a part of your client transaction.

Blocks were implemented in Firebird 2.0. InterBase® 2007 introduced something similar but it does not have all the functionalities that Firebird has.

When you are working with IBExpert, you can use IBEBlocks. Simply write IBEBLOCK instead of BLOCK and it still works!