SQL Tester

From Red Brook Software

Reviewed by Richard Stevens

SQL Tester is a utility which allows you to manage and test your SQL statements separately from your Delphi application. You can import the statements in a number of ways, test them individually or in batches, make amendments, and then restore them from whence they came.

Installation And Documentation

Installation was straightforward. SQL Tester uses the BDE, which is installed as part of the setup, as are the MDAC data components and the latest version of Crystal Reports. You're left with a single icon on your `Programs` menu which launches a neat-looking interface (Figure 1) that was obviously written with the Delphi database controls: our old friends `TDBNavigator` and `TDBGrid` are very much in evidence. The layout is clear and sensible, making it easy to select different means of data access (BDE or ADO), different database aliases, and different tables from within the selected database.

The documentation (a single help file) is a bit disappointing. It consists of a very bare and minimal reference guide to the various menu options, with nothing in the way of examples or 'how to' explanations. Even the individual reference entries lack a `Next` button, which means you have to go back to the top-level menu in order to move on to the next item.

Using The Program

You can use the program in a number of ways. The most straightforward, if hardly the most useful, is simply to select a database alias, type in a SQL statement, and press the `Execute` option. Parameters can be entered manually using a simple editor (Figure 2). Alternatively you can load them in, a statement at a time, from text files. Finally, and most usefully, you can extract them automatically from a Delphi .dfm file, providing (a very important proviso) that the file has been saved in text format.

Your statements are stored as records in a simple Paradox table, and along with each statement is stored an optional description, a group code (used in batch testing), the name of the database alias it is associated with, and the type of database access used. There are tools for searching and filtering records in order to locate particular statements, and they can also be printed out in a simple report format.

The 'testing' of SQL statements is rather disappointingly basic. Pressing the `Execute` button against a single statement simply attempts to execute the statement and returns you either the result set (in a grid at the bottom of the screen) or the error message in a standard database error dialog box. There didn't appear to be any additional interception or work done by the program in order to identify or diagnose any problems you may encounter.

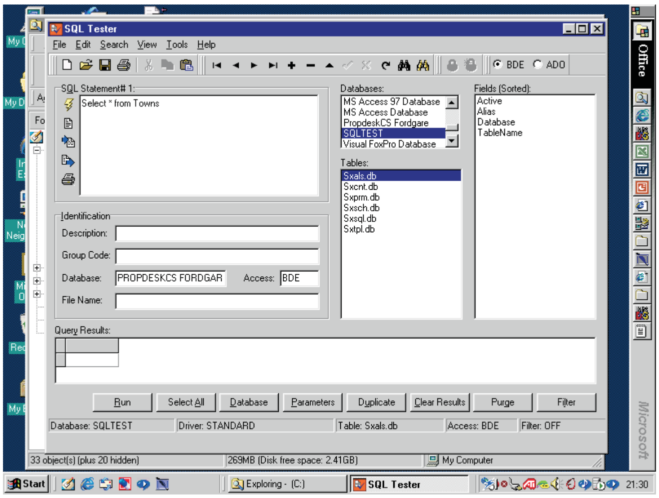
You can also execute statements in a batch, either by entering in a range of record numbers, by specifying a particular group code, or by specifying that all statements belonging to a particular database be executed. At the end of this process you get a simple dialog reporting back your success/failure ratio, along with a grid displaying the statements that failed (see Figure 3).

If you are entering in a lot of new SQL statements which follow a consistent pattern (eg, `SELECT * FROM Table where SomeCondition`) then you can create templates, allowing you to easily insert multiple commands of the same format.

Conclusions

I can't fault the product for not doing what it claims. It's easy to use and I encountered no problems using it. It's certainly a nice example of Delphi's database components in action. But it's also very difficult to get enthusiastic about it: the program lacks a number of those useful extra touches and polishes which would

➤ *Below: Figure 1*
Right: Figure 2



SQL Statement# 1: Select * from Towns

Databases: MS Access 97 Database, MS Access Database, PropdeskCS Fordgare, SQLTEST, Visual FoxPro Database

Tables: Sxals.db, Sxcnt.db, Sxprm.db, Sxsch.db, Sxsql.db, Sxtpl.db

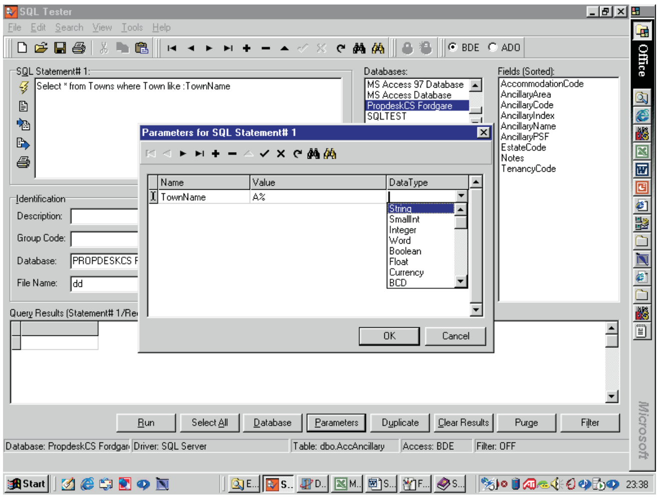
Fields (Sorted): Active, Alias, Database, TableName

Identification
Description:
Group Code:
Database: PROPDESKCS FORDGAR Access: BDE
File Name:

Query Results:

Database: SQLTEST | Driver: STANDARD | Table: Sxals.db | Access: BDE | Filter: OFF

33 object(s) (plus 20 hidden) | 269MB (Disk free space: 2.41GB) | My Computer



SQL Statement# 1: Select * from Towns where Town like :TownName

Databases: MS Access 97 Database, MS Access Database, PropdeskCS Fordgare, SQLTEST

Fields (Sorted): AccommodationCode, AncillaryArea, AncillaryCode, AncillaryIndex, AncillaryName, AncillaryPSF, EstateCode, Notes, TenancyCode

Parameters for SQL Statement# 1

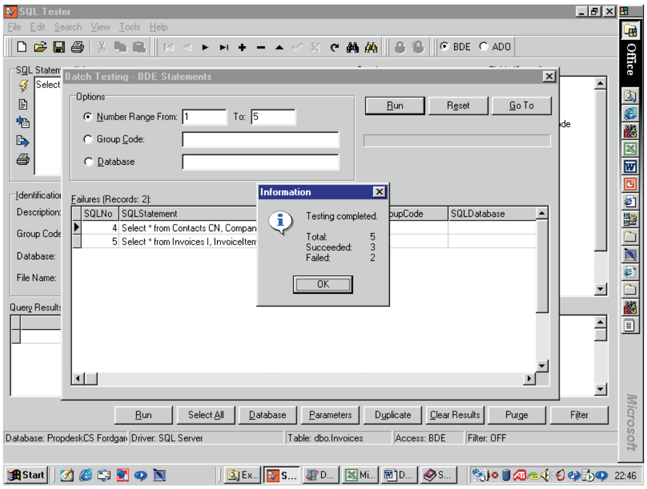
Name	Value	DataType
TownName	A%	

DataType: String, SmallInt, Integer, Word, Boolean, Float, Currency, BCD

Identification
Description:
Group Code:
Database: PROPDESKCS F
File Name: dd

Query Results (Statement# 1/Re

Database: PropdeskCS Fordgar | Driver: SQL Server | Table: dbo.AccAncillary | Access: BDE | Filter: OFF



● Number Range From: 1 To: 5
○ Group Code:
○ Database

Failures (Records: 2):

SQLNo	SQLStatement	...upCode	SQLDatabase
4	Select * from Contacts CN, Compan...		
5	Select * from Invoices I, InvoiceItem...		

Testing completed.

Total: 5
Succeeded: 3
Failed: 2

Database: PropdeskCS Fordgar... Driver: SQL Server Table: dbo.Invoices Access: BDE Filter: OFF

➤ *Figure 3*

push it into the 'highly recommended' category. One example of this is in the batch testing, where you are told which statements failed but not what the error message was. Similarly, you can't execute statements in a 'safe' mode: if you execute a statement that drops every table in the database, it will do just that. And for a product that is geared around writing and correcting SQL, it's a bit disappointing to be simply confronted with a plain text editor in which to write and edit your statements: some kind of syntax highlighting would have been useful.

If you write a lot of SQL statements, then this is an easy way of working through them and testing them. I feel that potentially its most useful application is as a 'reverse engineering' tool for Delphi applications: there are few tasks more tedious than picking through a form or a data module and repeatedly opening up the SQL editor to find out exactly what each query component is doing. It could also be useful for checking whether statements that work against one database server will transfer happily to another, or in a training environment.

In conclusion I'd say that SQL Tester is a competent if uninspired program. There's the potential for a handy product here, but it needs a lot more work to make it a 'must buy' purchase. If it was offered as shareware I would be more forgiving of its rather limited functionality, but I feel a price tag of $195 is a little steep for what you actually get. Visit www.redbrooksoftware.com for more information.

Richard Stevens is a Director of BOE Information Systems Ltd, a UK-based firm specialising in software for the Commercial Property Sector.