

Tutorial 19 - Direct SQL programming

The default form of the Database Retrieval component provided by Scribe is designed to relieve the user from having to code SQL queries manually. The Retrieval component's *Structured* form provides a graphical interface for selecting the retrieved table fields, search criteria, sorting orders and many other aspects of data retrieval, covering most of the needs of the report developer.

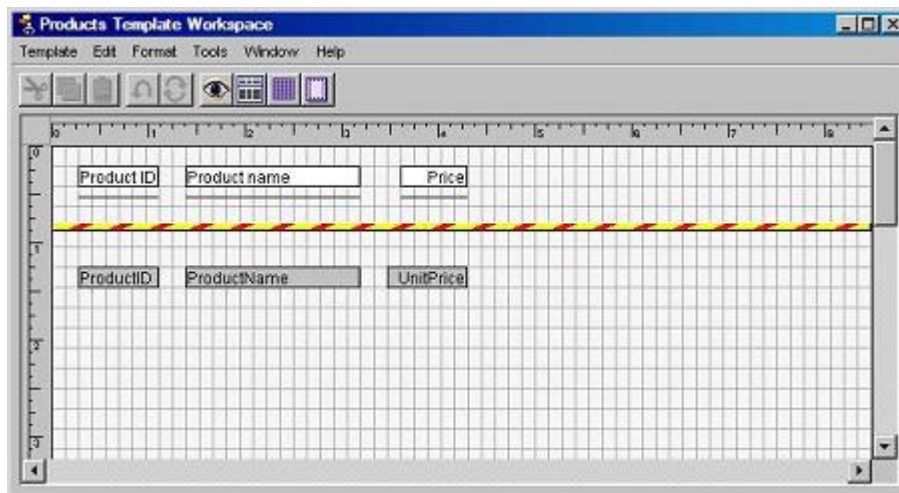
However, sometimes there is a need to construct a database query that does not lend itself to the structured (and, therefore, constrained) way of the Database Retrieval component that it provides "out of the box". Such may be the case of a union of two or more tables, or use of syntactical constructs specific to the particular RDBMS providing data for the report.

Scribe provides an alternative form of the Retrieval component, called *Direct SQL*. Using this form, the query can be coded in "raw" SQL, taking advantage of the efficiency and flexibility of a "one-step" database retrieval afforded by complex SQL statements.

In this Tutorial, we will construct a simple report that lists products offered by the Northwind Traders company, sorted by a unit price in descending order, and limited by a user-supplied number of products. Such report can be used in answering questions like "what are the top 5 most expensive products sold by N.T.?"

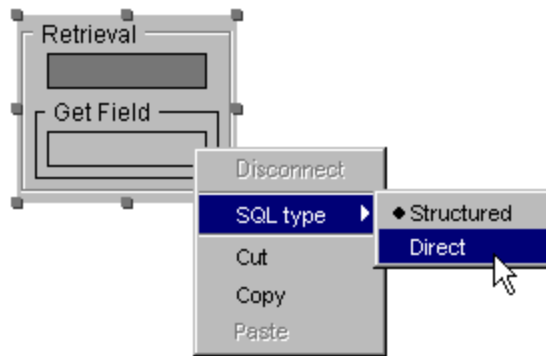
Note: the SQL statement used in this Tutorial is based on the SELECT TOP construct, that is supported by MS SQL Server 2000, but may not be applicable to other database systems. In particular, it is not supported in McKoi SQL.

We will begin our work by constructing a Print Template for the report, just to get it out of our way, so that it is there when we need it. Here is the Template:



The centerpiece of the Report Procedure will be the Retrieval component in its Direct SQL form, so we will start by dragging the Retrieval component into the Workspace and setting it to the Direct SQL option by clicking on the component with the right mouse and selecting the appropriate option from the pop-up menu, as shown below:





1. Right mouse click, select from the pop-up menu

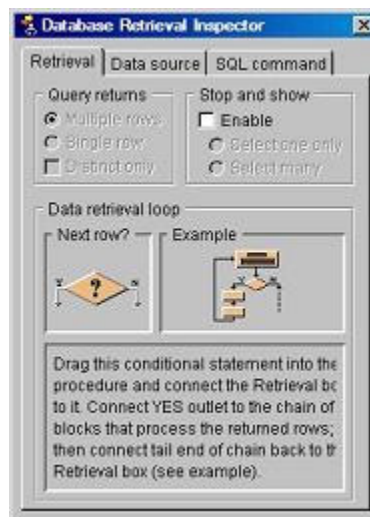
2. After selection

After setting it to the *Direct* SQL type, the Retrieval component changes its appearance and behavior:

- its Header area is set to "Direct SQL" heading, as opposed to a table name in the *Structured* form
- the Header area is not responsive to the mouse-dragging (i.e. a table name cannot be dragged-and-dropped into it)
- the component is no longer resizable, and like the Header, its field area does not respond to mouse-dragging.

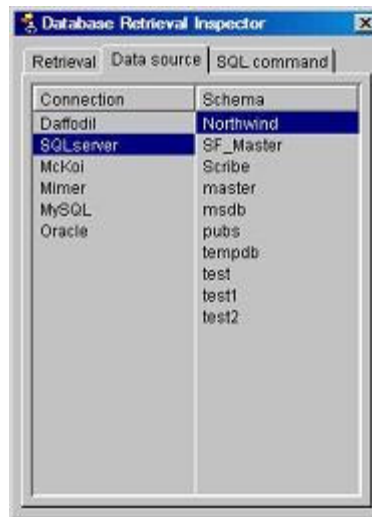
The main coding/configuration action in the Direct SQL happen in the Database Retrieval Inspector which now displays three tab views.

Retrieval tab



This view does not differ much from its Structured cousin, except that the "Query returns" options are rigidly set to "Multiple rows", and cannot be changed. The "Distinct only" modifier is not used in Direct SQL queries, instead its effect can be achieved by coding the DISTINCT keyword in the SELECT statement.

Data source tab



In this tab view we set the database connection and the default schema that will be used for executing the SQL query. We will not qualify the table name in the SELECT statement with the schema name (although we could), and the schema will be taken from the Data source Inspector view. At the run time, before running the SQL query, Scribe will issue the following command to the SQL Server:

```
use database Northwind
```

that will establish the default schema ("database" in SQL Server's terminology) to be used for all tables and fields not specifically qualified with the schema name.

SQL command tab

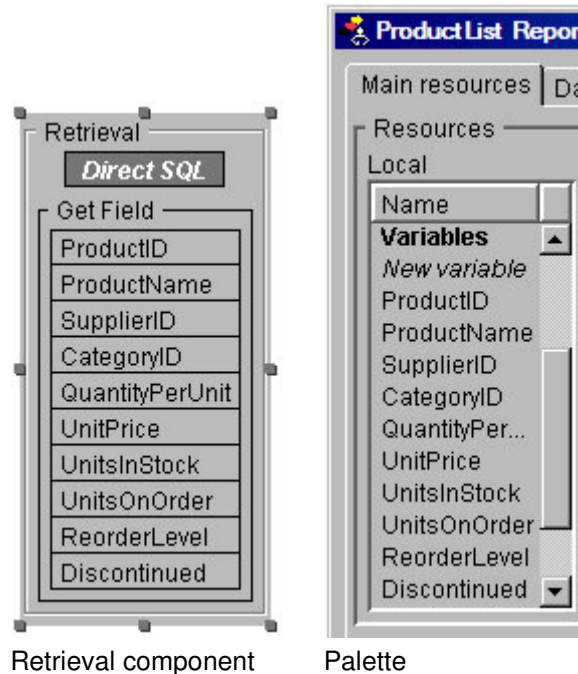
We are now ready to specify the actual SQL statement that will be used to retrieve the Products data. In it, we will use the "TOP N" construct to limit the number of returned rows:

```
SELECT * FROM Products
WHERE UnitPrice in
(SELECT TOP 5 UnitPrice FROM Products ORDER BY UnitPrice DESC)
ORDER BY UnitPrice DESC
```

Here is what it looks like in the Inspector:

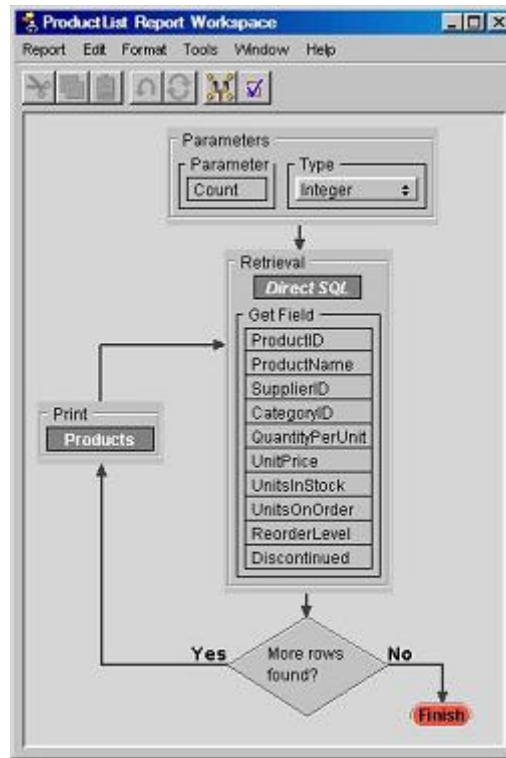


At this point, the Retrieval component does not yet know what fields will be returned by the query. The query in the form shown above uses a wildcard (*) to retrieve all the fields from the Products table, but even if it listed specific field names, the component still would not know their data types, until the query is *evaluated*. Pressing the Evaluate button at the bottom of the Inspector sends the query to the database server, and the response that comes from it is used by Scribe to set the list and data types of the returned fields in the Retrieval component, as shown in the following figure:



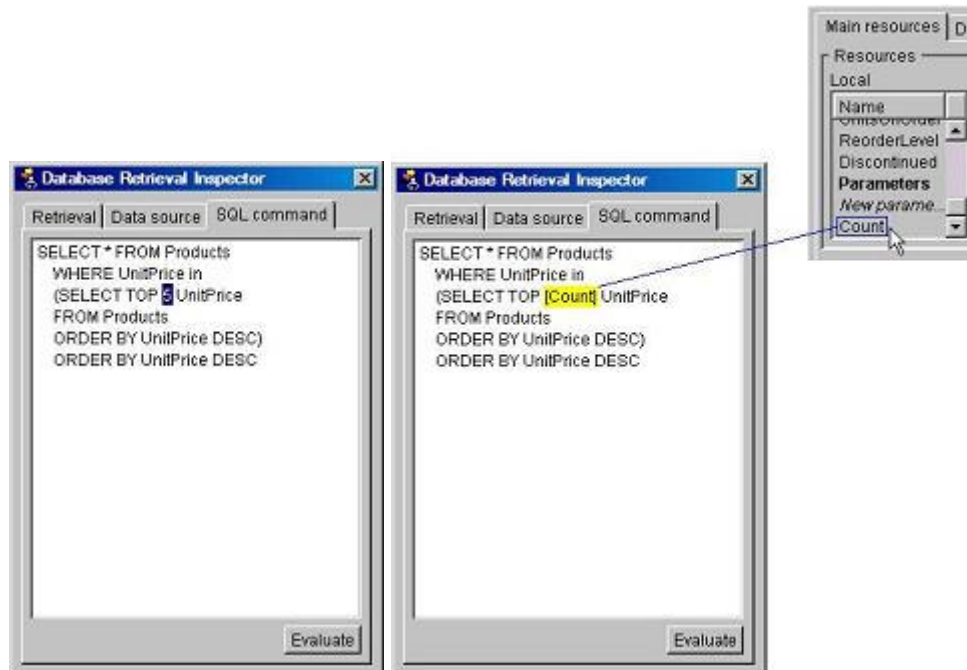
Note that, as the figure shows, the retrieved field names are also added to the Palette as local variables.

We can now configure the Report Procedure that retrieves and reports the Products data:



Variables in the Direct SQL

In order to let the user specify the desired number of returned products, we have to make the "TOP N" part of the query variable. Scribe provides for this by embedding the names of local variables and parameters inside the SQL query and enclosing them in square brackets ([and]). To replace the hardcoded number of rows (5) with a parameter, select the "5" in the Inspector text editor, and click on the "Count" parameter in the Palette:

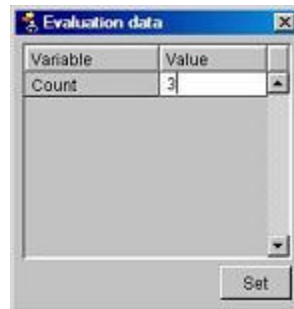


Select text to be replaced

[Count] is embedded into query

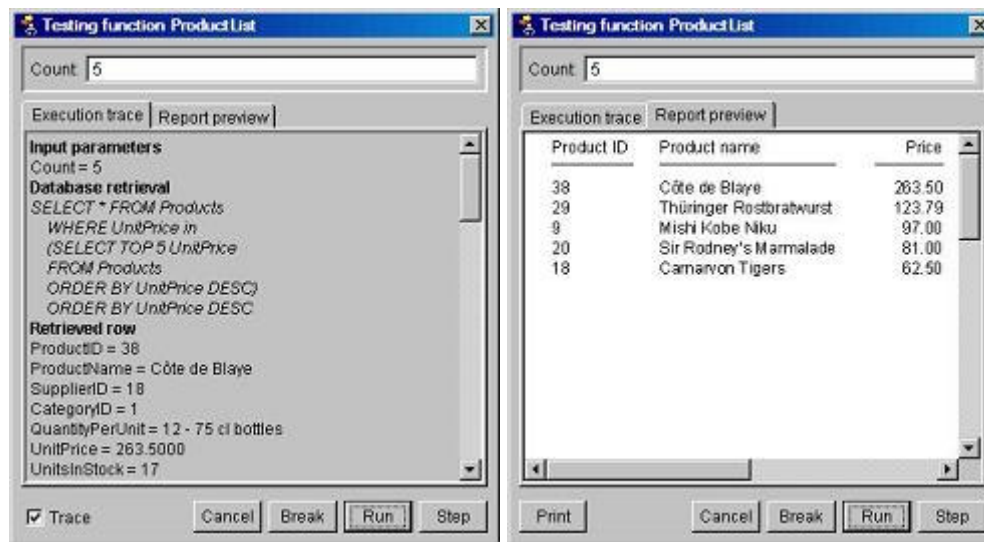
When the SQL query contains embedded variables, the evaluation function requires the user to supply sample

values for the variables so that Scribe can format a valid query acceptable to the database server. If we press the Evaluate button now, the following window is displayed:



In our particular case, any value for the Count variable is valid, as long as it is recognizable as a number. If the variable carries a text value, it would have to be enclosed in single quotes. If the provided sample value is invalid, Scribe will display a diagnostic message at the bottom of the Inspector.

Running the report in the test mode, we can get the following results:



Execution trace

Sample report