The Nearly Invisible Database
or
ForthQL

N.J. Nelson

Abstract

Structured Query Language (SQL) is a strange thing to Forth programmers, since it is neither structured, nor confined to queries, nor a language in the Turing sense of the word. It requires an interface written in another language in order to do anything. This paper describes our attempts to provide an SQL-Forth interface which is so smooth that you hardly know which side of it you're on.

N.J. Nelson B.Sc., C.Eng., M.I.E.T.
Micross Electronics Ltd.,
Units 4-5, Great Western Court,
Ross-on-Wye, Herefordshire.
HR9 7XP U.K.
Tel. +44 1989 768080
Fax. +44 1989 768163
Email. njn@micross.co.uk

Introduction

Structured Query Language (SQL) is the most widely used language for extracting information from a database. To a Forth programmer, it seems a rather strange language, and is particularly badly named. It has no words for structured programming - no IF..THEN, no DO..LOOP. It can be used to insert data and manage databases, as well as querying. It is not a language at all in the conventional sense of the word, as it has no input / output facilities of its own, and it cannot be compiled into an executable. It can only function in conjunction with some other program or language, in our case of course, Forth.

SQL

Like Forth, SQL has an ANSI standard, and also like Forth, that has not stopped there being a plethora of dialects. Fortunately, there are a number of implementations of SQL released under the General Public License (GPL). We chose to use MySQL, which has an excellent reputation for stability, and is available with a dual license - both GPL and commercial. There are also GPL support programs, such as browsers and management tools.

There are various methods of connecting to MySQL from a Forth program, including ODBC, which should give a degree of implementation independence. In practice, differences between the dialects means that independence is not very practical, and we opted for a direct connection to MySQL using its dynamic link library (dll).

Here is a very simple database query as expressed in SQL.

```
SELECT cuscode,cusname FROM customers
WHERE cusid BETWEEN 1 AND 10
```

For clarity, the SQL words are in capitals, and the parameters are in lower case. Cuscode and cusname are names of columns in the database table Customers. Cusid is another column which is used here to select a range of customers. The query must be passed to MySQL as a zero-terminated string. Note that, typically, part of the query string will be fixed, and part will need to be dynamically generated, for example, some of the parameters will be taken from the controls of a dialog box. This is not too bad in the case of a very simple query, and standard string handling words can be used.

In real life, queries are much more complex, sometimes running to hundreds of lines of code, all transferred as a single gigantic z-string! There may be multiple WHERE clauses, subqueries, temporary tables, sort order instructions, joining of two or more tables, and so on. Using conventional techniques, the program soon becomes completely unreadable.

A further interesting difficulty is that, once SQL has extracted the information for you, it then doesn't tell you what the answer is! You need to ask for it, item by item.

A more practical interface is clearly needed, and ideally, it should hardly be noticeable when you switch between Forth and SQL, with SQL providing the database access words, and Forth providing the parameters and program structure. We have called this the "Nearly Invisible Interface".

The nearly invisible interface

When designing this interface, we had the following aims:
a) The code should be easily understandable
b) Therefore, it must be possible to embed comments
c) The code should be as concise as possible
d) It must be possible to structure the code
e) The Forth to SQL switch should be barely visible
f) It should be possible to debug in interactive mode

We looked at the possibility of using a vocabulary for implementing the SQL words, however, this is not very useful here firstly because all SQL words essentially do the same thing (concatenate a z-string) and because there may be embedded numerics. Instead, we decided to trigger a different mode when SQL was selected.

A marker is needed for the switch from Forth to SQL, and almost all the symbols are already used. We chose the character ¦ because it is not used for anything else and is easily accessed on a UK keyboard.

We can suppose now that the example above looks in ForthQL

```
: TEST ( --- ) \ Sample query
  SQL¦ SELECT cuscode,cusname FROM customers \ Fixed part
  WHERE cusid BETWEEN
  ¦ LOWLIMIT ¦ AND ¦ HIGHLIMIT                \ Parameters
  ¦SQL> TESTOUT                               \ Output
;
```

Implementation

Looking at the above words and what they need to do:

SQL¦ (---zaddr)
Starts SQL mode, initialising a zero terminated string and concatenating verbatim until the following ¦ . Any bracketed or backslashed comments are removed. However, formatting is preserved by adding carriage returns into the string.

¦ (zaddr---zaddr')
Resumes SQL mode as above until the next ¦ .

¦SQL> (--- ; zaddr,<name>---)
On compilation, discards the address of the now completed query string (which is a fixed, known location). Compiles the the run-time part of the SQL query word, followed by the address of the following word. When executing, it first performs the query then retrieves the result piece by piece, passing each section to the following word which, perhaps, displays it.

Note that SQL queries sometime take quite a long time to execute and are therefore frequently executed in a separate thread. There then arises the possibility of multiple queries being executed simultaneously in different threads of one application. All the ForthQL words must therefore be thread-safe, and there must be a separate buffer for each thread to contain the string.

We can also image another word
¦SQL (--- ; zaddr---)
Similar to the above, but which returns no result, so is useful for inserting data or managing the database.

And a default word
SQL-RESULT
would be useful, which formats the result nicely onto the Forth console.

Examples

Suppose we have a table "Delegates", with columns Firstname, Lastname and Country_code

```
SQL¦ SELECT firstname,lastname FROM delegates
WHERE country_code = 49 ¦SQL> SQL-RESULT
+-----------+-----------+
| firstname | lastname  |
+-----------+-----------+
| Klaus     | Schleisik |
+-----------+-----------+
1 row in set (0.01 sec)
ok
```

The formatting of the default result word is designed to look similar to the MySQL Command Line Client, whose output is shown on most MySQL reference books.

```
SQL¦ INSERT INTO delegates
VALUES ('Chuck','Moore',1) ¦SQL
```

Could hardly be easier, could it?

Conclusion

Once again, Forth demonstrates its versatility by offering an SQL interface which is far more natural and readable than can be obtained in most other languages.