# OpenCOBOL SQLite engine

| | |
|---|---|
| **Author:** | Brian Tiffin |
| **Date:** | 09-Oct-2008 |
| **Version:** | 0.90 based on SQLite 3.5.9 |
| **Rights:** | Copyright (c) 2008, Brian Tiffin Licensed under the GNU Public License 2.0 ocsqlite.c is in the public domain in keeping with the original SQLite source code This Usage documentation licensed FDL |
| **Purpose:** | Demonstrate SQLite in OpenCOBOL |
| **Needs:** | libsqlite3-dev and sqlite3 packages or libsqlite and sqlite version 2 packages |
| **Tectonics:** | cobc -c ocsqlite.c cobc -x -lsqlite3 sqlscreen.cob ocsqlite.o |
| **Docgen:** | $ ./ocdoc sqlscreen.cob sqlscreen.rst sqlscreen.html ocfaq.css |

# Demo

sqlscreen.cob demonstrates some of the features provided by the SQLite shell interface.

# Tectonics

Requires sqlite3 and libsqlite3-dev packages (for Debian) Requires OC1.1 packaged on or later than Oct 13th 2008:

```
$ cobc -c ocshell.c
-or-
$ cobc -c -DSQLITE_OMIT_LOAD_EXTENSION ocshell.c

$ cobc -x -lsqlite3 sqlcaller.cob ocshell.o
-or-
$ cobc -x -lsqlite3 -fdebugging-line sqlcaller.cob ocshell.o
```

# Meta Commands

All of the meta (dot) commands supported by the **sqlite3>** console are included in the ocsqlite.c interface.

A query of *.help* will list the allowed meta commands.

```
Many of the meta commands output results to stdout.

A few of the commands can be dangerous, and may be
disabled in the 1.0 release.  Commands such as .bail on
```

# SQL

SQLite includes nearly all SQL verbs and most database management commands. Full details can be found at http://www.sqlite.org/lang.html

# Source code notes

The database name is passed as a zero terminated C string:

```
01 database             pic x(8) value 'test.db' & x'00'.
```

The initialization is somewhat misleading. The db pointer is set, but the internal ocshell.c database pointer is used on all the queries:

```
call "ocsqlite_init" using
        db
```

```
            database
            by reference errstr
            by value function length(errstr)
        returning result
    end-call
    if result not equal zero
        display "Result: " result end-display
    end-if
```

SQLite uses a callback for each row of a query:

```
    set callback-proc to entry "callback"
```

And a lot of code only executes if sqlscreen.cob is compiled with -fdebugging-line:

```
    >>Dmove ".echo on" to query
    >>Dperform ocsql-exec

    >>Dmove ".help" to query
    >>Dperform ocsql-exec

    >>Dmove ".tables" to query
    >>Dperform ocsql-exec

    >>Dmove ".timer on" to query
    >>Dperform ocsql-exec

    >>Dmove ".mode tcl" to query
    >>Dperform ocsql-exec

    >>Dmove 0 to row-counter
    >>Dmove "select * from trial;" to query
    >>Dperform ocsql-exec

    >>Dmove ".mode html" to query
    >>Dperform ocsql-exec

    >>Dmove 'insert into trial values (null, "string", "2008-10-10");'
    >>D  to query
    >>Dperform ocsql-exec

    >>Dmove "select * from thisfails;" to query
    >>Dperform ocsql-exec

       move "drop table trial;" to query
       perform ocsql-exec

       move "create table trial (first integer primary key, " &
           "second char(20), third date);" to query
       perform ocsql-exec

    >>Dmove "pragma count_changes=1;"  to query
    >>Dperform ocsql-exec

    >>Dmove "pragma database_list;"  to query
    >>Dperform ocsql-exec

    >>Dmove ".schema trial" to query
    >>Dperform ocsql-exec

       move 'insert into trial (first, second, third) values ' &
           '(null, lower(hex(randomblob(20))), datetime()); ' &
           'insert into trial values (null, "something",' &
           ' julianday());' to query
       perform ocsql-exec

    >>Dmove "select * from trial;" to query
    >>Dperform ocsql-exec

    >>Dmove "pragma count_changes=0;"  to query
    >>Dperform ocsql-exec
      *><*
      *><* Most of the default demo is here::
      *><*
       move 'insert into trial (first, second, third) values ' &
           '(null, lower(hex(randomblob(20))), datetime()); ' &
           'insert into trial values (null, "something",' &
```

```
           ' julianday());' to query
       perform ocsql-exec
```

The *.mode column* and *.width* meta statements are critical for this example. The sqlite shell will now return fixed length fields. Breaking DRY, the widths must be explicitly set.

A *.mode csv* could be used with an unstring for variable length datums, but then quotes will need processing as well.

```
move ".mode column" to query
perform ocsql-exec

move ".width 10 20 20" to query
perform ocsql-exec

move 1 to row-counter
move "select * from trial;" to query
perform ocsql-exec
display function trim(sql-table trailing) end-display

subtract 1 from row-counter giving row-max end-subtract
perform varying row-counter from row-max by -1
    until row-counter < 1
        move sql-records(row-counter) to main-record
        display "|" key-field "|" end-display
        display "|" str-field "|" end-display
        display "|" date-field "|" end-display
end-perform
```

Finally put up a screen, cycling through the records:

```
perform varying row-counter from 1 by 1
    until row-counter > row-max
    move sql-records(row-counter) to main-record
    accept entry-screen end-accept
end-perform
```

Once again, the alpha release of ocshell.c keeps its own database handle. That will change. The close call is still necessary, just that the db handle is not used.:

```
call "ocsqlite_close"
    using
        by value db
    returning result
end-call

move result to return-code
goback.
```

# Callback

Callback procedure. In sqlite_exec, the callback is passed:

```
void *user_data, int fields, char **columns, char **names
```

for each row, with the name and value data in separatearrays.

The OpenCOBOL callback procedure is called with:

```
pointer, int fields, row as alphanum, row length
```

Each line of row data is formatted according to the shell's *mode* setting. The expectation is .mode column with fixed .width

In a demonstration of very bad form; if the external value of row-counter is larger than 0, fill the external sql-records structure:

```
move value-display to main-record
if row-counter > 0
    move main-record to sql-records(row-counter)
```

```
          add 1 to row-counter end-add
      end-if
```

Cheers