NAME

`csvtab` — virtual table representing a CSV file

LIBRARY

SQLite User-defined function Library (csvtab.so)

SYNOPSIS

`create virtual table` *`tablename`* `using` **`csvtab`** (`csvfile`, [*`colname ...`*])

DESCRIPTION

The **`csvtab`** virtual table renders `csvfile` as a table with user-defined column names. If the *`colname`* list is empty, the first line of the file is used for column names.

To use **`csvtab`** in the SQLite shell **`sqlite3`**, add the following to your `${HOME}/.sqliterc` file:

```
.load /usr/local/lib/sqlite3/csvtab.so
```

IMPLEMENTATION DETAILS

`csvfile` is mapped into memory using `mmap`(2). The contents of the virtual table are *not* cached. That is, every query initiates a call to `mmap`(2) and each row is re-parsed.

Within **`csvtab`**, the caller provides the parser with a buffer and a callback function that the parser invokes for each parsed column. The **`parse`**() function parses a single row in the buffer, and returns a *`char *`* pointing to the position immediately following the last character in the row. Parsing is completed when **`parse`**() returns a pointer one past the end of the buffer.

This version of **`csvtab`** supplies no indexing or sorting features. If the query given to SQLite mentions a **`csvtab`** column in a *WHERE*, *JOIN*, *GROUP BY*, or *ORDER BY* clause, SQLite will require the data to be sorted. Because **`csvtab`** makes no attempt to do so, SQLite will sort the data itself by writing the rows returned by **`csvtab`** initially to a temporary table with an appropriate index. For files with millions of records, this will take some time. You may find it faster to import the CSV into a table using the SQLite shell or with a **`INSERT INTO ... AS SELECT * FROM ...`** query.

NOTES

The CSV parser is a separate module independent of **`csvtab`**. It is a pure context-free language parser, written expressly (and reluctantly) for the purpose because existing libraries were not suited to the task. For example, Robert Gamble’s well regarded libcsv (http://sourceforge.net/projects/libcsv/) parses buffer-by-buffer, not record-by-record. The SQLite virtual table methods request data row-by-row.

The first version of **`csvtab`** *was* written using libcsv, but the result was rather more complicated than the present version: the library forked a parser, which fed records into a pipe that the virtual table function **`xNext`**() read. Each parsed value had several copies in memory:

1. The buffer provided to the parser
2. The buffer returned by the parser to the caller
3. The kernel buffer for IPC pipe
4. The buffer into which **`xNext`**() read the value
5. The buffer SQLite creates for *`SQLTRANSIENT`* data

The first two copies are necessary because the parser’s output is not a mere copy of its input; quotes are removed from quoted data and may appear any number of times in a field. The last copy is also required if **`csvtab`** doesn’t retain the parsed data for subsequent reference by SQLite. The present implementation has only those three copies.

AUTHORS

`csvtab` was contributed by James K. Lowden