ODBC.jl Documentation

Release development

Jacob Quinn

March 30, 2015

A low-level ODBC interface for the Julia programming language

Current Version: Developer Preview v0.0.0 Last Update: 3/25/2013 Latest updated notes: * Added batch statement support (assuming DMBS supports) * Added 'Write to file' functionality for returning resultsets directly to file (see *query()* below) * Overhaul of ODBC API low-level functions; all ODBC functions are now implemented, and users are free

to inspect 'ODBC API.jl' for useful functions

- Added a preliminary 'test.jl' file. Connects to the public ensembl.org MySQL database, requires the MySQL Driver to be installed prior

The ODBC Julia module header file is ODBC.jl. The module contains SQL const definitions and function definitions aligned with the ODBC API (http://goo.gl/l0Zin) in the *ODBC API.jl* and *backend.jl* files, respectively.

The package is officially listed in the Julia package manager now, so users can load the ODBC package as follows:

```
'julia julia> ## (The first two commands below load and initialize the package
manager) julia> require("pkg.jl") # Loads the package manager code <br> julia>
Pkg.init() # Creates the package repository (if not already done)<br> julia>
Pkg.add("ODBC") # Creates the repo and downloads the ODBC package <br> julia>
using ODBC #Tells Julia to load and look in the ODBC module when you use
ODBC-defined functions <br> '
```

Defined functions can be split into two categories: user-facing and backend. The user-facing functions defined so far are:

- *connect()*
- *advancedconnect()*
- *query()*
- *querymeta()*
- *listdrivers()*
- *listdsns()*
- *disconnect()*

These functions implement the backend functions which are built to mirror the ODBC defined functions very closely (along with appropriate error-handling).

Function Reference

connect(DSN, username, password)

Connect to a DSN

Connect requires the DSN string argument as the name of a pre-defined ODBC datasource. Valid datasources (DSNs) must first be setup through the ODBC administrator prior to connecting in Julia.

The *username* and *password* arguments are optional as they may already be defined in the datasource.

connect() returns a Connection object which contains basic information about the connection and ODBC connection and statement handles.

Typically, *connect()* is implemented by storing the Connection object in a variable to be able to disconnect or facilitate handling multiple connections. It's unneccesary to store the object though, as a global *conn* object holds the most recently created Connection object and other ODBC functions will use it by default in the absence of a specified connection.

advancedconnect(connectionstring)

Advanced DSN connection

advancedconnect implements the native ODBC SQLDriverConnect function which allows flexibility in connecting to a datasource through specifying a 'connection string' (e.g. "DSN=userdsn;UID=johnjacob;PWD=jingle;") See ODBC API documentation (http://goo.gl/uXTuk) for additional details.

If the connection string doesn't contain enough information for the driver to connect, the user will be prompted with the additional information needed.

Furthermore, on Windows, if an empty string is provided ("") the ODBC administrator will be brought up where the user can select the DSN to which to connect, even allowing the user to create a datasource or add a driver.

(An excellent resource for learning how to construct connection strings for various DBMS/driver configurations is http://www.connectionstrings.com/)

query(connection, query, output, delim)

Query a database

If a connection object isn't specified, the query will be executed against the default connection (stored in the global variable *conn* if you'd like to inspect).

Once the query is executed, the resultset is stored in a DataFrame by default (*output="DataFrame"*). Otherwise, the user may specify a file name to which the resultset is to be written, along with a the desired file delimiter (default *delim=','*). Depending on DBMS capability, users may also pass multiple query statements in a single query call and the resultsets will be returned in an array of DataFrames, or the user may specify an array of filename strings into which the results will be written.

For the general user, a simple *query(querystring)* is enough to return a single resultset in a DataFrame. The alternative *sql"..."* string literal may also be used and is equivalent to calling *query(querystring)*. Results are pointed to in the connection object (connection.resultset). *querymeta()* will execute the query string, but only return metadata about the resultset (useful for large query result inspection).

listdrivers() and *listdsns()*

These two functions take no arguments and invoke the driver manager to list installed drivers or defined user and system DSNs, respectively.

disconnect(connection object)

Disconnect closes the open connection object, frees all handles and resets the default connection object as necessary. If invoked with no arguments (i.e. *disconnect()*), the default connection is closed.

Issues

- I've had trouble being able to test on Linux and OSX systems, so the ODBC driver manager shared object libraries may not work correctly. If someone could help in testing, I'd really appreciate it
- I have a list of potential features to implement listed in a TODO file in this repository, suggestions welcome.
- Please send feedback! I feel like this has gotten to a point where I can share, but it's definitely far from where I'd like to be

-Jacob quinn.jacobd@gmail.com