FastBase MySQL + Tcl Interface Program v1.06

Documentation

This extension requires the FBSQL.DLL and LIBMYSQL.DLL files to be in the current path or directory. The following table lists commands in some sort of order (hopefully meaningful).

load fbsql.dll	This loads the fbsql extension and provides the Tcl interpreter with 6 new commands: **sql sql1 sql2 sql3 sql4 sql5**. All 6 commands are the same but they are independent of each other and can therefore be used to access different database servers, databases or different query results. Any of the sql commands can be replaced with sql1, sql2 etc.
sql connect [database_host] [user_name] [password] [database_name] [mysql_port]	Connect to the database_host (default is localhost) using the user_name provided (default is logged in user) and the password (if provided). The options for specifying the database and alternate mysql port are now also provided (v1.06). The option of selecting the database name on the connect command line removes the need for using the "sql selectdb" command at start-up.
sql selectdb database	Use the specified mysql database (case sensitivity applies for mysql Unix servers).
sql disconnect	Disconnect from the mysql server.
sql version	Returns the fbsql version number/message.
sql [query] statement	Execute the sql statement, the command *query* is optional. If the SQL statement returns results (SELECT, DESCRIBE, SHOW, EXPLAIN) then the command returns a list of all rows, each list is itself a list of all column values. If the SQL statement does not return any results (eg: UPDATE, INSERT, DELETE) then the sql query command returns nothing also. See **sql startquery** for more advanced options.
sql numrows	Returns the number of rows returned by the last query command If the last query was an UPDATE type statement then this command returns the number of affected rows.
sql startquery statement [-huge] [-array name]	Execute the sql statement but do not return any data. The statement must return results otherwise an error will be reported. To retrieve the data use the **sql fetchrow** command and to stop the query use the **sql endquery** command. The option **-huge** signifies that the result could be very large and therefore do not store the results in memory. This command must be used carefully because as per MySQL documentation of the mysql_use_result() function the table(s) being accessed will have a READ LOCK on them until the query is finished, therefore you will not be able to use INSERT, UPDATE or DELETE on the table being queried. The option **-array name** signifies that the **sql fetchrow** command should assign the column values to the array name specified, eg: "set name(FIELD_NAME) $column_value" for each column.
sql fetchrow	The next row is retrieved from the result set (see **sql startquery**). If no more rows are available then this command returns nothing {}. If the **sql startquery** option was used with **-array name** then the array is set with an element for each column, the return value is the array name. If the array option was not used then this command returns a list of each column value.
sql endquery	Stop the query started via **sql startquery**. Usually this is only done after the **sql fetchrow** command returns nothing but it can be used before this without any problems.

Examples

Example using simple sql commands

```
load fbsql.dll
sql connect 192.168.3.8 root ""
sql selectdb test

foreach row [sql "SELECT NUMBER, NAME FROM CUSTOMER ORDER BY NAME"] {
        set number [lindex $row 0]
        foreach row [sql "SELECT DATE, REF, TOTAL FROM TRAN WHERE CUSTOMER_NO = $customer(NUMBER) ORDER BY DATE, REF"] {
                set tran(DATE) [lindex $row 0]
                set tran(REF) [lindex $row 1]
                set tran(TOTAL) [lindex $row 2]
                # process transaction record ...
        }
}

sql disconnect
```

Example using startquery/fetchrow/endquery

```
load fbsql.dll
sql connect 192.168.3.8 root "" test

sql2 connect 192.168.3.8 root "" test

sql startquery "SELECT NUMBER, NAME FROM CUSTOMER ORDER BY NAME" -array customer

while {[sql fetchrow] != ""} {
        sql2 startquery "SELECT * FROM TRAN WHERE CUSTOMER_NO = $customer(NUMBER) ORDER BY DATE, REF" -array tran
```

```
        while {[sql2 fetchrow] != ""} {
                # tran() array contains one element for each field (note: in this example we retrieved all fields)
                # process transaction record ...
        }
        sql2 endquery
}

sql endquery

sql disconnect
sql2 disconnect
```