Attached files are based on the original technique for using MySQL with LR 9.5 tested in 2010 at Omnifone. The instructions below are based on the revalidation of the above technique using LoadRunner 12 beta in January 2014.  
  
Installing MySQL  
For the purposes of learning the basics of MySQL and its use with LoadRunner, I recommend the following:  
  
Installing WAMP Server from <http://www.wampserver.com/en/>  
I installed the 64-bit version. I already had the prerequisite Visual C++ components installed.

I did a “next”, “next”, “next” type install and accepted all default settings. This creates a simple MySQL database with a root user that has a blank password. I configured Apache to respond on port 8080 (since port 80 was already in use).   
  
After checking connectivity to the database using phpMyAdmin, I could see that a "test" schema was already in place as shown in the screenshot below.

Machine generated alternative text:
, 4 127.0.0.1:8080/localhost X
php M yAdm in
(Recent tables) E
information_schema
mysql
performance_schema
test
Structure
No tables found in database
H  Create table
Name Number of columns

MySQL was configured as per the table below:

|  |  |
| --- | --- |
| **Variable** | **Value** |
| SERVER\_NAME | 127.0.0.1 |
| USER\_NAME | root |
| PASSWORD |  |
| SCHEMA\_NAME | test |
| PORT | 3306 |

The following SQL can be executed to create a table containing some sample test data.

[**DROP**](http://search.mysql.com/search?site=refman-%35%31&q=DROP) [**TABLE**](http://search.mysql.com/search?site=refman-%35%31&q=TABLE) [IF](http://dev.mysql.com/doc/refman/5.1/en/control-flow-functions.html) [**EXISTS**](http://search.mysql.com/search?site=refman-%35%31&q=EXISTS) `test`.`sample**\_**reads`;  
[**CREATE**](http://search.mysql.com/search?site=refman-%35%31&q=CREATE) [**TABLE**](http://search.mysql.com/search?site=refman-%35%31&q=TABLE) `test`.`sample**\_**reads` (  
`sqForename` [**varchar**](http://search.mysql.com/search?site=refman-%35%31&q=VARCHAR)(128) [**DEFAULT**](http://search.mysql.com/search?site=refman-%35%31&q=DEFAULT) [**NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NULL),  
`sqSurname` [**varchar**](http://search.mysql.com/search?site=refman-%35%31&q=VARCHAR)(128) [**DEFAULT**](http://search.mysql.com/search?site=refman-%35%31&q=DEFAULT) [**NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NULL),  
`sqJobDesc` [**varchar**](http://search.mysql.com/search?site=refman-%35%31&q=VARCHAR)(128) [**DEFAULT**](http://search.mysql.com/search?site=refman-%35%31&q=DEFAULT) [**NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NULL));  
[**INSERT**](http://search.mysql.com/search?site=refman-%35%31&q=INSERT) [**INTO**](http://search.mysql.com/search?site=refman-%35%31&q=INTO) test.sample\_reads  
(sqForename, sqSurname, sqJobDesc)  
[**VALUES**](http://search.mysql.com/search?site=refman-%35%31&q=VALUES)  
('Bob','Marley','Singer'),  
('Sammy','Davies','Entertainer'),  
('Shane','Warne', 'Cricketer'),  
('William','Wales','Prince'),  
('Harry','Redknapp','Football Manager'),  
('Samuel','Beckett', 'Author');  
[**SELECT**](http://search.mysql.com/search?site=refman-%35%31&q=SELECT) \* [**FROM**](http://search.mysql.com/search?site=refman-%35%31&q=FROM) sample\_reads;

Once this SQL code has executed, the test data can be seen in the sample\_reads table.

Machine generated alternative text:
/4 127.0.0.1:8080/ localhost
& ñ D 127.0.0.1 :8080/phpmyadmin/index.php?token= 1 232d9bf54a365fbe3fe2
php MyAdmTn E  bchost »test »Wsanwflads _______
I!I Browse  Structure ¿J SQL -4 Search *i! li
(Recent tables)
Show query box
[ This table does not contain a unique column. Grid edit. checkbm
SELECT *
FROM sample_reads
LIMIT 0, 30
sqForename
Bob
Sammy
Shane
William
Harry
Samuel
sq Surname
Marley
Davies
Warne
Wales
Redknapp
Beckett
sqJobDesc
Singer
Entertainer
Cricketer
Prince
Football Manager
Author
! information_schema
mysql
! performance_schema
test
New
sample_reads
Columns
_‘ Showing rows 0 - 5 (6 total Query took 00006 sec)
Show : Start row: I o I Number of rows: ¡ 30 ¡ Head
L ___________
+ Options

**Setting Up LoadRunner to use MySQL**

To enable the MySQL procedures within a LoadRunner script to run on a PC the following needs to be done:

Extract the zip file ‘MySQL LoadRunner libraries.zip’ to the LoadRunner home directory.

(e.g. for LR 12 C:\Program Files\HP\LoadRunner)

This adds the necessary .h files into the LoadRunner “include” folder and the required .dll into the “bin” folder.

In your LoadRunner script add the following to the top of the vuser\_init or Action section:

#include "Ptt\_Mysql.h"

In the script before you use any SQL commands add this:

lr\_load\_dll("libmysql.dll");

Your LoadRunner script is now able to communicate with a MySQL database.

**Sample script which reads from MySQL**

This script assumes that you have already set up your MySQL Database. If not, go back to the start of this document and follow the instructions to create a database. First of all, you need to tell your script where MySQL is located. To do this, add the following to the top of the vuser\_init section:

#define MYSQLSERVER "127.0.0.1"

#define MYSQLUSERNAME "root"

#define MYSQLPASSWORD ""

#define MYSQLDB "test"

#define MYSQLPORT "3306"

Once you’ve done that, use the following syntax anywhere in the body of your script. In our case, we will put it in the main section of Action.c. So, add the following commands:

char chQuery[128];

MYSQL \*Mconn;

lr\_load\_dll("libmysql.dll");

Mconn = lr\_mysql\_connect(MYSQLSERVER, MYSQLUSERNAME, MYSQLPASSWORD, MYSQLDB, atoi(MYSQLPORT));

sprintf(chQuery, "select sqForename, sqSurname, sqJobDesc from sample\_reads");

lr\_mysql\_query(Mconn, chQuery);

lr\_save\_string(row[0][0].cell, "sForename");

lr\_save\_string(row[1][0].cell, "sSurname");

lr\_save\_string(row[2][0].cell, "sJobDesc");

lr\_output\_message(lr\_eval\_string("Forename: {sForename}; Surname: {sSurname}; Job Description:{sJobDesc}"));

lr\_save\_string(row[0][1].cell, "sForename");

lr\_save\_string(row[1][1].cell, "sSurname");

lr\_save\_string(row[2][1].cell, "sJobDesc");

lr\_output\_message(lr\_eval\_string("Forename: {sForename}; Surname: {sSurname}; Job Description:{sJobDesc}"));

lr\_mysql\_disconnect(Mconn);

**Structure of returned data**

If you run a select statement, you will be expecting data to be returned.

The data is returned in a multi-dimensional array and is referenced in your script by the function:

row[x][y].cell where x is the column position (first column is 0) and y is the row position (first row is 0).

Querying this table with the above script:

Machine generated alternative text:
sqForename sqSurname sqiobDesc
Marley Singer
Sammy Davies Entertainer
S bane Warne Cricketer
William Wales Prince
Redknapp Football Manager
Samuel Beckett Author

Writes this output:

Action.c(18): Forename: Bob; Surname: Marley; Job Description:Singer

Action.c(24): Forename: Sammy; Surname: Davies; Job Description:Entertainer

**Other SQL functions**

You can run any SQL query from the ‘lr\_mysql\_query’ function. MySQL allows complex queries to be written, for example returning data only where a certain criterion is met, for example using data not already used by other scripts. Columns can be added to flag data as “in use” or “used”. When compared with VTS it is significantly more powerful since MySQL takes the burden of searching through data to find data which matches the search criteria, rather than iteratively searching through data one line at a time in LoadRunner using C code.

**Sample script which writes to MySQL**

It can also be beneficial to write data to tables during performance tests. For example as data is “used” it can be flagged as such so that other vUsers don’t use the data. Response time information together with the details of queries can be written to a database for more detailed response time analysis than is possible using the standard LoadRunner Analysis tool.

First create a table in the test database to contain our output. To do this, paste the following SQL commands into the query area and run them sequentially.

[**DROP**](http://search.mysql.com/search?site=refman-%35%31&q=DROP) [**TABLE**](http://search.mysql.com/search?site=refman-%35%31&q=TABLE) [IF](http://dev.mysql.com/doc/refman/5.1/en/control-flow-functions.html) [**EXISTS**](http://search.mysql.com/search?site=refman-%35%31&q=EXISTS) `test`.`sample**\_**writes`;  
[**CREATE**](http://search.mysql.com/search?site=refman-%35%31&q=CREATE) [**TABLE**](http://search.mysql.com/search?site=refman-%35%31&q=TABLE) `test`.`sample**\_**writes` (  
`sqTransactionName` [**varchar**](http://search.mysql.com/search?site=refman-%35%31&q=VARCHAR)(128) [**DEFAULT**](http://search.mysql.com/search?site=refman-%35%31&q=DEFAULT) [**NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NULL),  
`sqResult` [**varchar**](http://search.mysql.com/search?site=refman-%35%31&q=VARCHAR)(128) [**DEFAULT**](http://search.mysql.com/search?site=refman-%35%31&q=DEFAULT) [**NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NULL),  
`sqResponseTime` [**varchar**](http://search.mysql.com/search?site=refman-%35%31&q=VARCHAR)(128) [**DEFAULT**](http://search.mysql.com/search?site=refman-%35%31&q=DEFAULT) [**NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NULL),  
`sqTime` [**varchar**](http://search.mysql.com/search?site=refman-%35%31&q=VARCHAR)(128) [**DEFAULT**](http://search.mysql.com/search?site=refman-%35%31&q=DEFAULT) [**NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NULL)  
);

This LR code

sprintf(chQuery, "insert into test.sample\_writes (sqTransactionName, sqResult, sqResponseTime, sqTime) values ('%s','%s','%s','%s');",

lr\_eval\_string("{sTransactionName}"),

lr\_eval\_string("{sResult}"),

lr\_eval\_string("{sResponseTime}"),

lr\_eval\_string("{sTime}"));

lr\_mysql\_query(Mconn, chQuery);

lr\_mysql\_disconnect(Mconn);

Writes the values for the parameters/strings sTransactionName, sResult, sResponseTime and sTime into the database. The example below shows 10 iterations of this script where transaction names, results and response times were selected randomly from a parameter list and sqTime was determined using the lr\_save\_datetime function.

Populates the table with values similar to this:

Machine generated alternative text:
sqTransactionName sqResult sqResponseTime sqTime
Slow Pass 20 11:02:29
Fast Fail 74 11:02:29
Fast Pass 21 11:02:29
Fast Pass 21 11:02:29
Fast Pass 55 11:02:29
NotFound Pass 91 11:02:29
NotFound Pass 4 11:02:29
NotFound Fail 20 11:02:29
Slow Pass 100 11:02:29
Fast Pass 97 11:02:29