

# How to get kdb+ to talk with other databases:

John Ludlow

February 24, 2011

## 1 Introduction

In many of the installations where kdb+ is found, data is distributed across a number of different databases, including legacy applications. As such, it is essential for kdb+ to be able to communicate effectively with other databases. This poses a problem however, as the different DB vendors provide different interfaces to their products. One simple method of exchanging data in a portable way is via the intermediate step of importing and exporting csv or excel files. However, it is usually more elegant and more efficient if kdb+ can communicate directly with the other databases. Fortunately, a solution has been created in Open Database Connectivity (ODBC). ODBC provides an intermediary software layer between different databases enabling a portable interface for exchanging data. See:

[http://en.wikipedia.org/wiki/Open\\_Database\\_Connectivity](http://en.wikipedia.org/wiki/Open_Database_Connectivity)

For the purpose of illustrating the use of the ODBC interface, the freely available MySQL and Microsoft SQL Server Express databases will be used as examples.

The work presented in this report was carried out on a PC running Windows 7, with Ubuntu 10.10 installed under VirtualBox.

## 2 Connecting kdb+ and MySQL on Ubuntu

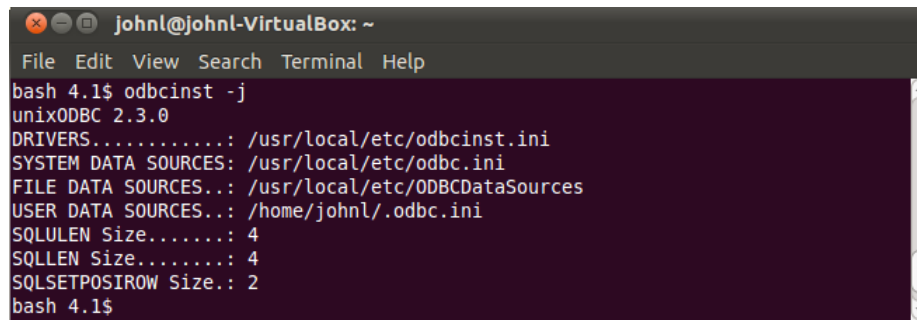
1. Download and install unixODBC from:

<http://www.unixodbc.org/>

2. Download and install MySQL:

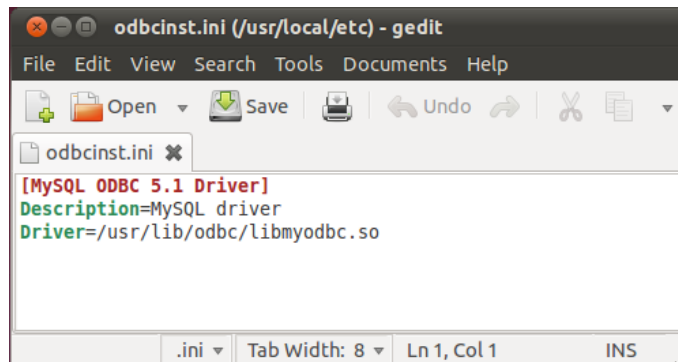
<http://dev.mysql.com/downloads/mysql/>

3. Download the sample world database:  
<http://dev.mysql.com/doc/world-setup/en/world-setup.html>
4. Install the MySQL driver:  
*sudo apt-get install libmyodbc*
5. Download odbc.k from:  
<http://kx.com/q/c/odbc.k>  
Place odbc.k in the QHOME directory
6. Download the odbc shared object (in this case odbc.so for l32) into the QHOME/l32 directory. See:  
<https://code.kx.com/trac/wiki/Cookbook/ODBC/qclient>
7. Next we want to add the driver to odbc. First output the odbc configuration:  
*odbcinst -j*



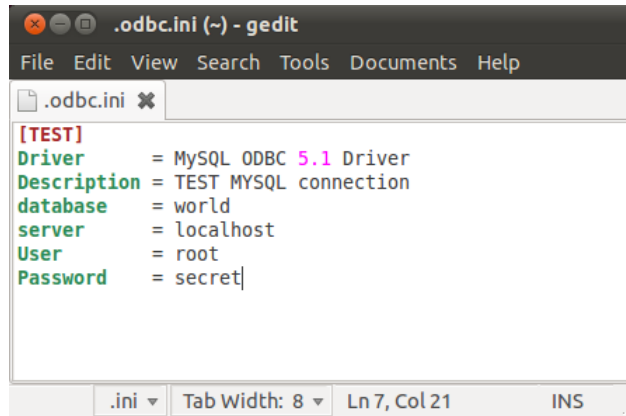
```
johnl@johnl-VirtualBox: ~  
File Edit View Search Terminal Help  
bash 4.1$ odbcinst -j  
unixODBC 2.3.0  
DRIVERS.....: /usr/local/etc/odbcinst.ini  
SYSTEM DATA SOURCES: /usr/local/etc/odbc.ini  
FILE DATA SOURCES..: /usr/local/etc/ODBCDataSources  
USER DATA SOURCES..: /home/johnl/.odbc.ini  
SQLULEN Size.....: 4  
SQLLEN Size.....: 4  
SQLSETPOSIROW Size.: 2  
bash 4.1$
```

8. Edit the /usr/local/etc/odbcinst.ini file to add the driver:



```
odbcinst.ini (/usr/local/etc) - gedit  
File Edit View Search Tools Documents Help  
Open Save Undo  
odbcinst.ini x  
[MySQL ODBC 5.1 Driver]  
Description=MySQL driver  
Driver=/usr/lib/odbc/libmyodbc.so  
.ini Tab Width: 8 Ln 1, Col 1 INS
```

9. Now we need to define a user DSN for this driver. A data source name (DSN) for a particular user contains the information required by the ODBC driver to connect to a particular database. Edit the `.odbc.ini` file in your `$HOME` directory to define a user DSN for this driver (in this case called TEST):

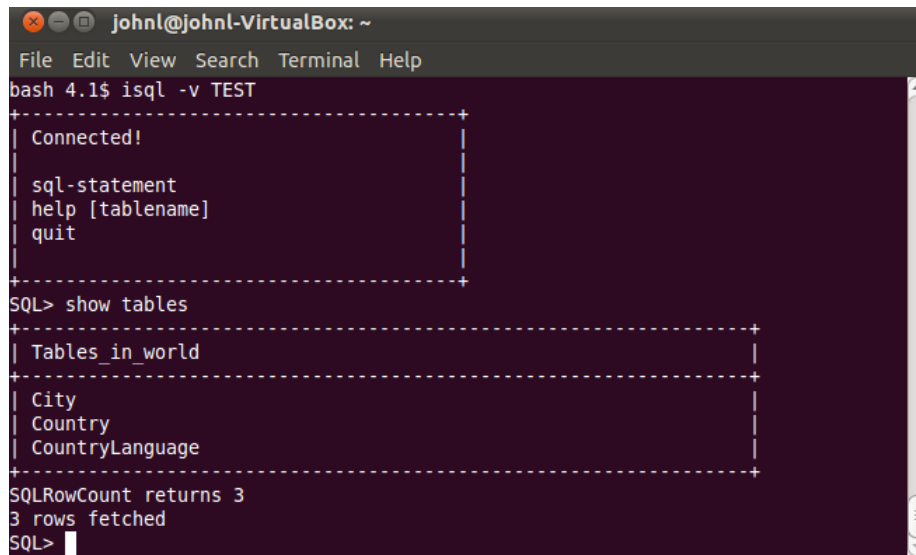


```
.odbc.ini (~) - gedit
File Edit View Search Tools Documents Help

.odbc.ini x
[TEST]
Driver      = MySQL ODBC 5.1 Driver
Description = TEST MYSQL connection
database    = world
server      = localhost
User        = root
Password    = secret|

.ini Tab Width: 8 Ln 7, Col 21 INS
```

10. Now we test the connection to the MySQL database using `isql`, a command line utility that comes with `unixODBC`. We start `isql` by supplying the userDSN:  
*isql -v 'userDSN'*  
Now we can enter MySQL commands into `isql`. For example:



```
johnl@johnl-VirtualBox: ~  
File Edit View Search Terminal Help  
bash 4.1$ isql -v TEST  
+-----+  
| Connected!  
|  
| sql-statement  
| help [tablename]  
| quit  
|  
+-----+  
SQL> show tables  
+-----+  
| Tables_in_world  
+-----+  
| City  
| Country  
| CountryLanguage  
+-----+  
SQLRowCount returns 3  
3 rows fetched  
SQL> |
```

11. Now the connection is working, we can also connect with `q` as the ODBC client. One straightforward way to do this is via a ‘DSNless’ connection. Create a file `connect.q` in your `QHOME` directory and load this into `q`. The file `connect.q` contains:

```
\l odbc.k  
h:.odbc.open "driver=MySQL ODBC 5.1 Driver;server=localhost;  
uid=username; pwd=password;database=world"
```

12. This defines a handle `h` to the MySQL world database. On a successful connection, `h` should be a long integer. Start `q` with `connect.q` and try the commands:

```
q) .odbc.tables [h]  
lists the tables in the database  
q) .odbc.eval[[h];"select * from City"]  
runs the MySQL command that is between the quotes
```

```
johnl@johnl-VirtualBox: ~  
File Edit View Search Terminal Help  
bash 4.1$ q connect.q  
KDB+ 2.7 2010.11.30 Copyright (C) 1993-2010 Kx Systems  
l32/ 1()core 496MB johnl johnl-virtualbox 10.0.2.15 PLAY 2011.02.28  
  
q).odbc.tables [h]  
`City`Country`CountryLanguage  
q).odbc.eval[[h];"select count(*) from City"]  
count(*)  
-----  
4079  
q)
```

13. We can also connect to MySQL from q using the TEST DSN that we have defined:

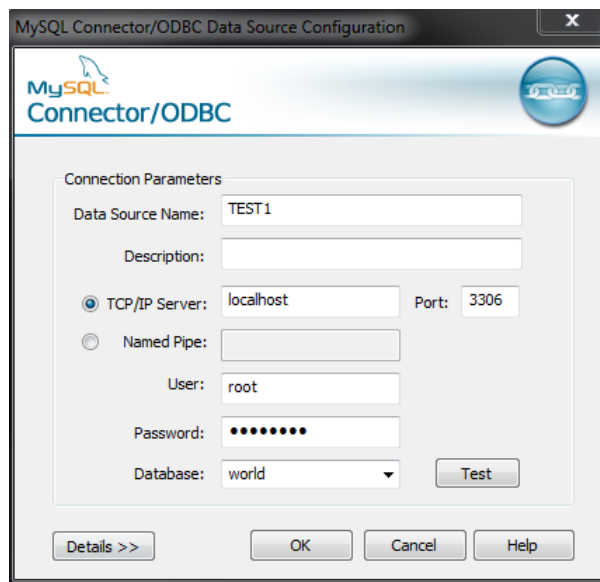
```
johnl@johnl-VirtualBox: ~  
File Edit View Search Terminal Help  
bash 4.1$ q  
KDB+ 2.7 2010.11.30 Copyright (C) 1993-2010 Kx Systems  
l32/ 1()core 496MB johnl johnl-virtualbox 10.0.2.15 PLAY 2011.02.28  
  
q)\l odbc.k  
q)h:.odbc.open `TEST  
q).odbc.tables [h]  
`City`Country`CountryLanguage  
q).odbc.eval[[h];"select count(*) from City"]  
count(*)  
-----  
4079  
q)
```

### 3 Connecting kdb+ and MySQL on Windows

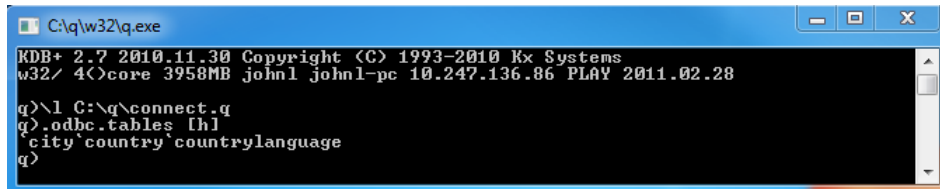
On Windows, connecting kdb+ and MySQL follows a similar process to that for linux with a few differences.

1. Download and install MySQL, the sample world database and the windows MySQL connector
2. Place the odbc.k file and the appropriate odbc.dll (32 bit in this case) into your C:\q\w32 directory
3. Make sure that you have the MSVCR71.dll which is the Microsoft Visual C Runtime library on your system. If it is missing, download it and place it in your C:\q\w32 directory

4. Note that the more familiar route to the OBCD would be via Start → Control Panel → Use the drop-down-menu → Click on "All Control Panel Items" → Administrative Tools → right click on ODBC . However the driver list is not displayed correctly. Therefore to define the DSN and load the driver, use C:\Windows\SysWOW64\odbcad32.exe Click configure and enter the details for the connection as shown below. Click test to confirm that the connection is working:



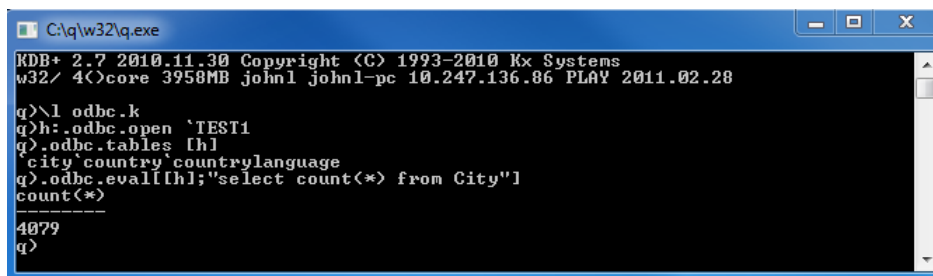
5. Now define your connect.q file and load it into q as before, where connect.q contains:
- ```
\l C:\q\w32\odbc.k
h::odbc.open "driver=MySQL ODBC 5.1 Driver;server=localhost;
uid=username;pwd=password;database=world"
```
6. As for kdb+ under Ubuntu, check that you can access the MySQL database from a q session using the 'DSNless' connection by loading connect.q:



```
C:\q\w32\q.exe
KDB+ 2.7 2010.11.30 Copyright (C) 1993-2010 Kx Systems
w32/ 4<core 3958MB john1 john1-pc 10.247.136.86 PLAY 2011.02.28

q>\l C:\q\connect.q
q>.odbc.tables [h]
'city'country'countrylanguage
q>
```

or by using the TEST1 DSN that we have defined:

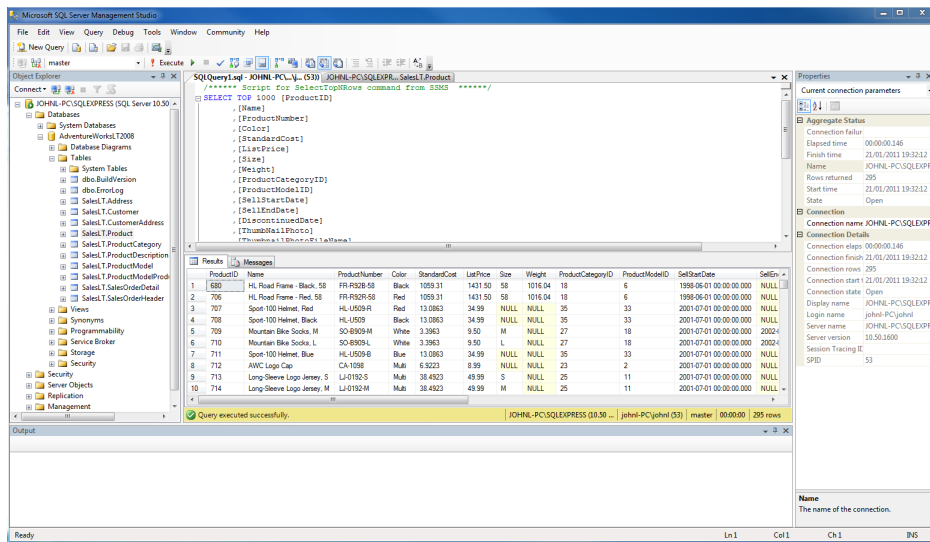


```
C:\q\w32\q.exe
KDB+ 2.7 2010.11.30 Copyright (C) 1993-2010 Kx Systems
w32/ 4<core 3958MB john1 john1-pc 10.247.136.86 PLAY 2011.02.28

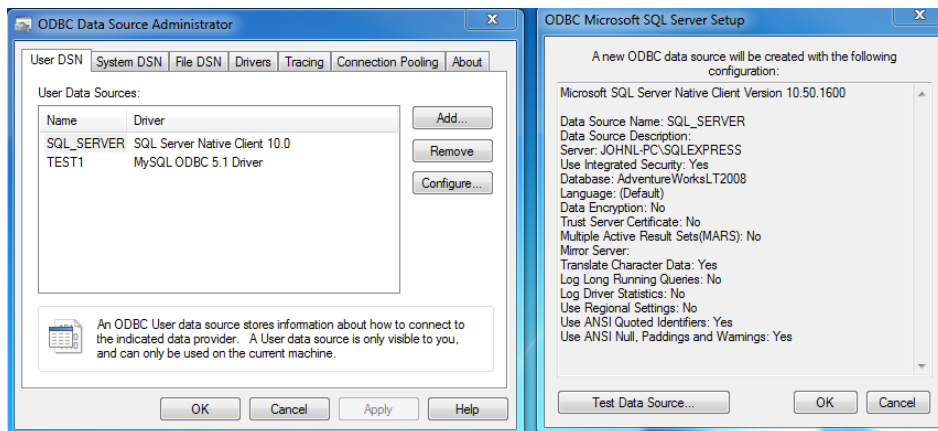
q>\l odbc.k
q>h:.odbc.open 'TEST1
q>.odbc.tables [h]
'city'country'countrylanguage
q>.odbc.eval[h]:"select count(*) from City"
count(*)
-----
4079
q>
```

## 4 Connecting kdb+ and Microsoft SQL Server Express

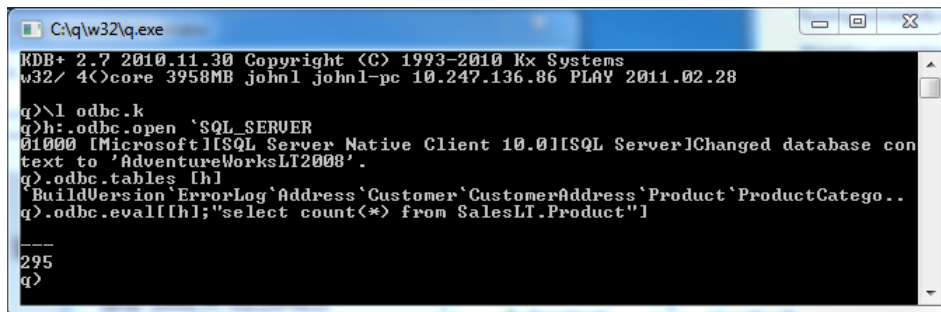
1. Download SQL Server Express from:  
<http://www.microsoft.com/express/Database/>
2. Download the smaller database AdventureWorksLT2008\_Data.mdf available at:  
<http://msftdbprodsamples.codeplex.com/releases/view/37109>  
and place it in the folder:  
C:\Program Files (x86)\Microsoft SQL Server\MSSQL10\_50.SQLEXPRESS\MSSQL\DATA
3. Load the sample database into SQL server via the object explorer window. Right click on databases, then attach the database:



4. Following the same procedure that was used for the MySQL database, define a userDSN using the Windows ODBC Data Source Administrator at C:\Windows\SysWOW64\odbcad32.exe



5. Now from q, load `odbc.k` and open the connection to the SQL Server using the userDSN that you have defined. Note that although table names are listed using the shorter form of the table name (for example the table *Product*), when using `.odbc.eval` with the SQL Server you have to give the fully qualified name of the table, in this case *SalesLT.Product*:



```
C:\q\w32\q.exe
KDB+ 2.7 2010.11.30 Copyright (C) 1993-2010 Kx Systems
w32/ 4< core 3958MB john1 john1-pc 10.247.136.86 PLAY 2011.02.28

q>\l odbc.k
q>h:.odbc.open `SQL_SERVER
01000 [Microsoft][SQL Server Native Client 10.0][SQL Server]Changed database con
text to 'AdventureWorksLT2008'.
q>.odbc.tables [h]
'BuildVersion' 'ErrorLog' 'Address' 'Customer' 'CustomerAddress' 'Product' 'ProductCatego..
q>.odbc.evalf[h];"select count(*) from SalesLT.Product"]

-----
295
q>
```

## 5 Comments

More detailed information on the ODBC interface to kdb+ can be found on the kx website, see:

- <http://kx.com/q/c/odbc.k>
- <http://kx.com/q/c/readme.txt>
- <https://code.kx.com/trac/wiki/Cookbook/ODBC>
- <https://code.kx.com/trac/wiki/Cookbook/ODBC/qclient>
- section 26 of: <http://www.kx.com/q/d/kdb+.htm>
- In addition, searching on the k4 listbox returns useful information.

A further extension is for kdb+ to act as server to ODBC rather than a client. Currently this is available for windows. See, <https://code.kx.com/trac/wiki/Cookbook/ODBC/qserver>