Team Developer and MySQL

Unify Corporation
By Jean-Marc Gemperle

Table of Contents

Introduction

Connecting to either SQLBase or Oracle on Microsoft Windows, or Linux from Team Developer can be achieved with the use of Unify or Oracle native routers.

However, connecting to other Relational Database Management Systems (RDBMS) must be done using ODBC. All other database systems have native Linux clients and there are third-party Linux/ODBC drivers for all of these RDBMS'.

This paper will show you how to set up your Windows and Linux ODBC environment for connectivity from Team Developer 2005.1 to MySQL 4.1. Team Developer 2005 did not offer an ODBC native router under Linux, and thus you can connect only to other database brands using the Windows ODBC router.

We will assume here for the purpose of this paper that you have a running MySQL 4.1 engine on Linux. Most distributions include the MySQL engine and client along with its ODBC driver.

Of course you can have the engine and client setup on Windows, but here we will show a Windows client connecting to a MySQL server on Linux and the local client on Linux.

Troubleshooting connectivity with ODBC on Windows and Linux will also be discussed.

Setting Up MySQL ODBC on Linux Using unixODBC

Install the ODBC driver for MySQL from the ODBC Red Hat Package Manager (RPM) installed under /usr/lib/.
shell> su root
shell> rpm –ivh MyODBC-3.51.01.i386-1.rpm
shell> export ODBCINI=/usr/local/etc/odbc.ini

Go to /usr/local/lib and create the following link files if they don't already exist:

ln –f-s libodbc.so.1.0.0 libodbc.so.1
ln –f-s libodbc.so.1 libodbc.so
ln –f-s libodbcinst.so.1.0.0 libodbcinst.so.1
ln –f-s libodbcinst.so.1 libodbcinst.so

Creating odbc.ini DSN

[ODBC Data Sources]
MySQL=MyODBC 3.51 Driver DSN

[MySQL]

Driver = /usr/lib/libmyodbc3.so
Description = MyODBC 3.51 Driver DSN
Server = localhost
Port = 3306
User = root
Password =
Socket =
Database = Unify

OPTION = 3

Creating odbcinst.ini DSN

[ODBC Drivers]
mysql-driver = Installed

[mysql-driver]
Driver= /usr/lib/libmyodbc3.so

Copy the contents of .odbc.ini to the odbc.ini in the /etc/unixodbc directory. Also, /usr/local/etc can be another place to copy these files if, for example, you recompiled unixODBC with its defaults.

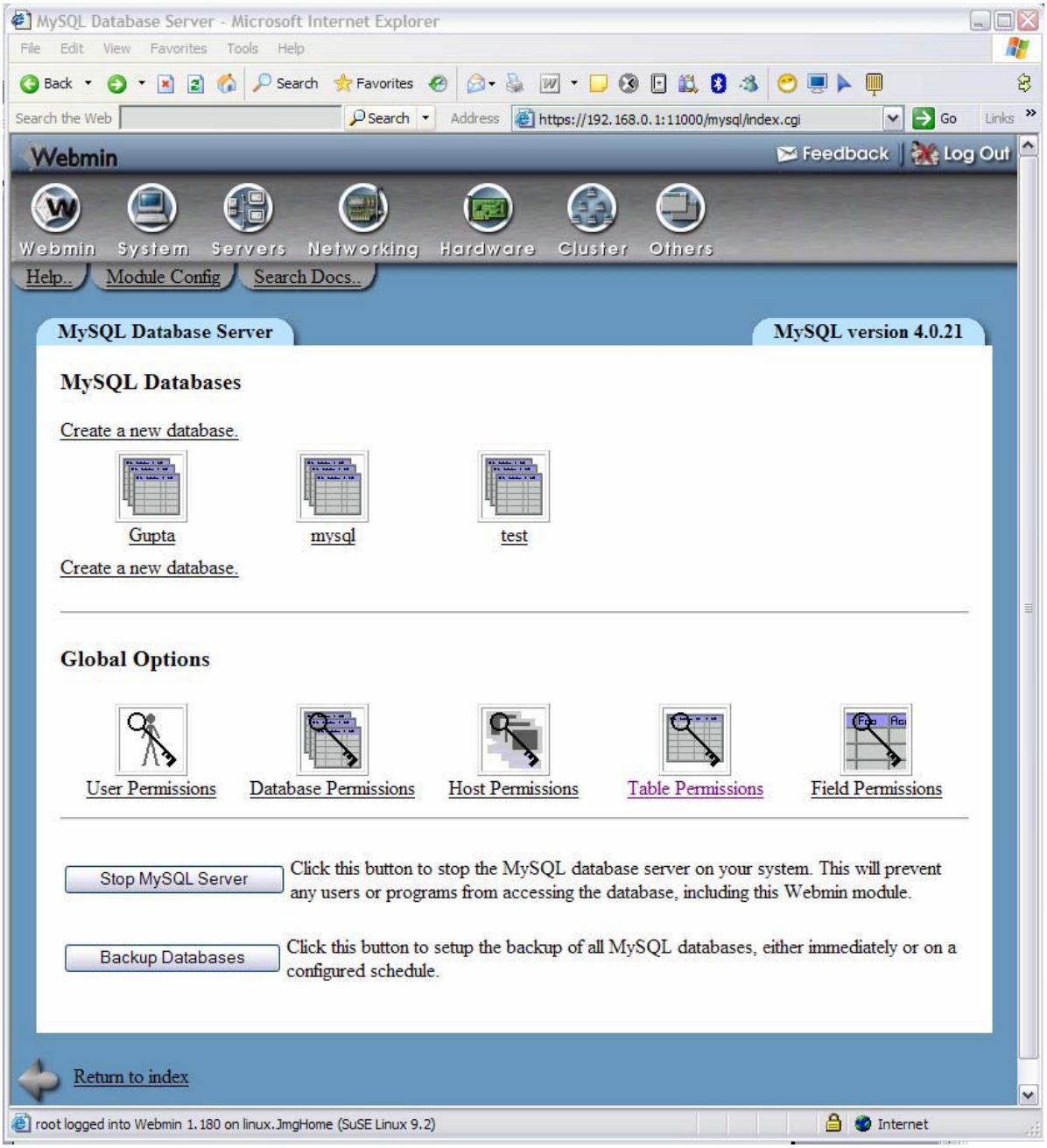
```
linux:/ # mysqladmin -p Create GUPTA
```

Now we want to create a *sysadm* user and grant access to this user both locally and remotely for our Win32 machine.

```
linux:/ # mysql -p mysql
Enter password:
Welcome to the MySQL monitor. Commands end with; or \g.
Your MySQL connection id is 142 to server version: 4.0.21
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> grant all privileges on *.* to 'SYSADM'@'192.168.0.%' identified by
'SYSADM';
Query OK, 0 rows affected (0.01 sec)
mysql> grant all privileges on *.* to 'SYSADM'@'localhost' identified by
'SYSADM';
Query OK, 0 rows affected (0.00 sec)
mysql>
linux:/ # mysql -p mysql
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 171 to server version: 4.0.21
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> show tables;
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv |
| db |
| func |
| host |
| tables_priv |
| user |
+-----------------+
6 rows in set (0.00 sec)
mysql> select host,user from user;
+---------------+------------+
| host | user |
+---------------+------------+
| 192.168.0.% | SYSADM |
| 192.168.0.10 | jmgemperle |
| linux.JmgHome | |
| linux.JmgHome | jmgemperle |
| linux.JmgHome | root |
| localhost | |
| localhost | jmgemperle |
| localhost | root |
| localhost | SYSADM |
```

```
+---------------+------------+
9 rows in set (0.00 sec)
mysql>
```

If you are not familiar with MySQL there is a Linux administration tool called Webmin that contains many modules you might like to consider using. The MySQL module allows you to create users, databases, etc. A screenshot of the Webmin user console appears below.



Testing unixODBC Connection with isql

We can test our unixODBC connectivity with the SYSADM user using the *isql* tool as shown below:

```
linux:/ # isql MySQL SYSADM SYSADM
+---------------------------------------+
| Connected! |
| |
| sql-statement |
| help [tablename] |
| quit |
| |
+---------------------------------------+
SQL> q
linux:/ #
```

Setting up MySQL ODBC on MS Windows

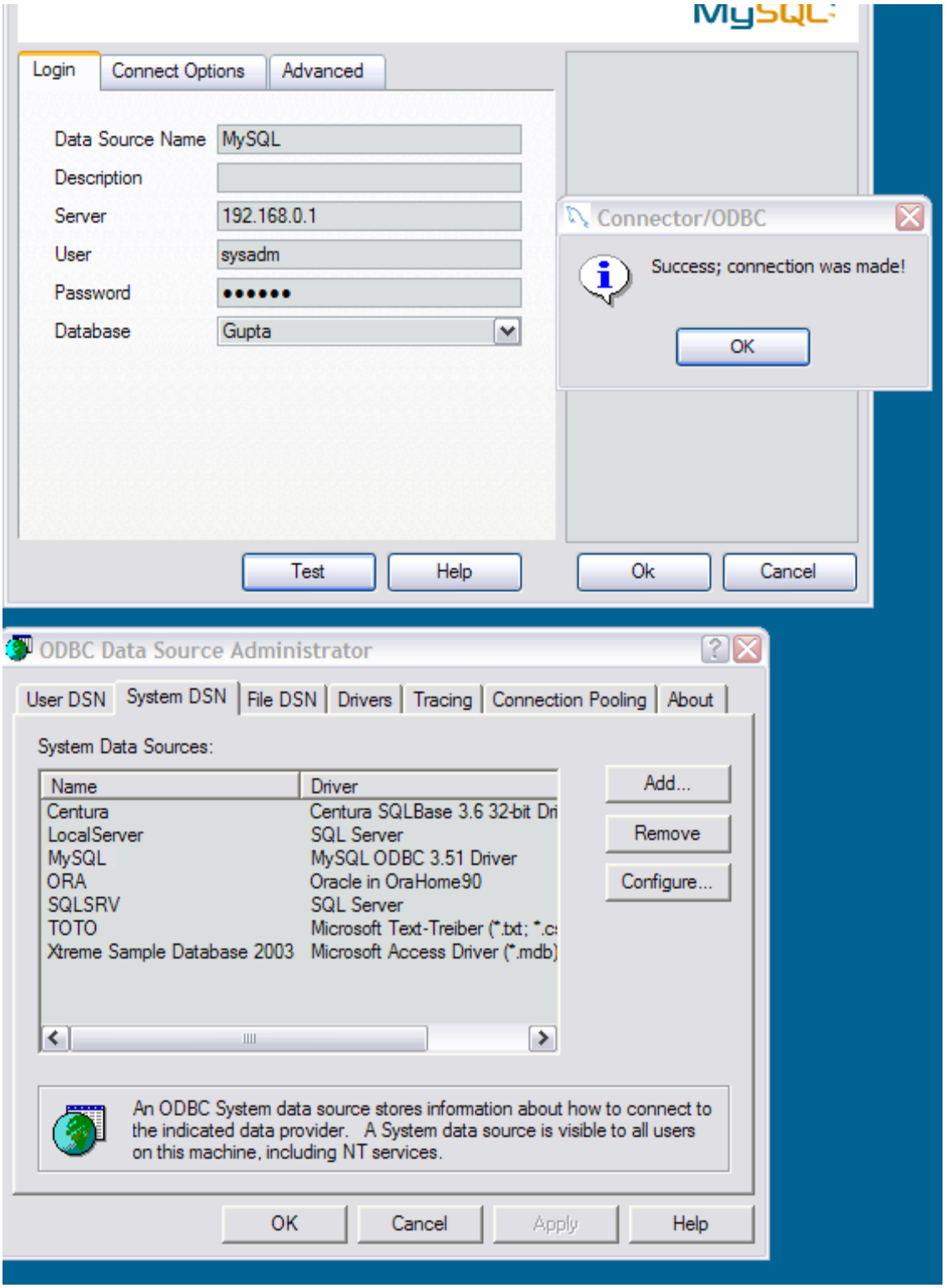
Setup

Simply fetch the WIN32 ODBC driver *Connector/ODBC 3.51* from http://dev.mysql.com/downloads/ and execute the setup.

Creating a MySQL Data Source

In the Windows Start menu, go to **Control Panel** > **Administrative Tools** and select *Data Sources (ODBC)*. On the System DSN tab create a MySQL data source name and test the connectivity. It is important that the user connecting to the database is allowed to connect to the MySQL host (see previous steps for creating of users).

Below is a screenshot of the ODBC Data Source Administrator connection.


Login
Connect Options
Advanced
Data Source Name
MySQL
Description
Server
192.168.0.1
User
sysadm
Password
Database
Gupta
Test
Help
Connector/ODBC
Success; connection was made!
OK
Ok
Cancel
ODBC Data Source Administrator
User DSN
System DSN
File DSN
Drivers
Tracing
Connection Pooling
About
System Data Sources:
Name
Driver
Centura
Centura SQLBase 3.6 32-bit Dri
LocalServer
SQL Server
MySQL
MySQL ODBC 3.51 Driver
ORA
Oracle in OraHome90
SQLSRV
SQL Server
TOTO
Microsoft Text-Treiber (*.txt; *.c
Xtreme Sample Database 2003
Microsoft Access Driver (*.mdb)
Add...
Remove
Configure...
An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.
OK
Cancel
Apply
Help

Team Developer Connection

Linux SQL.INI entry

```
[linuxclient]
clientname=sql-linux
clientruntimedir="/opt/tdx41/support/dotwine/fake_windows/DEV EL/TD401"
[linuxclient.dll]
comdll=sqlodb32
[odbcrtr]
odbctrace=off
odbctracefile=sql.log
remotedbname=MySQL,dsn=MySQL
```

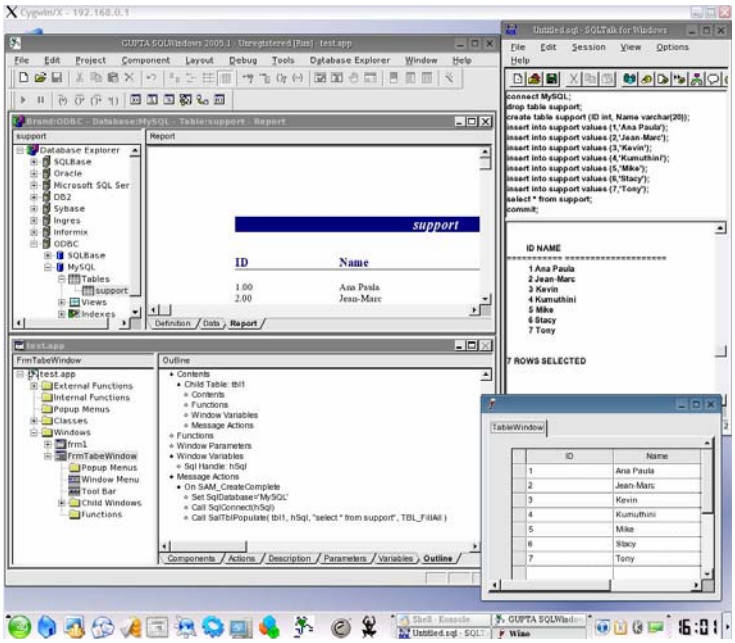
Windows SQL.INI entry

```
[win32client]
clientname=intlp4
clientruntimedir="C:\Devel\TD401"
[win32client.dll]
comdll=sqlodb32
[odbcrtr]
odbctrace=off
odbctracefile=sql.log
remotedbname=MySQL, DSN=MySQL
```

Connecting to MySQL

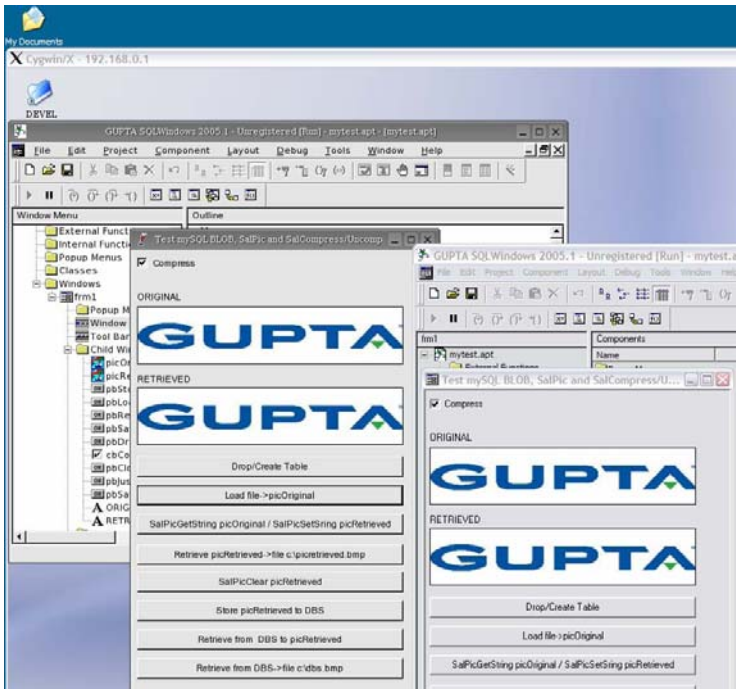
Connection can be done using SQLTalk, Database Explorer, Report Builder or a simple Team Developer application.

The screenshot below shows UNIFY on Linux connected locally to MySQL. Of course the same screenshot can also show UNIFY running on Windows and connected locally to MySQL.



Long Binary Data Insertion and Retrieval Sample

This simple example shows the insertion and retrieval of a *longblob* data type from MySQL by inserting a small BMP file. Of course the sample works both on Linux and Windows. If you attempt to insert a bigger BMP file you may receive the error: *MySQL: 1153 Got a packet bigger than 'max_allowed_packet*'.



A Simple Connection To Your MySQL Data Source

```
◆ Form Window: frm1
  ◇ Description:
  ◇ Named Menus
  ◇ Menu
  ◆ Tool Bar
  ◆ Contents
  ◇ Functions
  ◇ Window Parameters
  ◆ Window Variables
    ◇ Sql Handle: hSql
    ◇ Long String: sBuffer
    ◇ Number: nRetVal
    ◇ File Handle: fh
    ◇ String: saFileFilters[*]
    ◇ String: sChosenFile
    ◇ String: sChosenPath
    ◇ Number: nWhichFilter
    ◇ Number: nNumFilters
  ◆ Message Actions
    ◆ On SAM_CreateComplete
      ◇ ! Connection to mySQL, SYADM/SYSADM exist as user and password
      ◇ Set SqlDatabase = "MySQL"
      ◇ Call SalWaitCursor( TRUE )
      ◆ If SqlConnect( hSql)
        ◇ ! We set our DBP_LONGBUFFER
        ◆ If Not SqlSetParameterAll (hSql,DBP_LONGBUFFER,1000000,'',TRUE)
      ◇ Call SalWaitCursor( FALSE )
```

Creation of the Table

```
◆ Pushbutton: pbDropCreate
  ◆ Message Actions
    ◆ On SAM_Click
      ◆ When SqlError
      ◇ Call SqlPrepareAndExecute(hSql,"drop table imagetest")
      ◇ Call SqlPrepareAndExecute(hSql,"create table imagetest(pic longblob, num integer)")
```

Insertion in the Database

```
◆ Pushbutton: pbStoreToDBS
  ◆ Message Actions
    ◆ On SAM_Click
      ◇ Set sBuffer=""
      ◇ Call SalPicGetString( picOriginal, PIC_FormatBitmap, sBuffer )
      ◇ Call SalWaitCursor( TRUE )
      ◆ If cbCompress
        ◇ Call SalMessageBox("about to compress...", "error",MB_Ok)
        ◆ If NOT  SalStrCompress( sBuffer )
          ◇ Call SalMessageBox("SalStrCompress", "error",MB_Ok)
        ◆ Else
          ◇ ! This is needed...see SalStrCompress() HLP
          ◇ Call SalStrSetBufferLength( sBuffer, SalStrGetBufferLength( sBuffer ) + 1 )
      ◇ ! Our fist column is of type long binary...
      ◇ Call SqlSetLongBindDatatype(1,23)
      ◇ Call SqlPrepareAndExecute(hSql, '       INSERT
                                                INTO imagetest( pic, num)
                                                VALUES (:sBuffer, 1)')
      ◇ Call SqlCommit(hSql)
      ◇ Call SalWaitCursor( FALSE )
```

Retrieval from the Database

```
◆ Pushbutton: pbRetrieveFromDBS
  ◆ Message Actions
    ◆ On SAM_Click
      ◇ Call SalWaitCursor( TRUE )
      ◇ Set sBuffer=''
      ◇ Call SqlPrepareAndExecute( hSql, '      SELECT pic
                                                INTO :sBuffer
                                                FROM imagetest
                                                WHERE num = 1')
      ◇ Call SqlFetchNext(hSql,nRetVal)
      ◇ Call SqlCommit(hSql)
      ◇ Call SalWaitCursor( FALSE )
      ◆ If cbCompress
        ◇ Call SalMessageBox(''about to uncompress...'', ''error'',MB_Ok)
        ◆ If NOT  SalStrUncompress(sBuffer )
          ◇ Call SalMessageBox(''SalStrUnCompress'', ''error'',MB_Ok)
      ◇ Call SalPicSetString( picRetrieved,  PIC_FormatBitmap, sBuffer)
```

Troubleshooting

If you experience any problems, try some of these suggestions:

- Always try to install unixODBC from the RPM file and ***not*** from source.
- Ensure that one copy of libodbc.so/libodbc.so.1 is present. It can be found in /usr/local/lib or /usr/lib. Preferably, have this in /usr/local/lib/ and remove the files from /usr/lib/.
- Ensure that the LD_LIBRARY_PATH is updated with the /usr/local/lib/ path.
- Ensure that the ODBCINI is updated with /etc/odbc.ini.
- Ensure that the ODBCSYSINI is updated with /etc/.
- Check whether you are able to connect to the database with isql.
- Check the links under /usr/local/lib/ or /usr/lib/. For UnixODBC libodbc.so.1 should link to libodbc.so.1.0.
- Check the entry for comdll=sqlodb32 in the sql.ini under the *linuxclient* section, or under the win32client if MS Windows is concerned.

If you can connect to MySQL via ODBC using a native tool like isql, and if you can also connect to MySQL using SQLTalk but connectivity fails both with a Team Developer application or Database Explorer/Report Builder, then check the file which is found in the Team Developer root installation directory. Delete it and attempt the connection again.

About Unify

Unify is a global provider of software development technology and solutions that helps IT customers participate in Service-Oriented Architecture (SOA). Unify's productive and re-liable development tools, migration solutions and databases enable organizations to build and modernize business essential applications for SOA. Composer for Lotus Notes offers a complete, like for like, production to production migration solution for Lotus Notes applications. Unify's award-winning NXJ Developer enables IT teams to be extremely productive, learn new technologies fast and deliver Web services-based applications on time and on budget. The Team Developer, SQLBase, DataServer, ACCELL and VISION product families enable cross-platform rapid development on Java/J2EE, Linux or Windows. Unify has a rich heritage in delivering rich, cost-effective technologies to its thousands of IT customers and ISV, VAR and distributor partners. Unify is headquartered in Sacramento, Calif., and can be reached at (916) 928-6400 or by visiting www.unify.com.

Unify Corporation
2101 Arena Blvd., Suite 100
Sacramento, CA 95834
USA
Phone: 1.916.928.6400
Toll Free: 1.800.468.6439
Fax: 1.916.928.6404
Munich: +49 8 115 55430
United Kingdom: +44 (0)1753 245 510
France: +33 (0)1 34 58 28 30