TimesTen Driver Manager

User Guide

Version: 1.9

Date: 15th July 2009

Introduction to the TimesTen Driver Manager

This is the User Guide for the TimesTen Driver Manager (TTDM). TTDM is a lightweight ODBC Driver Manager specifically developed and optimised for use with the Oracle TimesTen In-Memory Database.

What is a Driver Manager and why might I need one?

When an ODBC based application connects to a datasource (typically a database) it connects to a logical name, the Data Source Name (DSN), which identifies the datasource to which it wants to connect. It does this by calling various ODBC functions provided by an ODBC driver. Some external repository of configuration information holds the various DSN values that are available, together with the necessary configuration and control information needed by the ODBC driver to establish a connection and manage usage of the datasource.

An ODBC driver is a piece of software (typically a library of some kind) that implements the ODBC API and provides the functionality to connect to a specific kind of datasource. For example, there are ODBC drivers available for most popular databases; Oracle, SQL Server, Sybase, DB2 etc. etc. Each type of database (datasource) requires a different ODBC driver. Herein lies a potential problem…

Since each ODBC library defines and exposes (largely) the same set of functions (those defined in the ODBC API), an application can only be linked with one ODBC driver library at any one time. This means that if the application has to support different kinds of datasource, different versions of the application must be built for each type of datasource. If the application needs to connect concurrently to different types of datasource (i.e. needs to concurrently use more than one ODBC driver) then this becomes impossible.

A solution to this issue is to use a Driver Manager (DM). A DM itself implements and exposes the ODBC API , thus the application can link directly to the DM library instead of the individual ODBC Driver libraries. Based on some configuration data or other mechanism, the DM will dynamically load the relevant ODBC Driver libraries at runtime as the application requires them. The DM sits between the application and the individual ODBC drivers and arbitrates all ODBC calls. The application can now connect to multiple datasources, using different ODBC drivers, concurrently.

Advantages and disadvantages of using a Driver Manager?

The advantages of a DM are:

1. It enables an application process to concurrently use multiple, different ODBC drivers.

2. It can sometimes enable an application that expects a newer version of the ODBC API standard (e.g. 3.5) to work with a driver that implements an older version of the API (e.g. 2.5). Again, this is not guaranteed and depends on how the application has been written.

The disadvantages of a DM are:

1. Although there is a DM built into the Windows O/S, for Unix/Linux O/S there is no standard DM available. There are a few Open Source DMs and maybe even one or two commercial DMs (which require a license fee).

2. DMs, by their nature, are generic. They focus on providing the full diverse functionality of ODBC in order to cater for all possible ODBC drivers and their

capabilities. As a result, they typically impose a significant performance penalty; sometimes as much as 20% or more.

3. Generic DMs often do not provide access to special capabilities or optimisations offered by the underlying ODBC drivers.

Why is any of this relevant to Oracle TimesTen?

Oracle TimesTen is a high performance, relational, In-Memory Database (IMDB). Its native API is ODBC and its access language is SQL. TimesTen provides two connection mechanism for applications; direct mode and client/server. Here is a brief summary of the similarities and differences between them.

Direct mode

- The application and the TimesTen database are tightly coupled. They must both reside on the same physical computer system.
- There is no IPC involved in application <-> database communication resulting in reduced response times and increased throughput for database (SQL) operations.
- The API used is ODBC.
- There is a proprietary event notification and change tracking API called XLA available. This API is part of the direct mode ODBC driver library.
- There is a proprietary API to various administrative and utility functions. This is provided by a separate Utility Library.

Client/server

- The application and the TimesTen database are loosely coupled. They can reside on the same physical computer system or different computer systems.
- Database access is performed by a server proxy process on behalf of the client application. The server proxy always executes on the computer system where the TimesTen database is located.
- The communication between the application and the server proxy is via some form of IPC. This may be intra-system IPC such as Unix sockets or shared memory or it can be inter-system IPC – a TCP/IP socket. The server proxy itself is a direct mode application. As a result, the performance of client/server access is significantly less than that of direct mode access.
- The API used is ODBC.
- The XLA API is not available.
- The Utility Library is not available.

Although both connection mechanisms use the ODBC API, they are implemented as separate driver libraries. Thus, without a Driver Manager, a TimesTen application can use one or the other, but not both concurrently.

So, if one wishes to use a commercial or Open Source DM with TimesTen in order to allow use of both direct mode and client/server connections concurrently from the same process, what problems or issues might you encounter?

1. The DM will almost certainly not have specific support for TimesTen so it might not work reliably with TimesTen.

2. The DM will almost certainly not support any TimesTen specific optimisations and due to its generic nature the performance overhead may be quite high.

3. When using the DM access to TimesTen, special features and function libraries (XLA, utility Library) are not available.

The TimesTen Driver Manager (TTDM) is a lightweight DM focussed specifically on overcoming these limitations. Its main features are:

- No application source code changes are needed to use TTDM (apart from optionally including the TTDM header file ttdrvmgr.h instead of the timesten.h header file).

- TTDM allows a single process to concurrently use both client/server and direct mode connections. TTDM dynamically determines the correct connection type based on the DSN that is being used for a connection.

- TTDM provides the same level of ODBC API support as TimesTen, including all TimesTen extensions.

- For direct mode connections, TTDM provides access to the full set of XLA functionality.

- For direct mode connections, TTDM provides access to the full set of TimesTen utility library functionality.

- TTDM has a fairly low performance overhead. See later for details.

- TTDM operates with all TimesTen installation configurations; Data Manager and Client/Server, Data Manager only and Client only without any special configuration. For example, in a client only installation, attempts to access a direct mode DSN will simply return an ODBC error.

Things that TTDM does not do are:

- Emulate higher levels of the ODBC API (e.g. 3.0 or 3.5).

- Provide/emulate ODBC functions not provided by TimesTen

- Support the use of non-TimesTen ODBC drivers

- Provide driver manager specific functions not provided by the TimesTen drivers.

TTDM Performance Overhead

The performance overhead of using TTDM has been measured in various configurations on a number of platforms. For all the platforms evaluated, the results were as follows.

For direct mode connections the overhead of TTDM typically ranges from 0% to 9% with 3%-5% being typical.

For remote client/server connections the overhead of TTDM typically ranges from 0% to 3% with 1% being typical.

TimesTen Version and O/S Support

This version of TTDM currently supports the following TimesTen versions:

TimesTen 11.2.1 - 32 and 64 bit

Building and Installing TTDM

TTDM is provided as source code and is automatically built when the ODBC and ttClasses Quick Start sample programs are built.

Building and Running TTDM Applications

Assuming you already have some application source code built and linked with TimesTen then there should be no code changes needed to build and link with TTDM instead.

The rest of this section assumes you have successfully built TTDM as per the previous section.

Source code changes

No source code changes are needed in order to use TTDM. If desired, you can change any source file that currently includes the main TimesTen header file, `timesten.h`, to include the TTDM header file, `ttdrvmgr.h`, instead. The TTDM header file itself includes `timesten.h`. Including `ttdrvmgr.h` gives access to symbolic names for some additional native error codes that may be returned by TTDM and to the constants used for the **SQLGetConnectOption()** extension. It is not mandatory to include this file, but if you do not then you cannot reference the additional error codes or SQLGetConnectOption() constants in your program.

If your application uses XLA then you should continue to include `tt_xla.h` in your code.

If your application uses the TimesTen Utility Library then you should continue to include `ttutillib.h` and `ttutil.h` as appropriate.

Linking changes

Currently you will likely be linking with one of the TimesTen ODBC libraries (`libtten.so` for direct mode or `libttclient.so` for client/server mode) and maybe also with the TimesTen Utility Library (`libttutil.so`) or possibly with some other driver manager library.

You should change the linker library options in your Makefile(s) to instead link just with the TTDM library (`libttdrvmgr.so` or `ttdrvmgr.lib`). For example, on Unix/Linux platforms the linker option required will be `-lttdrvmgr` instead of `-ltten` or `-lttclient` and `-lttutil`.

Runtime changes

At run time the TTDM shared library, plus all TimesTen shared libraries, must be located in directories defined in LD_LIBRARY_PATH (or its equivalent for your platform). On Windows the TTDM DLL (`ttdrvmgr.dll`) must be located in a directory that is part of your PATH.

If you have deployed TTDM as per the instructions in the previous section then no additional actions should be required.

TTDM ODBC Extensions

TTDM provides an extension to the ODBC **SQLGetConnectOption()** function. If you pass the value **TTDM_CONNECTION_TYPE** for the **fOption** parameter and a pointer to a SQLINTEGER for the **pvParam,** as follows:

```
SQLINTEGER   connType;


rc = SQLGetConnectOption(hdbc, TTDM_CONNECTION_TYPE, &connType);
```

then on successful return (`rc == SQL_SUCCESS`), `connType` will contain a value that indicates the type of connection represented by hdbc as follows:

TTDM_CONN_NONE - the hdbc is currently not connected

TTDM_CONN_DIRECT - the hdbc is connected in direct mode

TTDM_CONN_CLIENT - the hdbc is connected in client/server mode

Additional Errors & Warnings

In addition to the regular TimesTen native errors and warnings, as defined in the TimesTen Error Reference, TTDM can return the following additional native errors and warnings.

Errors

Error code mnemonic	Meaning

tt_ErrDMNoMemory	The driver manager was unable to allocate some required memory.
tt_ErrDMDriverLoad	The driver manager was unable to dynamically load a required library (direct mode driver, client driver or utility library).
tt_ErrDMNotDisconnected	An attempt was made to call SQLFreeConnect() on a connection that is still connected. Disconnect the connection first by calling SQLDisconnect().

Warnings

No additional warnings currently defined.