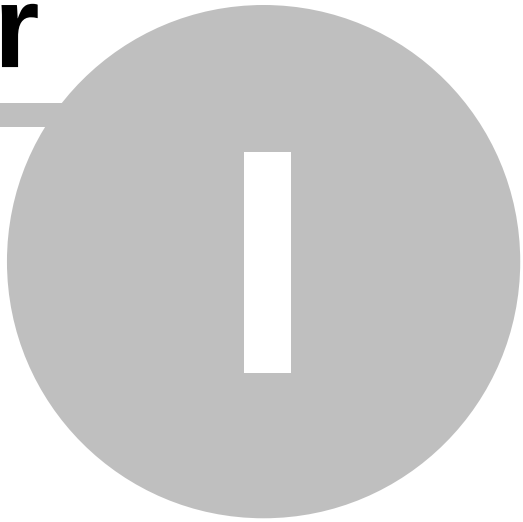
TMS Data Modeler Library Manual

Table of Contents

Chapter



Introduction

1 Introduction

1.1 Overview

TMS Data Modeler Library (DMLib) is a collection of classes that allows you to read the whole structure of a database (tables, fields, procedures, indexes, etc.) from a TMS Data Modeler (*) project file, using a Delphi / C++ Builder application. After modeling your database with TMS Data Modeler, you can use TMS Data Modeler Library classes to have the database structure information available in your Delphi application.

Using this classes you can make it easier to build your database-based applications. An example of how you can develop a database-application using TMS Data Modeler Library:

1. Model/design your database using TMS Data Modeler (*). A project file with the model will be generated
2. Create/update your database structure from TMS Data Modeler
3. Use TMS Data Modeler Library to read the database model information from the project file saved by the Data Modeler.
4. With step (3) above, you will be able to know which tables, fields and other database objects are available in your database
5. By deploying the project file with your application, you will always know the database objects available
6. Whenever you need to update/change your database structure, use TMS Data Modeler to udpate your model and generate SQL statements to update your database
7. Update your project file in your client application and you will have the most up-to-date information about your database structure, without needing to update/recompile your application.

This way you will have a centralized and unique place with all information about your database structure. From modeling, to generating, to your application, a single file holds all the information and you don't need to perform duplicated tasks like updating your application source code with the new fields and tables added in the last version.

By knowing the database structure at runtime, specially tables, fields, and field data types, you can easily perform some tasks in your application like for example:

- Provide report/query tools which lists the available tables and fields in database
- Generate dynamic forms to edit records in database based on the tables and fields available
- Create applications that automatically generate forms, web pages, source code, all based in the current database information
- Other tasks that need information about database structure

(*) TMS Data Modeler is a database modeling/design software tool made by TMS. You can design ER diagrams with it, generate DDL SQL Scripts, compare databases, archive/compare versions, among other tasks. More information at http://www.tmssoftware.com/site/tmsdm.asp.

1.2 Copyright Notice

TMS Data Modeler Library is free for use in non-commercial and comercial applications, as long as the applications are not competitor products of TMS Software (http://www.tmssoftware.com) products.

The component cannot be distributed in any other way except through free accessible Internet Web

pages or ftp servers. The component can only be distributed on CD-ROM or other media with written autorization of the author.

1.3 What's New

version 1.7 (Dez-2010)
new: TMS Data Modeler 1.7 (ElevateDB support).

version 1.6 (Nov-2010)
new: TMS Data Modeler 1.6 (SQL Azure support).

version 1.5 (Oct-2010)
new: TMS Data Modeler 1.5.1 updates.
new: Delphi 2010/XE support.

version 1.0 (Jun-2009)
new: first version released. Supports TMS Data Modeler 1.1 files.

1.4 Getting Support

General notes

Before contacting support:
- Make sure to read the tips, faq and readme.txt or install.txt files (when available) in component distributions.
- Make sure you have the latest version of the component(s).

When contacting support:
- Specify with which component you have a problem.
- Specify which Delphi or C++Builder version you're using and preferably also on which OS.
- Send email from an email account that
1) allows to receive replies sent from our server
2) allows to receive ZIP file attachements
3) has a properly specified & working reply address

Getting support
For general information: **info@tmssoftware.com**
Fax: +32-56-359696
For all questions, comments, problems and feature request for VCL components : **help@tmssoftware.com**.
To improve efficiency and speed of help, refer to the version of Delphi, C++Builder, Visual Studio .NET you are using as well as the version of the component. In case of problems, always try to use the latest version available first

1.5 Installation

No installation is needed for TMS Data Modeler Library, since all classes must be instantiated at runtime from source code. The only install procedure you might need is to register help files in to Delphi/C++ Builder environment.

Installing help files

- For Delphi 5,6,7 & C++Builder 6

  For Delphi 6,7 and C++Builder 6 : put file dmlibX.als in the {$DELPHI}\Help directory first
  For Delphi 7: put file dmlibx.als also in the {$DELPHI}\Projects directory

  Add the appropriate help file in the IDE. Choose menu option Help, Customize, Index tab

  dmlibd5.hlp : Delphi 5
  dmlibd6.hlp : Delphi 6
  dmlibd7.hlp : Delphi 7
  dmlibb6.hlp : C++Builder 6

- For Delphi 2005, 2006, 2007, 2009, 2010 & XE and C++Builder 2006, 2007, 2009, 2010 & XE

- Close the IDE
- Run Windows command line (Menu Start, Run, type "cmd" <Enter>)
- Change current directory to the directory you installed TMS DMLib, subfolder "help"
- Execute the command line:

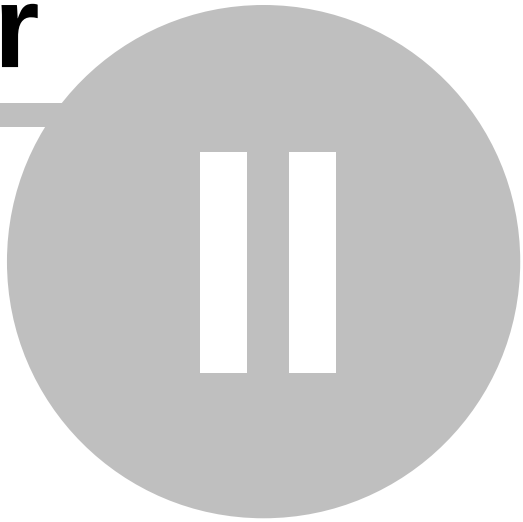
  RegHelp2 /i <hxx file name>

where <hxx file name> is the following according to your Delphi version:

Delphi 2005: dmlibdBDS3W.hxx
Delphi 2006: dmlibdBDS4W.hxx
Delphi 2007: dmlibdBDS5W.hxx
Delphi 2009: dmlibdRS2009W.hxx
Delphi 2010: dmlibdRS2010W.hxx
Delphi XE: dmlibdRSXEW.hxx
C++ Builder 2006: dmlibbBDS4W.hxx
C++ Builder 2007: dmlibbBDS5W.hxx
C++ Builder 2009: dmlibbRS2009W.hxx
C++ Builder 2010: dmlibbRS2010W.hxx
C++ Builder XE: dmlibbRSXEW.hxx

- Restart the IDE

Chapter



Using TMS Data Modeler Library

2 Using TMS Data Modeler Library

2.1 Getting started

Using TMS Data Modeler Library is simple: load the project file, and then use data dictionary properties to check the content. Here are the initial steps to load project file:

1. Add units uAppMetaData, uGDAO and dgConsts to your unit:

```
uses
  ...,
  uAppMetadata, uGDAO, dgConsts;
```

2. Load the project file using LoadFromFile method

```
var
  MyAMD: TAppMetaData;
begin
  MyAMD := TAppMetaData.LoadFromFile('c:\myproject.dgp');
  ...
```

3. Read all information using TAppMetaData properties. Main property is DataDictionary which holds subproperties like Tables, Fields, etc.:

```
  for c := 0 to MyAMD.DataDictionary.Tables.Count - 1 do
    Memo1.Lines.Add(MyAMD.DataDictionary.Tables[c].TableName);
```

4. Destroy the TAppMetaData object after using it.

```
  MyAMD.Free;
```

And that's it. Main step is step 3, which you will do all the operation you need (get tables, fields, indexes, etc.), as described in the example retrieving tables and fields.
This manual lists the main key properties of the data dictionary

2.2 Small example: retrieving tables and fields

Here is a small example that retrieves table and fields from the data dictionary in project file and display the names in a memo:

```
uses
  ...,
  uAppMetadata, uGDAO, dgConsts;

procedure TMyForm.RetrieveTablesAndFields;
var
  MyAMD: TAppMetaData;
  c: integer;
  d: integer;
begin
  MyAMD := TAppMetaData.LoadFromFile('c:\myproject.dgp');
  try
    for c := 0 to MyAMD.DataDictionary.Tables.Count - 1 do
    begin
      Memo1.Lines.Add('Table name: ' + MyAMD.DataDictionary.Tables[c].TableName);
      Memo1.Lines.Add('Available fields: ');
```

```
      for d := 0 to MyAMD.DataDictionary.Tables[c].Fields.Count - 1 do
        Memo1.Lines.Add('  ' + MyAMD.DataDictionary.Tables[c].Fields[d].FieldName);
      Memo1.Lines.Add('---');
    end;
  finally
    MyAMD.Free;
  end;
end;
```

2.3 Overview of properties and methods

TAppMetaData
Main property in TAppMetaData class is DataDictionary, which returns a TGDAODatabase class.

TGDAODatabase
The TGDAODatabase class provides the following main properties:

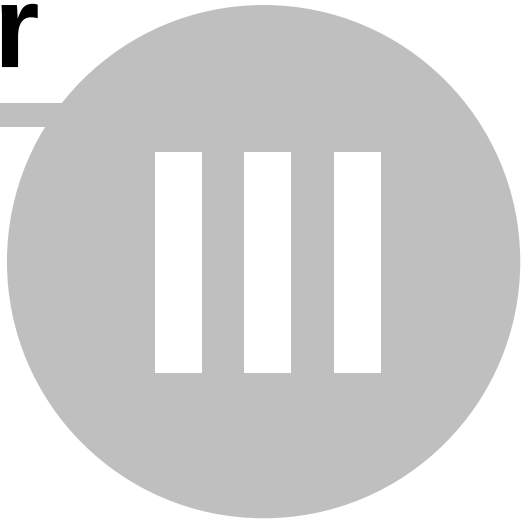
Tables: a TGDAOTables collection which contain the list of tables.
Relationships: a TGDAORelationships collection containing the existing relationships
Categories: a TGDAOCategories collection containing other database objects (procedures, triggers, etc.)
Domains: a TGDAODomains collection containing the existing domains

TGDAOTable
each TGDAOTable object has the following key properties:

Fields: a TGDAOFields collection containing all the table fields
Indexes: a TGDAOIndexes collection with all the indexes belonging to that table
Triggers: a TGDAOTriggers collection with all the table triggers
Constraints: a TGDAOTableConstraints collection with all the table constraints

each class has several properties with information about that object (field name, data type, trigger code, parent table of relationship, etc.). For a complete description of all classes, properties and methods, see the component reference.

Chapter



Component reference

3 Component reference

3.1 Component reference

TMS Data Modeler Library provides a full component reference with a detailed description of classes, properties and methods.
The component reference is provided in a separated help files, located in the "help" subdirectory in your TMS Data Modeler Library folder.
The component reference file also provides context-sensitive help from the Delphi IDE. The "Installation" topic of this manual explains how to integrate help files in to Delphi IDE.