

Matisse[®] Modeler

User's Guide

February 2012



Matisse Modeler User's Guide

Copyright ©1992–2012 Matisse Software Inc. All Rights Reserved.

This manual and the software described in it are copyrighted. Under the copyright laws, this manual or the software may not be copied, in whole or in part, without prior written consent of Matisse Software Inc. This manual and the software described in it are provided under the terms of a license between Matisse Software Inc. and the recipient, and their use is subject to the terms of that license.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. and international patents.

TRADEMARKS: Matisse and the Matisse logo are registered trademarks of Matisse Software Inc. All other trademarks belong to their respective owners.

PDF generated 23 February 2012

Contents

1	Introduction	4
1.1	What is Matisse Modeler?	4
1.2	Scope of This Document	4
1.3	System Requirements	4
1.4	Installing the Matisse Modeler	4
2	Creating a Schema Diagram	5
2.1	Overview	5
2.2	Modeler Environment	5
2.3	Specifying a Schema Namespace	6
2.4	Specifying a Schema Class	7
2.5	Specifying Inheritance with Generalization relationships	7
2.6	Specifying Attributes with Field properties	8
2.7	Specifying Relationship with Associations	11
2.8	Specifying an Index	12
2.9	Specifying an Entry-Point Dictionary	13
2.10	Specifying a Sql Method	13
2.11	Specifying a Comment	15
2.12	MtObject superclass	15
3	Loading an existing Schema	16
4	Generating Source Code	17
5	Save an Image of a Diagram	18
6	Printing a Diagram	19

1 Introduction

1.1 What is Matisse Modeler?

The Matisse Modeler provides you with a modeling tool to manage your database schema. With Matisse Modeler typically, you will:

- Draw a UML-like diagram of your application models, then export it into a Matisse database.
- Load the schema into the modeler as a UML diagram, work on the diagram, then save it back into the Matisse database to manage the evolution of your application.

Note that Matisse database schema can be designed using either ODL (Object Definition Language) or SQL DDL.

1.2 Scope of This Document

This document is a guide to using the Matisse Modeler to manage Matisse database schema with the Matisse Modeler tool. General information about designing Matisse database schema is covered in [Getting Started with MATISSE](#), which is available on the Matisse web site at <http://www.matisse.com/developers/documentation/>.

1.3 System Requirements

Before installing the Matisse Modeler, you must have installed:

- Matisse DBMS

1.4 Installing the Matisse Modeler

To install Matisse Modeler, run the Matisse Modeler installer. You can download it from:

<http://www.matisse.com/developers/downloads/>

2 Creating a Schema Diagram

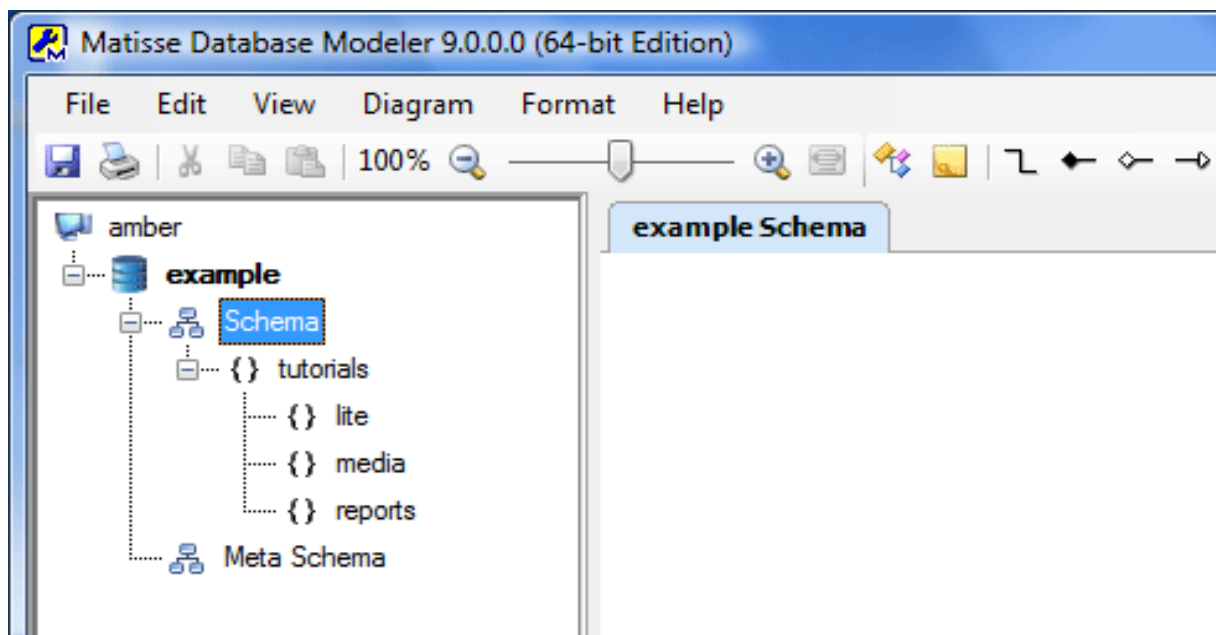
2.1 Overview

The Matisse Modeler allows you to manage your database schema in a UML-like diagram:

- Draw and update a UML diagram, then save the diagram into a Matisse database.
- Load a Matisse database schema into the Modeler for browsing or modification.

2.2 Modeler Environment

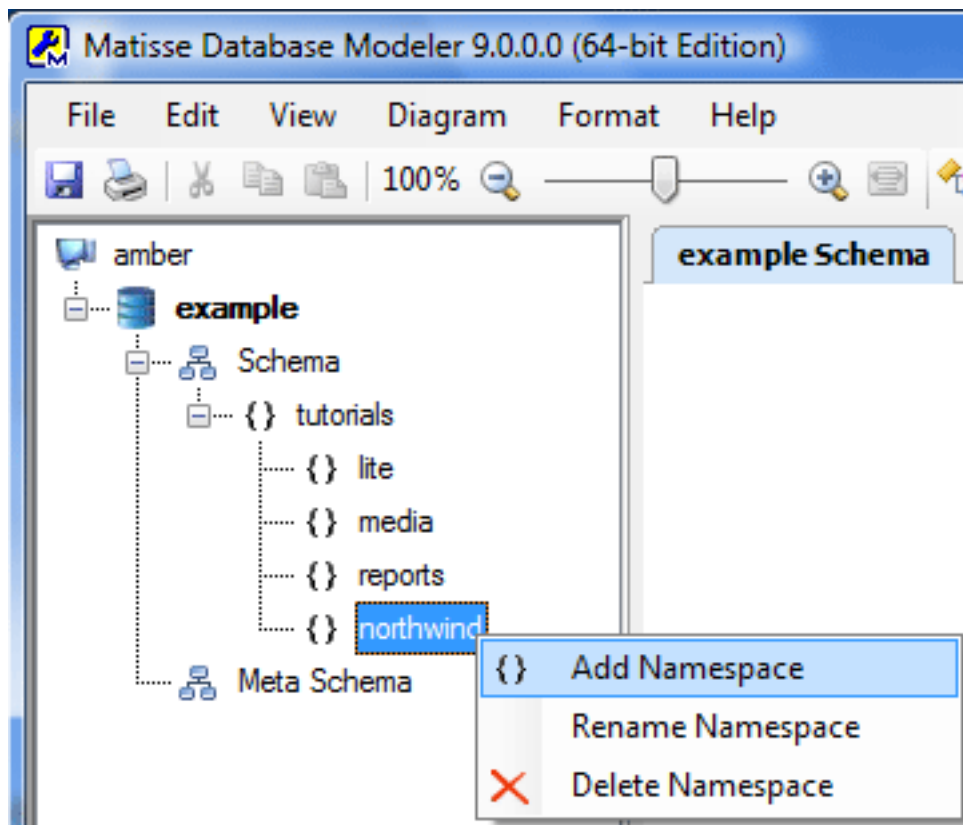
The first step before creating a diagram in the Modeler is to start a database since only online databases on a local machine will be listed by the Modeler tool.



1. For each database node, two diagram sub-nodes are created. The **Meta-Schema** node presents the diagram for the meta-classes and the **Schema** node and sub-nodes present the diagram of your application schema. The Schema node represents the root namespace and the sub-nodes represents the hierarchy of namespaces.

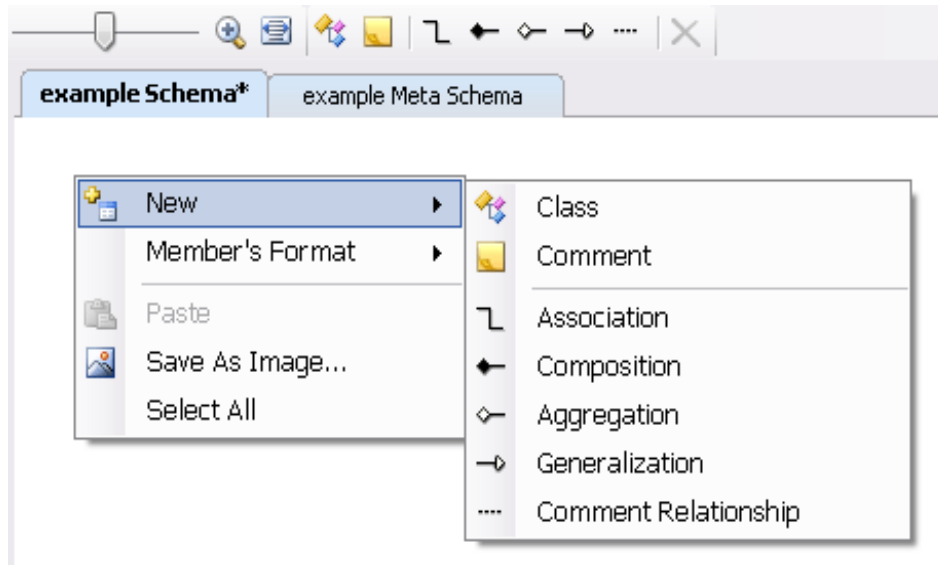
2. For each database, the modeler maintains a Database Schema Diagram (DSD) file that holds information about the positioning of the diagram entities as well as information not stored into the database. The DSD file is located in the **schema/Diagrams** directory in MATISSE_HOME. This file is not shared by users on the same machine. There is one file per database and per user.
3. The Modeler also logs the operations executed during a session into a log file located in the **log** directory in MATISSE_HOME.

2.3 Specifying a Schema Namespace



1. In the Schema Node popup menu, click the **Add Namespace** menu item.
2. A sub node representing the namespace object is created with the cursor positioned to updated the node label. Type the name of the namespace.

2.4 Specifying a Schema Class



1. In the Modeler toolbox or in the Schema Diagram popup menu, click the **Class** icon.
2. Click in the Schema Diagram window to create a new class icon at the cursor position.
3. Type the name of the class.
4. Click on the **Save** button to store the changes into the database.

NOTE: Deleting a class icon from the Schema class diagram does not remove the class from the database schema until you click on the **Save** button.

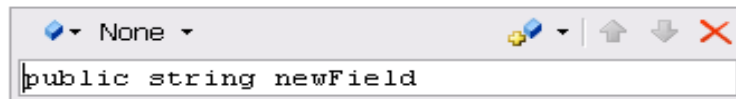
2.5 Specifying Inheritance with Generalization relationships

1. Create the two classes.
2. In the Modeler toolbox or in the Schema Diagram popup menu, click the **Generalization** icon.
3. Click in the **subclass**, move the mouse cursor to the **superclass**, and click again.

The `MatisseClsExample.odl` located in the `schema/ODL` directory provides an example of classes with inheritance. Load the file into your database via the Enterprise Manager and in the Modeler, click **Reload Schema** on the Database node.

NOTE: Deleting a generalization relationship from the Schema class diagram does not remove the subclass-superclass relation until you click in the **Save** button.

2.6 Specifying Attributes with Field properties



1. In the Schema class diagram, **double-click** on the class in which you want to create one or more attribute and select and click **New Field** to open the Member Editor.
2. Update the attribute name, the attribute type and press **Enter**. All the Matisse data types are supported. The syntax for data types is as follows:

```
type
type<n>
list<type>
list<type, n>
```

where n the maximum size for a string and the maximum number of elements for a list type.

3. Change the attribute modifier to **Nullable** if you want to define a nullable data type.
4. You can also define a default value for the attribute. Default values are described with the following syntax:
= *type(value)*

Here are some example of default values for various types:

```
boolean attBoolean = boolean(true)
byte attByte = byte(8)
short attShort = short(16)
integer attInteger = integer(32)
long attLong = long(64)
float attFloat = float(32)
double attDouble = double(64)
numeric attNumeric = numeric(19.20)

list<boolean> attBooleanList = list<boolean> (true,false)
list<short> attShortList = list(short) (1,8,16)
list<integer, 10> attIntegerList = list(integer) (1,8,16,32)
list<long> attLongList = list(long) (1,8,16,32,64)
list<float, 10> attFloatList = list(float) (1,32)
list<double> attDoubleList = list(double) (1,32,64)
list<numeric, 10> attNumericList = list(numeric) (1.123456,19.600000)
```

```

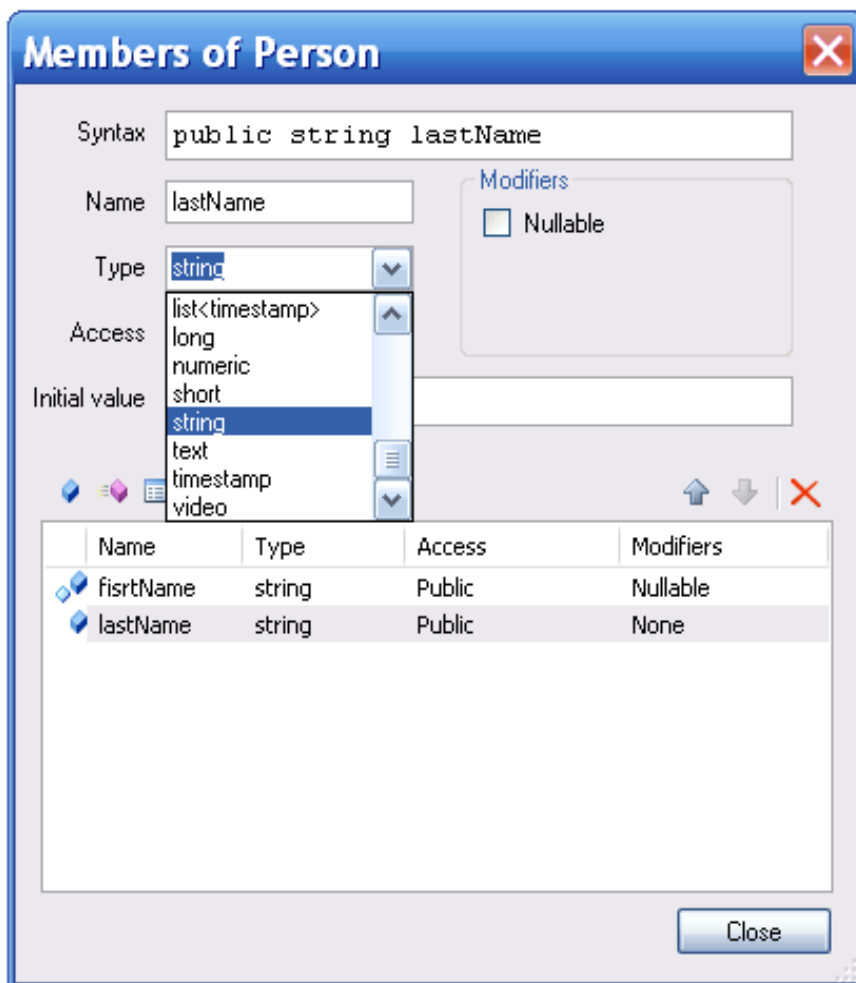
image<512000> attImage
audio attAudio
video attVideo = video()
bytes attBytes = bytes(0B1621)

char attChar = 'A'
string<256> attString = string("abcdefghi")
text attText = string("abcdefghi")
list<string> attStringList = list<string> ("abc","def","ghi")
utf16 string<10> attUtf16String = utf16 string("abcdefghi")
utf16 text attUtf16Text = utf16 string("abcdefghi")
utf16 list<string, 10> attUtf16StringList = utf16 list<string>
("abc","def","ghi")

date attDate = date("2011-01-01")
timestamp attTimestamp = timestamp("2011-02-02 12:34:56.000000")
interval attInterval = interval("-1234 01:02:03.000000")
list<date, 10> attDateList = list(date) ("2011-01-01","2011-01-01")
list<timestamp> attTimestampList = list(timestamp) ("2011-01-01
12:00:00.000000","2011-01-01
12:00:00.000000")
list<interval, 10> attIntervalList = list(interval) ("123
12:00:00.000000","-6 12:00:00.000000")

```

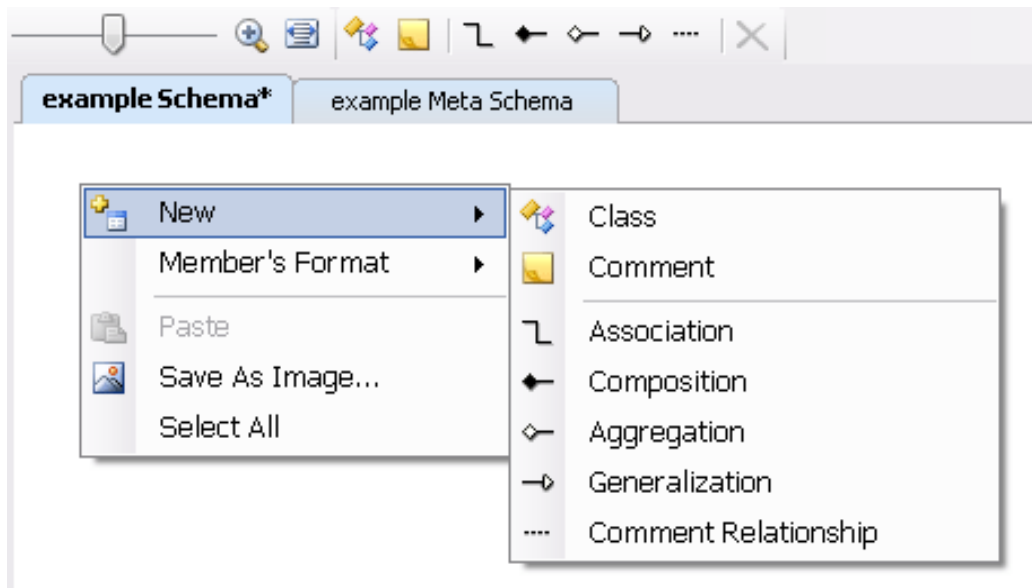
If you right-click on the class and select **Edit Members...**, the Edit Member dialog opens. It provides a more guided means for creating attributes.



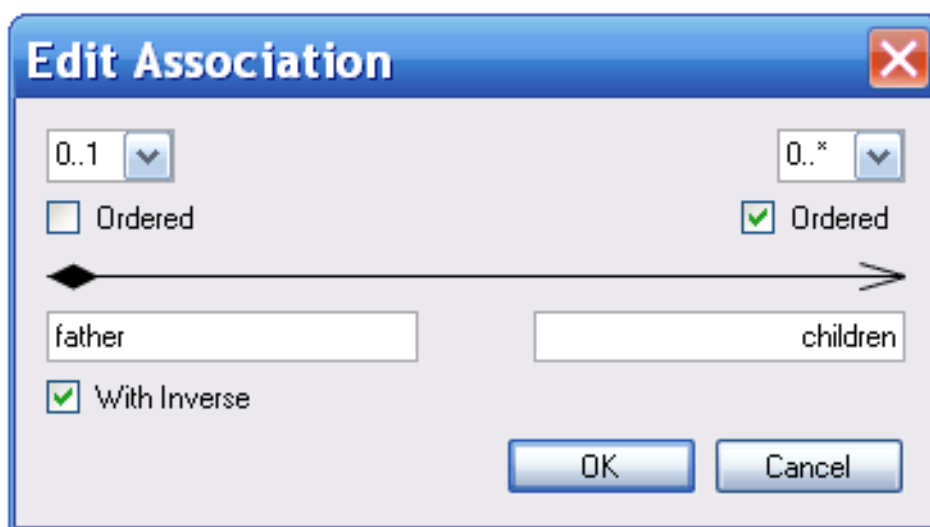
The `MatisseAttExample.odl` located in the `schema/ODL` directory provides an example of classes with attributes of various types. Load the file into your database via the Enterprise Manager and in the Modeler, click **Reload Schema** on the Database node.

NOTE: Deleting an existing attribute from a class does not remove it from the class into the database until you click on the **Save** button.

2.7 Specifying Relationship with Associations



1. In the Modeler toolbox or in the Schema Diagram popup menu, click on one of the **Association** icons.
2. In the Schema class diagram, click in one class icon, move the cursor to the other class, and click again.
3. **Double-click** on the association line or **right-click** on the association arrow and select **Edit Association** to open the Edit Association Dialog.



4. In the End Role (right side and Matisse direct relationship) and/or Start Role (left side and Matisse inverse relationship) fields, enter names for the relationships. When **With Inverse** check box is not selected, you are creating an uni-directional relationship.
5. Set the multiplicity (cardinality) for each defined role to one of the following:
 - 0..1: may have one successor, or none ([0, 1])
 - 0..*: may have any number of successors, or none ([0, -1])
 - 1: must have one and only one successor ([1, 1])
 - 1..*: must have at least one successor, may have any number ([1, -1])

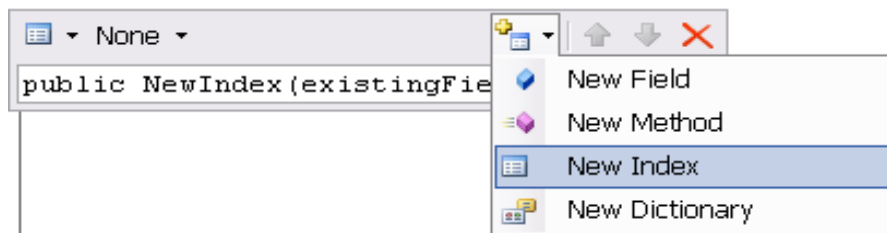
You can also enter your specific multiplicity in the format of: n or n..m or n..* where n and m are numbers.

6. Click on the extreme left side of the line changes the type of association. The type of association selected has no impact on the relationship definition in Matisse. The information is only preserved into the DSD file for convenience.
7. Click on the extreme right side of the line changes the direction of the relationship.

The `MatisseRelExample.odl` located in the `schema/ODL` directory provides an example of classes with relationships. Load the file into your database via the Enterprise Manager and in the Modeler, click **Reload Schema** on the Database node.

NOTE: Deleting an existing relationship does not remove the relationship from the database until you click on the **Save** button.

2.8 Specifying an Index

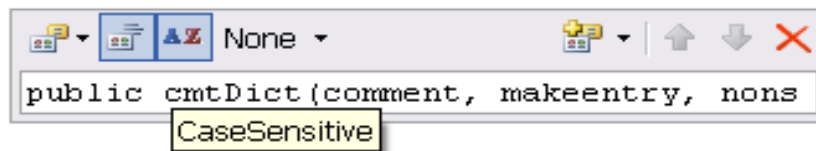


1. Create the schema class and the attributes that define the values to be indexed.
2. In the Schema class diagram, **double-click** on the class in which you want to create an index and select and click **New Index** to open the Member Editor.
3. In the Member Editor change the index name, the attribute name and the order `asc` or `desc` and press **Enter**. You can list up to four attributes separated by a comma (`att1 asc, att2 desc`).
4. Change the index modifier to **Unique Key** to define a primary key.

The `MatisseIdxExample.odl` located in the `schema/ODL` directory provides an example of classes with indexes. Load the file into your database via the Enterprise Manager and in the Modeler, click **Reload Schema** on the Database node.

NOTE: Deleting an existing index does not remove it from the database until you click on the **Save** button.

2.9 Specifying an Entry-Point Dictionary

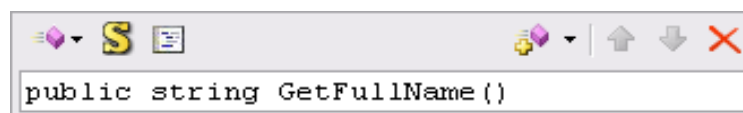


1. Create the schema class with an attribute (type `string` or `text`) for which the entry-point dictionary will be maintained.
2. In the Schema class diagram, **double-click** on the class in which you want to create an entry-point dictionary and select and click **New Dictionary** to open the Member Editor.
3. In the Member Editor change the dictionary name, the attribute name and press **Enter**.
4. Click on the **Case Sensitive** button or **Entry Function** button to change the characteristics of the dictionary.
5. Change the dictionary modifier to **Unique Key** to define a primary key.

The `MatisseEpdExample.odl` located in the `schema/ODL` directory provides an example of classes with dictionaries. Load the file into your database via the Enterprise Manager for example and in the Modeler tool click **Reload Schema** on the Database node.

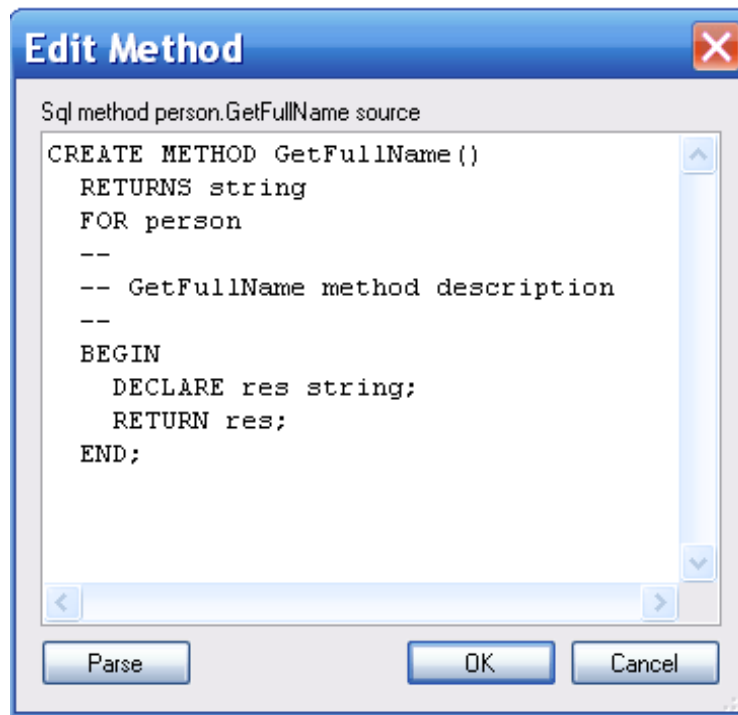
NOTE: Deleting an existing dictionary does not remove the dictionary from the database until you click on the **Save** button.

2.10 Specifying a Sql Method



1. Create a schema class.
2. In the Schema class diagram, **double-click** on the class in which you want to create a SQL method and select and click **New Method** to open the Member Editor.

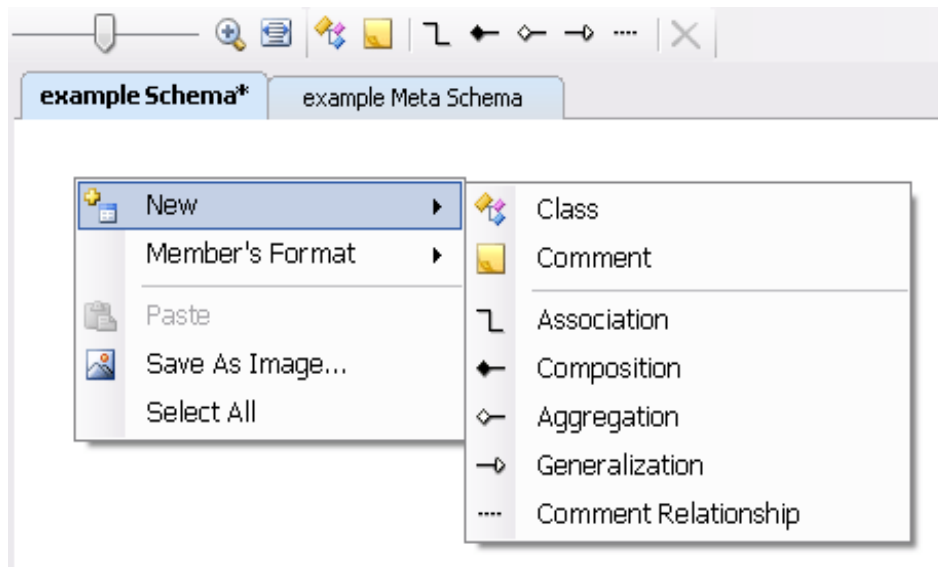
3. In the Member Editor change the method name, the return type, add parameters with the format (arg1 typeA, arg2 typeB) and press **Enter**.
4. Change the method modifier to **Static** to define a static method.
5. Click on the **View Method** button to edit the method source code.



The `MatisseMthExample.odl` located in the `schema/ODL` directory provides an example of classes with methods. Load the file into your database via the Enterprise Manager and in the Modeler, click **Reload Schema** on the Database node.

NOTE: Deleting an existing method does not remove the method from the database until you click on the **Save** button.

2.11 Specifying a Comment



1. In the Modeler toolbox or in the Schema Diagram popup menu, click the **Comment** icon.
2. Click in the Schema Diagram window to create a new comment icon at the cursor position.
3. Type the text of the comment directly into the icon.
4. In the Modeler toolbox or in the Schema Diagram popup menu, click the **Comment Relationship** icon to link a comment with another entity

NOTE: Comments and Comment Relationships have no effect on the schema and they are not saved into the database.

2.12 MtObject superclass

The Matisse `MtObject` class and an associated comment is represented in both the **Schema** Diagram and **Meta-Schema** Diagram as a reminder that all classes inherit from the base type `MtObject`.

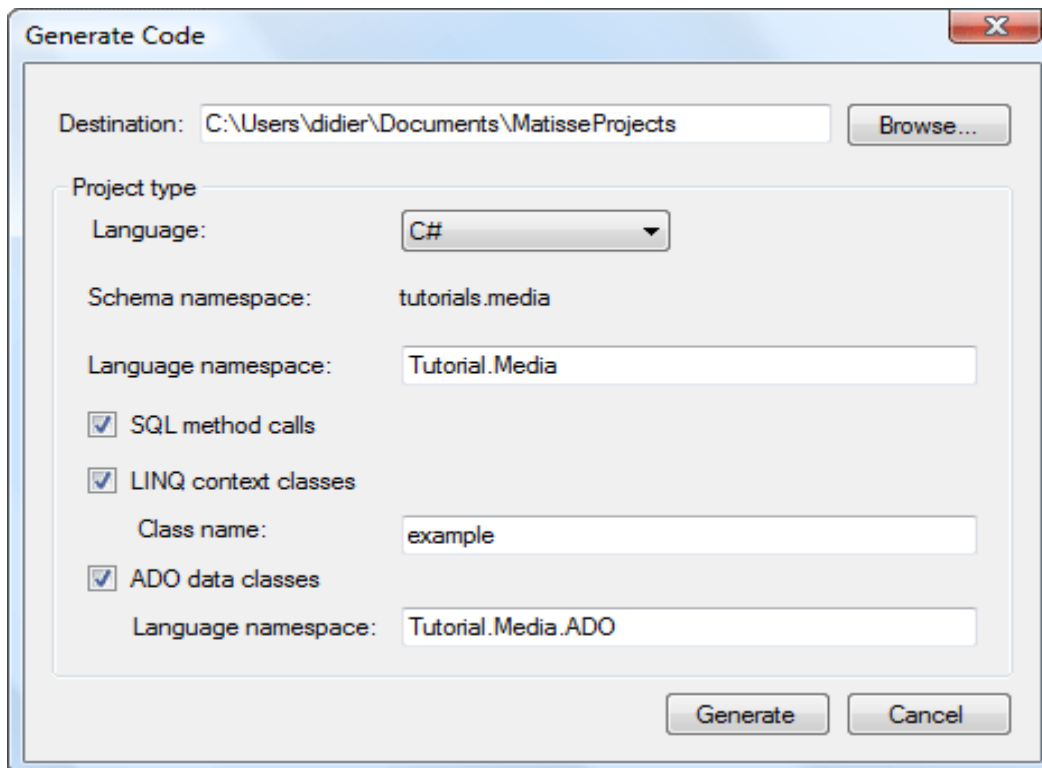
NOTE: The Default Super Class menu item in the Diagram menu can be unchecked to hide the `MtObject` class and an associated comment.

3 Loading an existing Schema

You can load an existing database schema from a Matisse database into the Modeler to benefit from the graphical representation of your database schema. Since there is no existing DSD file the classes will be positioned randomly next to one another in a square-like representation. You can move them as you see fit and click **Save** to generate a DSD file and preserve your new layout.

4 Generating Source Code

The Matisse Modeler is a schema design tool rather than a development environment, but for convenience it supports the code generation of stub classes of the main supported programming languages.



5 Save an Image of a Diagram

The Matisse Modeler also generates images of the schema diagram. You can export the whole diagram or selected elements in it. Major image formats are supported.

6 Printing a Diagram

The Matisse Modeler allows you to print a schema diagram. You can print the whole diagram or selected elements in it.