

Matisse® Rose Link User's Guide

September 2010



Matisse Rose Link User's Guide

Copyright ©1992–2010 Matisse Software Inc. All Rights Reserved.

Matisse Software Inc.
930 San Marcos Circle
Mountain View, CA 94043
USA

Printed in USA.

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 21 September 2010

Contents

1	Introduction	4
1.1	What is Matisse Rose Link?	4
1.2	Scope of This Document	4
1.3	System Requirements	4
1.4	Installing the Matisse Rose Link	4
2	Creating a Matisse Schema	6
2.1	Overview	6
2.2	Adding a New Package to a Rose Project	6
2.3	Specifying a Schema Class	6
2.4	Specifying Inheritance (Subclass - Superclass Relation)	7
2.5	Specifying Attribute Descriptors	7
2.6	Specifying Relationship Descriptors	7
2.7	Specifying an Index	8
2.8	Specifying an Entry-Point Dictionary	9
2.9	Current Limitations in Matisse Rose Link	9
3	Exporting from Rose to Matisse	10
4	Importing from Matisse	11
5	Generating Stub Code	11
6	Tutorial	12
6.1	Create the Project	12
6.2	Create the Person Class	12
6.3	Create the Employee and Manager Classes	13
6.4	Define Inheritance	13
6.5	Define the team / reportsTo Relationships	13
6.6	Define the assistant / assistantOf Relationships	14
6.7	Define the children / father and spouse Relationships	14
6.8	Define the personName Index	15
6.9	Define the commentDict Entry-Point Dictionary	15
	Appendix A: A Public Model for Genetic Analysis	16

1 Introduction

1.1 What is Matisse Rose Link?

With Matisse Rose Link, you can manage the Matisse database schema using the Rational Rose modeling tool. Typically, you will

- Draw a UML diagram with Rational Rose, then export it into a Matisse database.
- For updating the database schema, import the schema into Rational Rose as a UML diagram, work on the diagram, then export it back into the Matisse database.

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

1.2 Scope of This Document

This document is a guide to using the Matisse Rose Link to manage Matisse database schema with Rational Rose. 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 Rose Link, you must have installed:

- Matisse 8.0.2 or later
- Rational Rose 98 or later

1.4 Installing the Matisse Rose Link

To install Matisse Rose Link, simply download and run the Matisse Rose Link installer from:

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

The installer registers four COM components in Matisse_Rose_Addin.dll:

Matisse_Rose_Addin.DB
Matisse_Rose_Addin.MatisseExport
Matisse_Rose_Addin.MatisseImport
Matisse_Rose_Addin.MatisseReset

It also adds the following key to the registry:

HKEY_LOCAL_MACHINE\SOFTWARE\Rational Software\Rose\AddIns\Matisse_Rose_Addin

2 Creating a Matisse Schema

2.1 Overview

The Matisse Rose Link allows you to do following things:

- Draw/update UML diagram in Rose, then export the diagram into
 - a) a new Matisse database, or
 - b) an existing Matisse database to update the schema
- Import a Matisse database schema into Rose for browsing or modification

2.2 Adding a New Package to a Rose Project

The first step in creating a diagram in Rose, which you want to export to the Matisse database, is to add a package to hold the schema classes. You may skip this step.

1. Create a new Rose project, or open the project for which you want to design a Matisse schema.
2. In the Rose Browser, right-click on the Logical View folder and select New / Package.
3. Type Matisse and press Enter.
4. Right-click on the Matisse folder and select New / Class Diagram.
5. Type Schema and press Enter.
6. Double-click on the Schema icon to open it.

You may name the package and class diagram anything you like. For ease of description, throughout this document we will assume that they are named Matisse and Schema.

2.3 Specifying a Schema Class

1. In the Rose Toolbox, click the Class icon.
2. Click in the Matisse / Schema window to create a new class icon at the cursor position.
3. Type the name of the class.
4. In the Rose Browser, right-click on the class icon and select Open Specification.
5. Select the Detail tab, set Persistence to Persistent, and click OK.

Note: Deleting a class icon from the Matisse / Schema class diagram does not remove the class from the project. To delete, right-click on the class's icon in the Rose Browser and select Delete.

2.4 Specifying Inheritance (Subclass - Superclass Relation)

1. Create the two classes. (It makes no difference whether you add attribute descriptors, relationship descriptors, indexes, or entry-point dictionaries before or after specifying inheritance.)
2. In the Rose Toolbox, click the Generalization icon.
3. Click in the subclass, drag the mouse cursor to the superclass, and release.

Note: Deleting a generalization relationship from the Matisse / Schema class diagram does not remove the subclass-superclass relation. To delete, double-click the subclass's icon, select the Relations tab, right-click on the "Specialize Class [superclass]" entry, and select Delete.

2.5 Specifying Attribute Descriptors

1. In the Matisse / Schema class diagram, right-click on the class in which you want to create one or more attribute descriptors and select New Attribute.
2. Type the attribute name, a colon, and the attribute type. Supported types are: audio, boolean, boolean[], byte, byte[], char, date, date[], double, double[], float, float[], image, int, int[], int64, int64[], interval, interval[], numeric, numeric[], short, short[], string, string[], text, timestamp, timestamp[], video. For additional information, see the *Matisse Data Types Reference*.
3. To add another attribute, press Enter and repeat step 2; or, to stop adding attributes, click outside of the class.

2.6 Specifying Relationship Descriptors

1. In the Rose Toolbox, click the Unidirectional Association icon.
2. In the Matisse / Schema class diagram:

To specify a relationship to associate objects of different classes, click in one class icon, drag the cursor to the other class, and release.

To specify a relationship to associate objects of the same class: in the Matisse / Schema class diagram, click in the class icon, drag the cursor outside of the class, release the mouse, then click in the class icon again.

3. Double-click on the association arrow.
4. In the Role A and/or Role B fields, enter names for the two relationships or at least one of them. If you leave one of the names blank, Matisse will give a name, which is generated based on the other name, and will be invisible from Rational Rose.
5. Select the Role A Detail tab. Set Multiplicity (cardinality) for the relationship in the first class you clicked to one of the following:

- 0..1: may have one successor, or none (Matisse 0, 1)
- 0..n: may have any number of successors, or none (Matisse 0, -1)
- 1: must have one and only one successor (Matisse 1, 1)
- 1..n: must have at least one successor, may have any number (Matisse 1, -1)

6. Select the Role B Detail tab and set the cardinality for the other relationship.
7. By default, the Role B relationship is read-only; to make it read-write, check Navigable. When Navigable is checked for both roles, the schema diagram does not display any arrowheads for the relationship. At least one of the roles must have Navigable checked.
8. Click OK.

Note: Deleting a relationship from the Matisse / Schema class diagram does not remove the relationship from the schema. To delete, in the Rose Browser expand the Associations section of the Matisse package, right-click on the relationship, and select Delete.

2.7 Specifying an Index

1. Create the schema class and the attribute descriptors that define the values to be indexed.
2. Open the specification for the attribute to be associated with the first column of the index table (the one that is sorted first) and select the Matisse tab.
3. Double-click in the Value column to the right of MtIndex<i>, type the index name, and click Override.
4. Change the MtIndexUniqueKey<i>, and/or MtCriteriaOrder<i> settings as necessary, then click OK. For MtCriteriaOrder<i>, 1 means ascending (MtAscend), 0 descending (MtDescend). If the attribute type is string, the MtMaxSize field for the attribute needs to be set to a maximum number of characters for the attribute. For details, see *Getting Started with Matisse*, section for “Defining Indexes.”
5. Open the specification for the attribute to be associated with the next column and select the Matisse tab.
6. Set MtIndex to the same name you specified in step 3, and make sure that the attribute is not nullable.
7. Change MtCriteriaOffset to 2 for the second attribute, 3 for the third, or 4 for the fourth.
8. Change the MtIndexUniqueKey<i>, and/or MtCriteriaOrder<i> settings as necessary, then click OK.
9. Repeat steps 5 through 8 for any remaining attributes (up to a maximum of four).

Once you have exported a schema with an index to Matisse, the indexed attributes cannot be modified. If it is necessary to modify them, delete the index, modify the attributes, then recreate the index. To delete an index, clear the MtIndex field for all of the attributes, then export with the “Remove Rose objects not in Matisse” option checked.

2.8 Specifying an Entry-Point Dictionary

Use entry-point dictionaries with attributes of type `String` only. For other data types, use indexes.

1. Create the schema class and the attribute descriptor for which the entry-point dictionary will be maintained.
2. Open the specification for the attribute and select the Matisse tab.
3. Double-click in the Value column to the right of `MtEntryPoint`, type the dictionary name, and click Override.
4. Change the `MtEntryPointUniqueKey`, `MtCaseSensitive`, and/or `MtMakeEntryFunction` settings (discussed in *Getting Started with Matisse*, section for “Defining Entry-Point Dictionaries”) as necessary, then click OK.

2.9 Current Limitations in Matisse Rose Link

- An attribute may be used as a criterion in up to four database indexes. Import from Matisse will show an error message if the schema contains an attribute associated with more than four indexes.
- Microsecond values are not supported for timestamp or interval default values. Any such values will be truncated on import from Matisse.

3 Exporting from Rose to Matisse

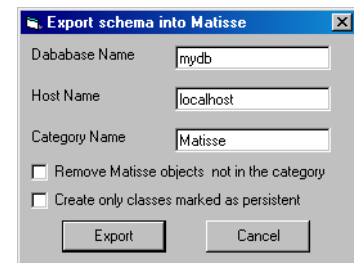
You can export the UML diagram in Rose to a Matisse database to create a new schema in a newly initialized database, or to update a database's existing schema. See *Getting Started with Matisse*, section for "Loading or Updating a Schema" for discussion of limitations on schema updates.

1. Save the Rose project.
2. Select Tools / Matisse / Export to Database.
3. Enter or revise the settings in the "Export schema into Matisse" dialog as necessary:

Database Name: The name of the database to which the schema will be exported. This will be blank for a schema not previously exported from or imported to this Rose project.

Host Name: The database server's network name. If it is a local machine, you can put 'localhost'.

Package Name: The name of the package in which you created the Matisse schema (typically "Matisse"). Leave this field blank if you want to export all the packages. Note that all the sub-packages of the specified package are also exported to the database.



Remove Matisse objects not in the category: If checked, any schema elements that exist in the database but not in Rose will be deleted from the database. Use this option to synchronize the database with Rose after you delete a schema class, attribute or relationship descriptor, index, or entry-point dictionary in Rose.

Create only classes marked as persistent: If the specified package(s) contains transient classes in addition to the persistent Matisse schema classes, check this box.

4. Click Export.
5. If you saved the project as instructed in step 1 above, click No to skip another save. Otherwise, click Yes. This will initiate a transaction with the Matisse database.
6. If Matisse encounters errors during the transaction, the transaction will be aborted; otherwise, you will get a Matisse Transaction Confirmation dialog box. Click OK to confirm or Cancel to abort the update with no changes to the database.
7. Check Rose's Log window. If the transaction was aborted or commit failed, it will display one or more error messages followed by a "transaction was aborted in Matisse" warning.

4 Importing from Matisse

You may import a database schema from a Matisse database into Rose for revision or simply to use the diagramming tools.

1. Save the Rose project.
2. Select Tools / Matisse / Import.
3. Enter or revise the settings in the “Import schema from Matisse” dialog as necessary:

Database Name: The name of the database from which the schema will be imported. This will be blank for a schema not previously imported to or exported from this Rose project.

Host Name: The database server's network name.

Package Name: The name of the package in which you created the Matisse schema (typically “Matisse”).

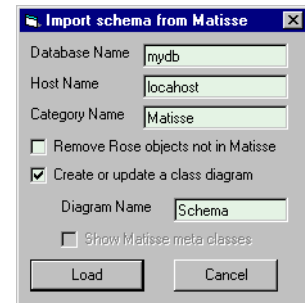
Remove Rose objects not in Matisse: If checked, any Matisse schema elements that exist in Rose but not in the database will be deleted. Use this option to synchronize Rose with the database after you delete a schema class, attribute or relationship descriptor, index, or entry-point dictionary from the database another tool, such as `mt_editor`, a SQL DROP statement, or C API functions.

Create or update a class diagram: If this is checked, the class diagram specified in the following field will be created or updated with the imported schema.

Diagram Name: The name of a class diagram (typically “Schema”) in the package specified in Package Name.

Show Matisse Meta-Classes: Not supported in this release.

4. Click Load.



5 Generating Stub Code

Since Rose is a design tool rather than a development environment, the Matisse Rose Link does not directly support code generation of stub classes for Java, C# and other supported languages.

However, once you have exported a diagram created in Rose to a Matisse database, you may export the database schema to an ODL file with `mt_sdl`, and then generate Java, C++, or Eiffel class files with the `mt_sdl` utility. C# or VisualBasic.NET classes are generated directly from the database using the `mt_dnom` utility. See the *Matisse Editor User Guide for MS Windows* and the *Matisse ODL Programmer's Guide* for instructions.

6 Tutorial

The following tutorial walks you through the steps required to create the schema used in the *Matisse C++ Programmer's Guide* and the *Matisse Java Programmer's Guide* (examples.odl).

6.1 Create the Project

1. Create a new Rose project.
2. In the Rose Browser, right-click on the Logical View folder and select New / Package.
3. Type Matisse and press Enter.
4. Right-click on the Matisse folder and select New / Class Diagram.
5. Type Schema and press Enter.
6. Double-click on the Schema icon to open it.
7. Save the project as `example.mdl`.

6.2 Create the Person Class

1. In the Rose Toolbox, click the Class icon.
2. Click in the Matisse / Schema window to create a new class icon at the cursor position.
3. Type Person.
4. In the Rose Browser, right-click on the NewClass icon (its name will then change to Person) and select Open Specification.
5. Select the Detail tab, set Persistence to Persistent, and click OK.
6. Right-click on the Person icon and select New Attribute.
7. Type the following (do not press Enter after the last line):

```
firstName:string  
lastName:string  
comment:string  
age:int  
photo:image=NULL
```

8. Make the `firstName` attribute nonnullable: in the Rose Browser, expand the Person branch, double-click on `firstName`, select the Matisse tab, change the `MtNullable` property to False, and click OK.
9. Make the `lastName` attribute nonnullable as well.

6.3 Create the Employee and Manager Classes

1. Create the class Employee to the right of the class Person, and set it Persistent.
2. Add the following attributes:

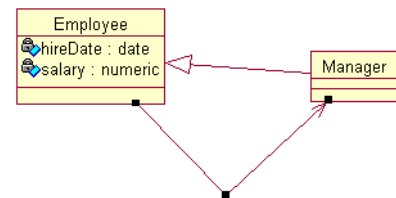
```
hireDate:date
salary:Numeric
```
3. Set both attributes nonnullable as described in step 8 of the previous section.
4. To the right of the Employee class, the Manager class. It has no attributes.

6.4 Define Inheritance

1. In the Rose Toolbox, click the Generalization icon.
2. In the Matisse / Schema window, click in Employee, drag the mouse cursor to Person, and release. This defines Employee as a subclass of Person.
3. Click the Generalization icon again, click in Manager, and drag to Employee. This defines Manager as a subclass of Employee.

6.5 Define the team / reportsTo Relationships

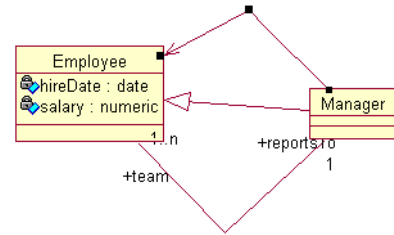
1. In the Rose Toolbox, click the Unidirectional Association icon.
2. For readability, we will draw the relationship as a line with an angle, as shown at right. In the Matisse / Schema class diagram, click in the Employee, drag to the point of the V, release the mouse, move the mouse cursor over Manager, and click.
3. Double-click on the new association arrow.
4. In the Role A field, enter “reportsTo” and in the Role B field enter “team”.
5. Select the Role A Detail tab and set Multiplicity to 1.
6. Select the Role B Detail tab, set Multiplicity to 1..n, check Navigable, and click OK.



By default, Role B is set read-only; checking Navigable makes it read-write. When both roles are navigable, Rose does not display any arrowheads, so when you click OK the existing arrowhead disappears.

6.6 Define the assistant / assistantOf Relationships

1. Create a unidirectional association as before, but this time draw it from Manager to Employee, and make the V point up, as shown at right.
2. Double-click on the new association arrow.
3. Name Role A “assistant” and Role B “assistantOf”.
4. On the Role A Detail tab, set Multiplicity to 0..1.
5. On the Role B Detail tab, set Multiplicity to 0..n, and click OK.

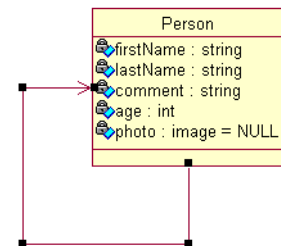


Since the schema calls for the assistantOf relationship to be read-only, Role B’s default setting for Navigable (unchecked) is appropriate.

6.7 Define the children / father and spouse Relationships

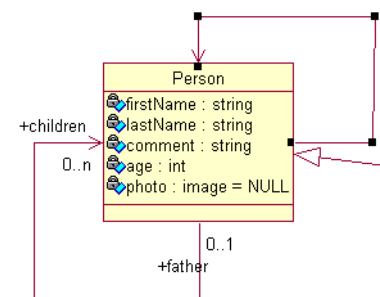
This relationship defines associations among objects of the same class. Consequently, the association arrow loops back to its origin class.

1. In the Rose Toolbox, click the Unidirectional Association icon.
2. For readability, we will draw the relationship as a rectangle, as shown at right. Click in Person, drag down out of the class, release the mouse, click at the bottom left corner, click at the top left corner, and click in Person. If you wish you can drag the corner points to adjust the angles to 90 degrees.
3. Double-click on the new association arrow.
4. Name Role A “children” and Role B “father”.
5. On the Role A Detail tab, set Multiplicity to 0..n.
6. On the Role B Detail tab, set Multiplicity to 0..1, and click OK.



The spouse relationship is one of the exceptions to the general rule that Matisse relationship descriptors are always declared in pairs. In Rose, that means that only Role A has a name, and no properties are set for Role B.

1. In the Rose Toolbox, click the Unidirectional Association icon.
2. For readability, we will draw the relationship as a rectangle, as shown at right. Click in Person, drag left out of the class, release the mouse, click at the top right corner, click at the top left corner, and click in Person.
3. Double-click on the new association arrow.
4. Name Role A “spouse”.



5. On the Role A Detail tab, set Multiplicity to 0..1 and click OK.

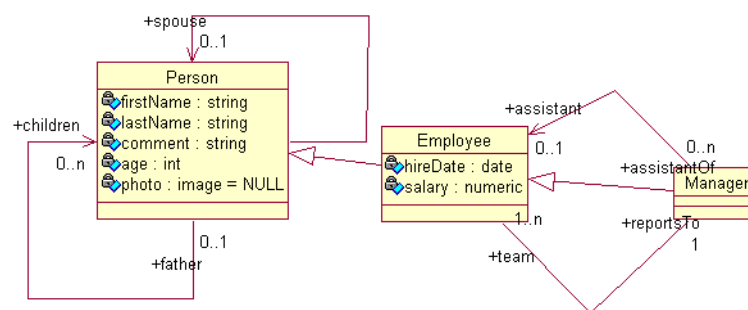
6.8 Define the personName Index

1. Open the specification for the lastName attribute and select the Matisse tab.
2. Double-click in the Value column to the right of MtIndex, type “personName”, and click Override.
3. Change MtCriteriaSize to 16, then click OK.
4. Open the specification for the firstName attribute and select the Matisse tab.
5. Set MtIndex to “personName” and click Override.
6. Change MtCriteriaSize to 16 and click Override.
7. Change MtCriteriaOffset to 2 and click OK. (This step was unnecessary for the first attribute since the setting defaults to 1.)

6.9 Define the commentDict Entry-Point Dictionary

Use entry-point dictionaries with attributes of type String only. For other data types, use indexes.

1. Open the specification for the comment attribute and select the Matisse tab.
2. Select MtEntryPoint, click in the empty Value column to its right, type “commentDict”, and click Override.
3. Change the MtMakeEntryFunction setting to “make-full-text-entry”, then click OK.
4. Save the model. It should look something like this:



This schema is now ready to be loaded into a newly-initialized Matisse database as described in [Exporting from Rose to Matisse](#) on page 10.

Appendix A: A Public Model for Genetic Analysis

To help you measure the benefits of managing complex data models in UML and to see how easy it is to load them into a Matisse database, we are providing an example UML representation of the MAGE Object Model.

The MAGE model is a standard for the representation of microarray gene expression used by many life science researchers. More detailed information about the MAGE project can be found on their web site at <http://www.mged.org/Workgroups/MAGE/mage.html>

The MAGE UML model has about 150 classes, 200 attributes, and 450 associations (relationships) in about 20 different packages.

A copy of the MAGE Object Model can be downloaded from our web site at <http://www.matisse.com/pdf/developers/Mage-om.zip>