Manual

Diploma thesis with working title Open DB Designer

dataMagus
Monika Schwitter & Jürg Zgraggen, 8Ibb-a

Revision

Version	Date	Comment	Author
1.0	18.11.2007	First version created	MOS

Table of contents

1 Introduction

dataMagus is a software to learn database modelling. The main goal is to help students to understand the concepts of various databases. Therefore, we begin by modelling an abstract sample of the real world in the logical layer and transform it to several existing database models such as the relational or the object-oriented model. In the end, we can even generate source codes for different targets which work with the relational or with the object-oriented model. dataMagus can show your models in different notations.

With the first version, dataMagus is distributed with tree different layers – there's the logical one which is essential, the above mentioned physical relational and the physical object-oriented layer. Furthermore, there are four notations implemented. While the notations of Chen and Modfied Chen go with the logical model, the physical relational layer displays its models with the aid of the Crowfoot notation (also known as the notation of Information Engineering). The object-oriented models are shown with the ODLG notation which was initiated by the ODMG standard.

2 Overview

The program is divided into four areas:

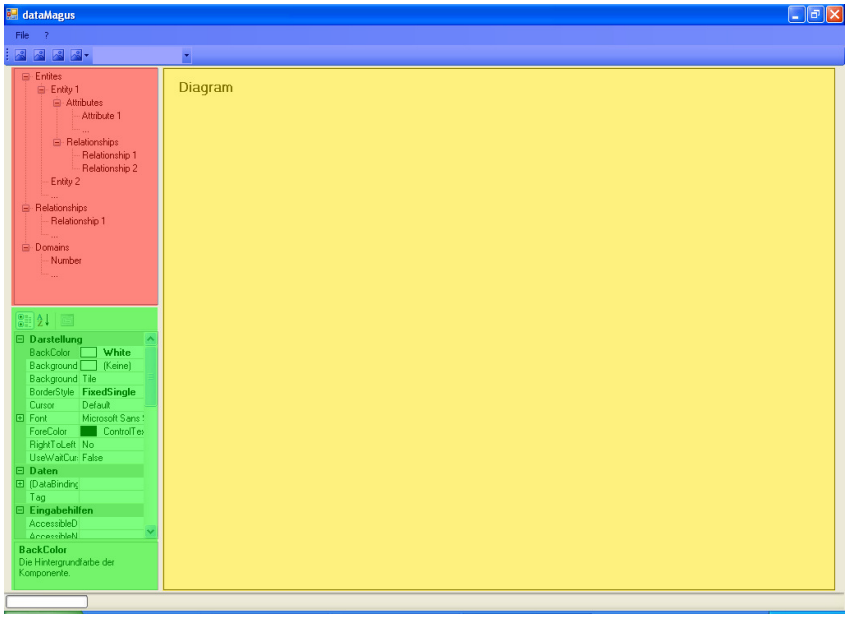

Figure 1 - Overview of the program parts

Blue range: Menu bar
In the menu bar you'll find general program options. At the right end of the toolbar you'll find two dropdowns. The first one allows switching between the logical and the physical layers, whereas you can change the notations with the second one.

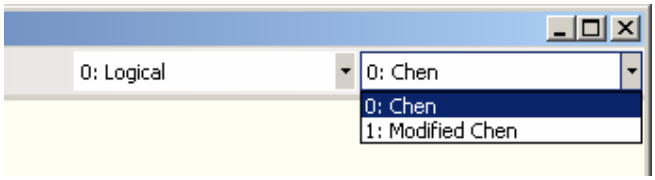

Figure 2 - Dropdown lists for changing the layers and the notations

Red range: Navigation tree
The navigation through the elements of a model works like a tree. It contains all elements depending on the model which is currently selected. The model can only be managed with the use of the navigation tree. Every node in the tree has its own context menu with which you can edit your schema. The content of the tree is displayed in the modelling window (yellow range). If you add a new item in the tree, it will then be displayed at the leftmost point in the modelling window (at the location of 0/0). The notation which is currently active specifies how the elements are displayed in the end. If you select an item in the tree, its properties are then shown in the property window (green range).

Green range: Property window
This window is handy for the management of properties no matter what type the elements are (entities, relationships, attributes etc.). You can set the value of simple properties directly in the window. Complex properties can be set with the use of the *Extendend Properties* button.

Yellow range: Modelling window
The modelling window shows the current schema with the active notation. The elements can be moved by using the mouse-cursor.

3 Create a logical model

You can only design your model in the logical layer. You have to use the navigation tree to add new elements to the model.

3.1 Manage entities

Just click the *Entities* node with the right mouse button. A context menu appears where you can click on *Add a new entity*. Immediately, a new entity pops up in the tree. You can change the name directly in the property window or in the dialog which is shown by a click on the *Extended Properties* button.

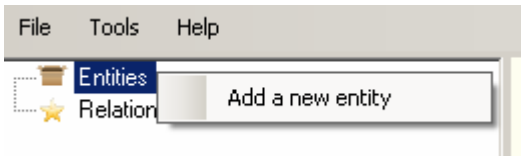

Figure 3 - Add a new entity

If you want to delete an entity, you can do that by single-clicking with the right mouse button on the selected entity. You'll find the delete function in the context menu. The entity and all its attributes and references will be deleted.

Figure 4 - Delete an entity

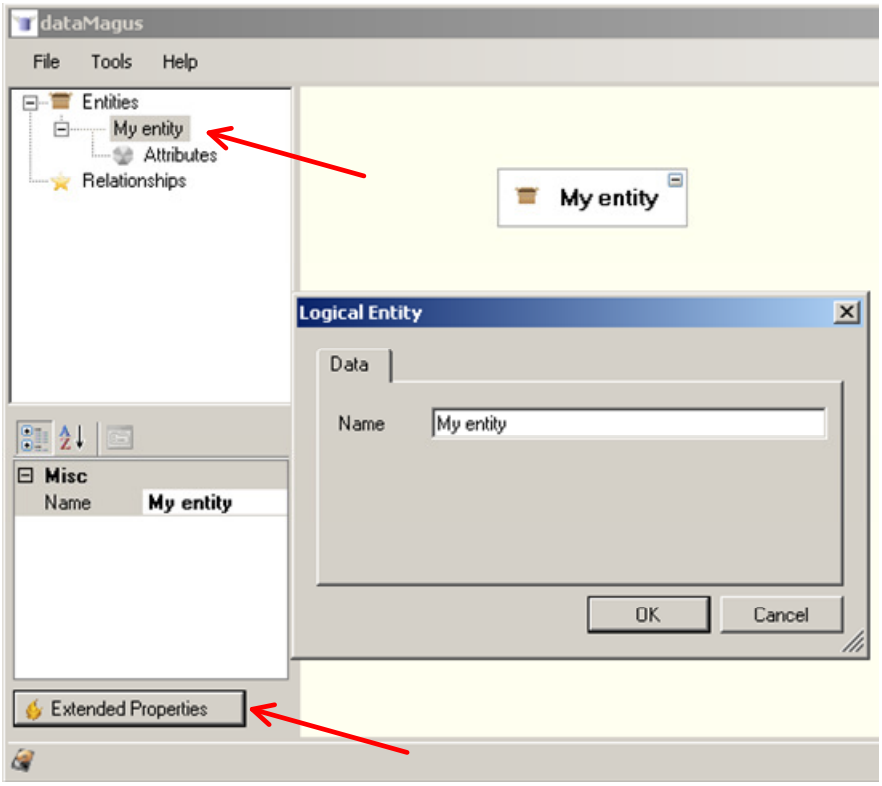

Figure 5 - Changing the entity's name

3.2 Manage relationships

In order to add a new relationship single-click with the right mouse button on the *Relationship*s node.

Figure 6 - Add a new relationship

A dialog box appears where you can specify the name and type of the relationship, the entities between which the relationship should be created and both of the role names.

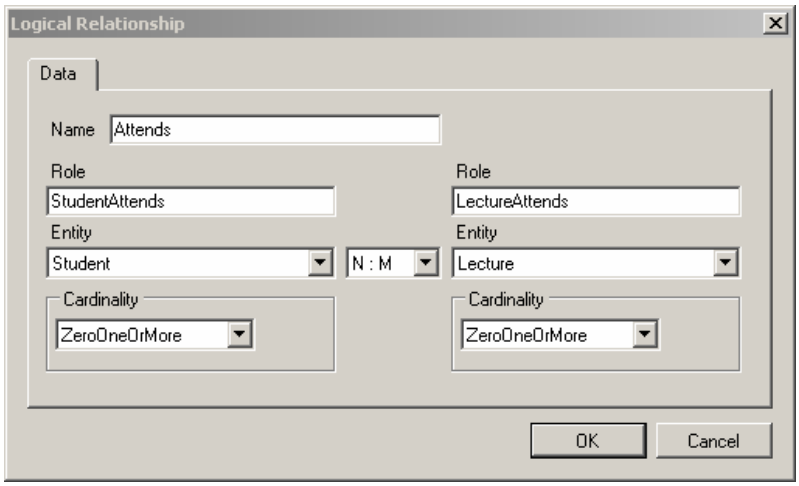

Figure 7 - Dialog box to add a new relationship

There are four different types available: you can design a 1:1 relationship, a 1:N relationship, a N:M relationship or a is-a relationship. Each relationship can define its own cardinalities which are immediately shown if you choose a relationship type. You have to type in at least a name. If you don't specify role names, the system will automatically generate default values for them. To take the most out of the program, we recommend the proper definition of name and role names.

You can design the same relationship more than once between the same two entities if it's not an is-a relationship. You can only define one is-a relationship between two entities because the program doesn't allow multiple inheritance. Even more, with is-a relationships it checks the model for circular references. If a circular reference would be created with the new relationship, the program gives you an error.
Self-relationships are allowed except for the is-a relationship.

Click the *OK* button and the relationship will be drawn in the modelling window.

If you want to delete a relationship, you just single-click with the right mouse button on the specific relationship and choose *Delete the relationship* in the context menu. All related elements such as attributes and entities will be deleted, too.

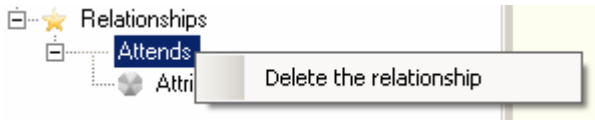

Figure 8 - Delete the relationship

You can always change the properties of the relationship by using the property windows.

3.3 Manage attributes

You can define attributes for an entity and for all the different relationship types except the is-a relationship. To add a new attribute, single-click the specific item with the right mouse button and choose *Add a new attribute* in the context menu.

Figure 9 - Add a new attribute

You can change the name of an attribute and specify if it should be a part of the key. The key definition is only allowed for attributes of simple entities or of a N:M relationship. In all other cases, the system won't accept your changes.

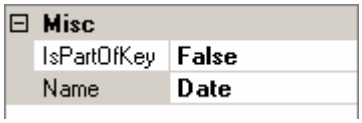

Figure 10 - Properties of an attribute

If you want to delete an attribute, you just single-click the specific attribute using the right mouse button and choose *Delete the attribute* in the context menu.

Figure 11 - Delete the attribute

3.4 Visual effects and notation handling

With the first version of dataMagus, the logical schema can be displayed with Chen or Modified Chen notation. Switch the notations by using the dropdown in the toolbar and have a look at the differences.

Figure 12 - Example by using the Chen notation

Figure 13 - Example by using the Modfied Chen notation

The attributes on the entities or the relationships can be hidden or displayed by expanding the specific item.

Figure 14 - Expanding attributes

If there are drawing faults (sometimes the program miscalculated the location of the relationship by a few points), you can refresh the whole schema by using the menu item *Tools / Refresh*.

If you don't feel like displaying the relationship names, role names and cardinalities, you can simply change the program options by selecting the menu item *Tools / Options*:

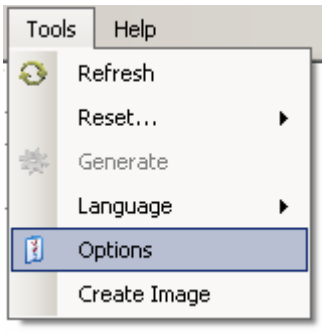

Figure 15 - Menu item for the program options

An options dialog box will appear where you can set the properties to your liking.

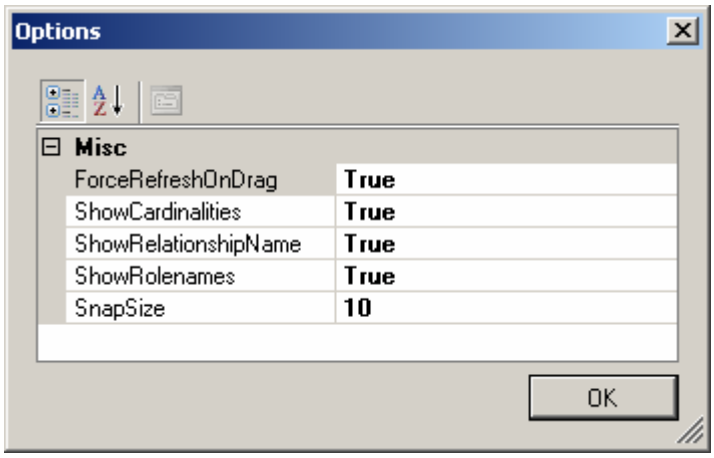

Figure 16 - Program options

If you set *ForceRefreshOnDrag* to false, the schema is drawn faster but sometimes improper and with delays.
You can deactivate the displaying of the cardinalities, relationship names and role names.
With *SnapSize* you can define how smooth the entities can be moved on the schema. The less it is defined; the smoother your entities will move.

4 Transform into physical models

Now, it's high time you had transformed your logical model into physical ones. You can initiate this process by switching from the logical layer to a physical one.

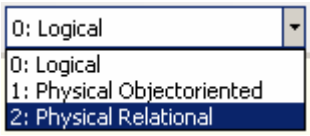

Figure 17 - Changing layers

And so the generation process begins… the program asks you questions about your model. You need some concentration to answer them. Depending on what you have modelled, the program needs to know exactly how to transform your logical construct. For example, let's look at a 1:1 relationship: on a relational database you have three different options to realize it. You can define the foreign key constraint in one of the two tables or you can define a link table. If you think about minimizing the null values in a table, the best option probably is the link table.
This step with the questions is very important for the understanding of the different database concepts!

Every single question will be posed by a separate dialog box. In the following example a N:M relationship has been designed. Now, the program now asks whether you'd like to have a generated technical primary key in the table or if you'd like to use the foreign key constraints for a natural primary key.

You only have to answer the question once for a physical model. dataMagus will remember your answer if you transform the model again.

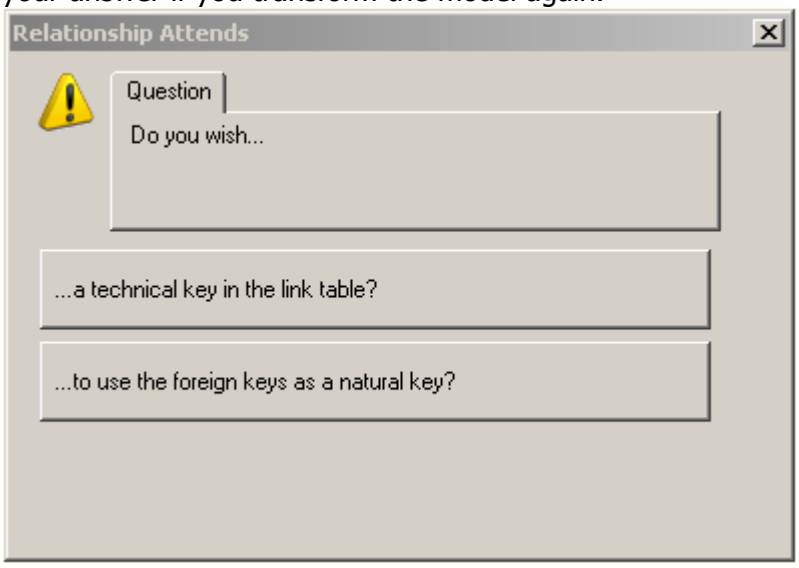

Figure 18 - User interaction within the generation process

If your logical model was correctly transformed, the program will not only display your physical model but only reposition it on the screen for you.

5 Handle physical layers

Before we have a look at each specific physical layer we'll discuss the elements and the handling which is common for each physical layer.

You might have noticed that your physical model has completely different elements on it. That's true: every database model has its own elements. However, the handling of these elements is similar to the logical model. The elements are also displayed in the navigation tree. Selected properties can be modified by using the property window.

There is no modelling work to do on a physical layer. If you have to change your model, you have to go back to the logical layer and change it there.

5.1 Manage data types

Up to now, we have been very abstract with our logical model. On the physical layer, we are getting closer to the target database which will later hold our schema. Therefore, we now can specify a data type for each column or property element.

Each physical navigation tree holds a *Datatypes* node. You can manage the data types by single-clicking with the right mouse button on this node and choosing the option *Manage the custom datatypes* in the context menu.

Figure 19 - Management of the custom data types

A dialog box will appear where you can see which basic data types are already defined for you and ready to use.

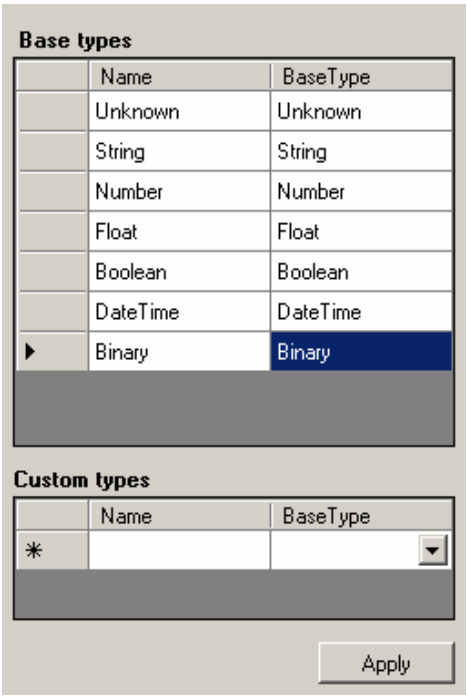

Figure 20 - Base data types

If you like, you can add your own custom types. Just make sure that they have a base type defined, otherwise the program will give you an error.

5.2 Making changes

Even if you're not allowed to do any modelling work on a physical layer, you can change a whole bunch of properties. You can change the name of an element for example. All this changes are saved in the background by the program. If you generate a particular physical layer again, the program will restore your self-made changes.

5.3 Reset physical data

If you would like to get rid of the saved changes and answers in the background because you'd like to check out another answer and to learn the differences in the generation process, you can reset the physical data.
Go back to the logical layer and have a look at the menu item *Tools / Reset*.

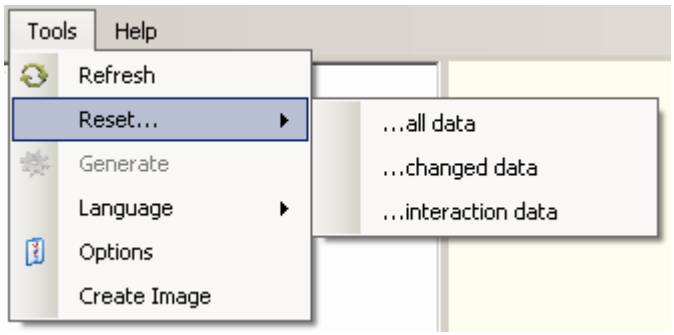

Figure 21 - Reseting physical data

Now, you are spoilt for choices. You can only reset the changes you made on the physical layers (such as a name etc.), you can only reset the interaction data (the answers to the questions from the generation process) or all physical data which includes both the physical changes and the interaction data.

6 Manage the physical relational layer

So, after we've discussed the main handling of the physical layer, we'll have a closer look on the physical relational layer.

After the generation process, your schema will be displayed by using the Crowfoot notation.

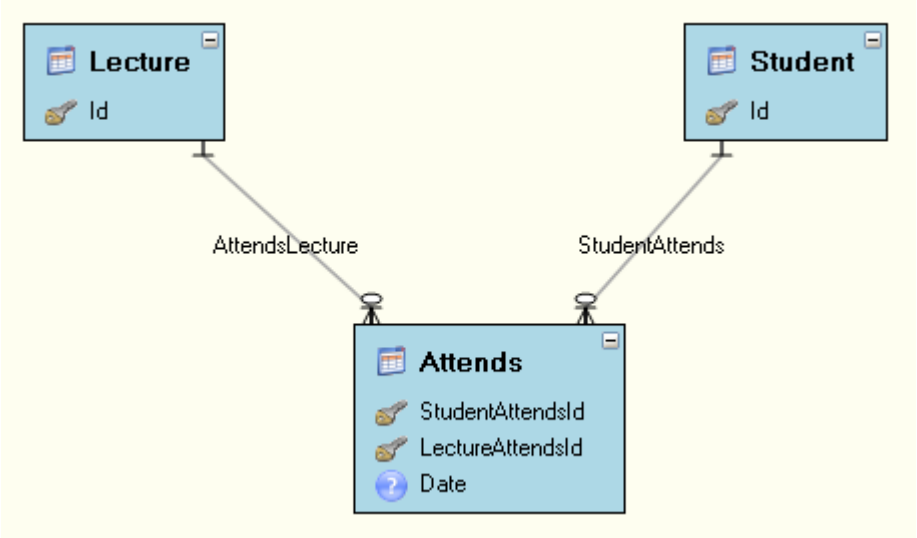

Figure 22 - Physical relational model by using the Crowfoot notation

A relational database works, among other things, with tables, relations, columns, primary keys, foreign keys, indexes and unique constraints. You will find all these things in the navigation tree.
Common relational databases are the MS SQL Server or the PostgreSQL database. Those are referenced by targets in dataMagus. Each target system can handle SQL code. We'd like to generate SQL code from our physical model to run it on a real database (or target system). dataMagus supports every target for which there has been implemented a code generator by a plugIn. By default, there are code generators available for MS SQL Server 2000 and for the PostgreSQL database. But before we can generate SQL code, we need to somewhat prepare our physical schema.
That fore, each element has its own properties which can be changed by the property window. Now, we recommend using the *Extended Properties* button because you can make target-specific settings if there are any defined.
Each dialog box consists of two tab pages. One of them allows changing the physical data; with the other one, which is named *Target*, we can handle the above mentioned target-specific settings.

Subsequent, we'll look at the most important settings on the relational model.

6.1 Defining the data type mapping

You have already learned to add custom data types. Because these types are physical abstract only, we need to define a data type mapping for each of them. Each target defines its own data types.

Please open the data type dialog box again and switch to the *Taget* tab page.

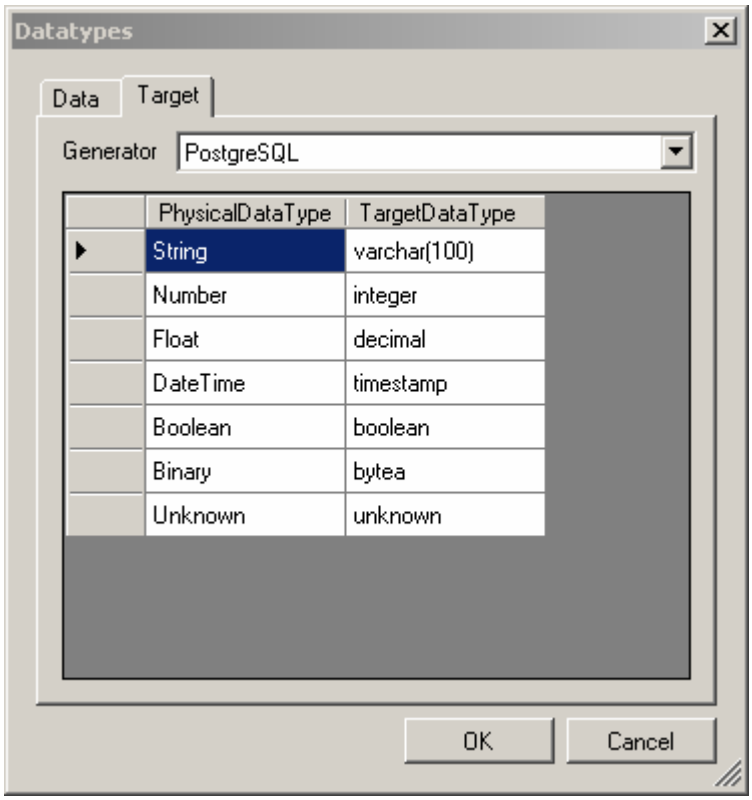

Figure 23 - Define the data type mapping for each target

The dropdown holds an entry for each generator which is available for the relational model. You can choose one and define the data type mapping by changing the entry in *TargetDataType* for every single base type. By default, there are mappings defined. If you have created any custom data types, they will be displayed here, too.
If you make modifications in the *Data* tab page, make sure that you click on *Apply* before switching back to the target-specific settings.

6.2 Changing column settings

There are a couple of settings to apply to a column. Please open the *Extended Properties* dialog for a column.

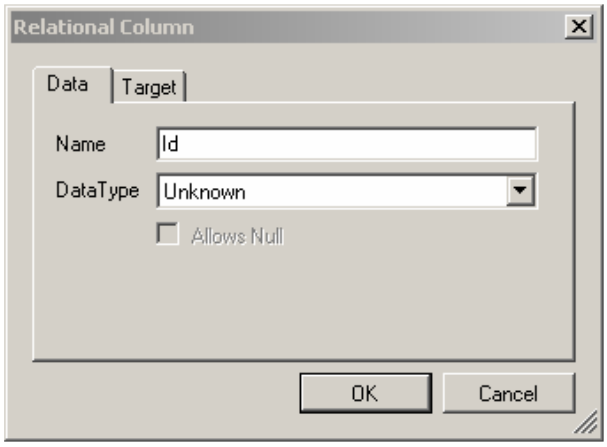

Figure 24 - Changing column properties

You can define a data type for every column. Furthermore, you can specify whether a column can have NULL values. But there are a couple of things to consider.

Firstly, you can only set the *Allows Null* flag on columns which are not part of the primary key. Why that? A primary key can only be UNIQUE and NOT NULL.
Secondly, you can't set a data type on a column which is part of a foreign key constraint. How comes this? A foreign key constraint always works together with the primary key of the table it references. Therefore, the primary key columns specify the data type for the foreign key columns. So, if you change a data type on a primary key column, all related foreign key columns automatically change to the same data type as the primary key has.

6.2.1 Column settings for the MS SQL Server 2000

The generator for the MS SQL Server 2000 allows us to specify additional settings. We can mark the column as an IDENTITY column. But be aware: if the program doesn't allow you to set the *Identity* flag, there usually is a good cause for that: You can only define one column per table as an IDENTITY column. IDENTITY columns must be numeric (data type *Number* or *Float*). Last but not least, an IDENTITY column must not allow nulls. Have a look at the SQL Server 2000 manual and think about it.

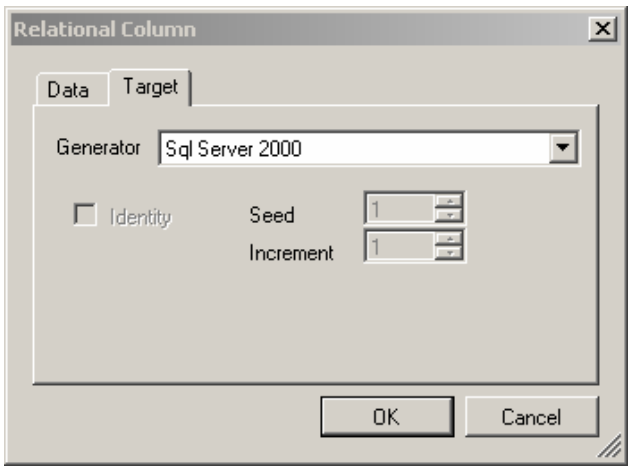

Figure 25 - Column settings for the MS SQL Server 2000

If you mark a column as an IDENTITY column, you can also set a seed and increment value. By default, both settings are set to one.

6.2.2 Column settings for the PostgreSQL database

Sorry, there are no target-specific options implemented so far.

6.3 Changing relationship settings

Please open the *Extended Properties* dialog box for a relationship.

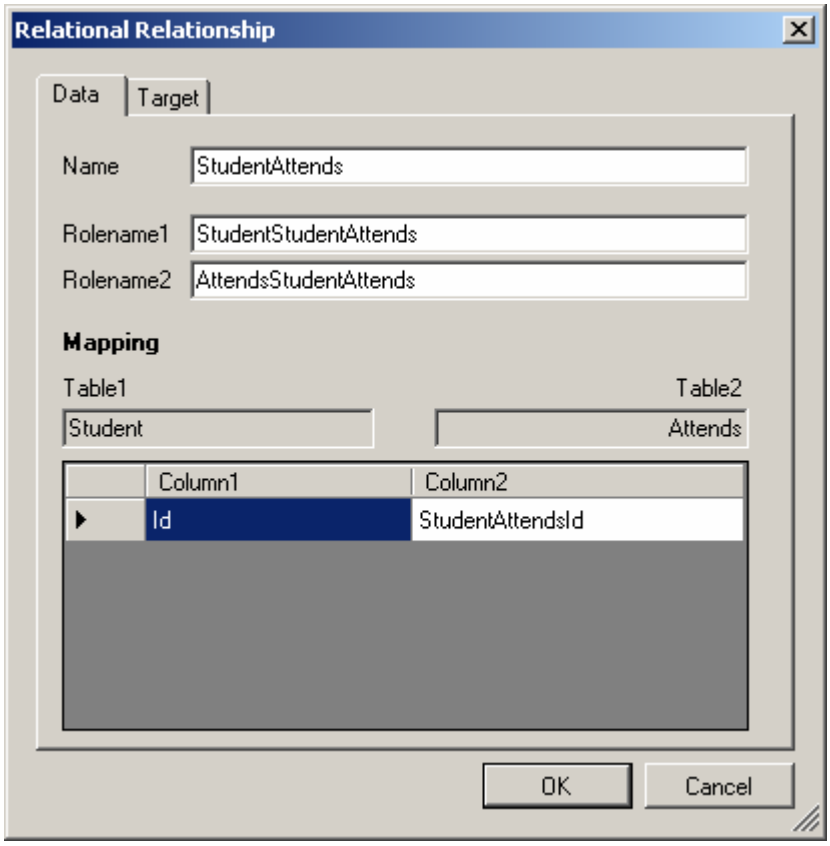

Figure 26 - Changing relationship properties

The dialog box shows you the relationship and the role names and even more importantly the column mapping. If you study the column mapping, you are able to understand how a foreign key constraint works.

6.3.1 Relationship settings for the MS SQL Server 2000

The SQL Server 2000 generator allows you to specify actions for the referential integrity. You can set actions for updating and deleting statements.

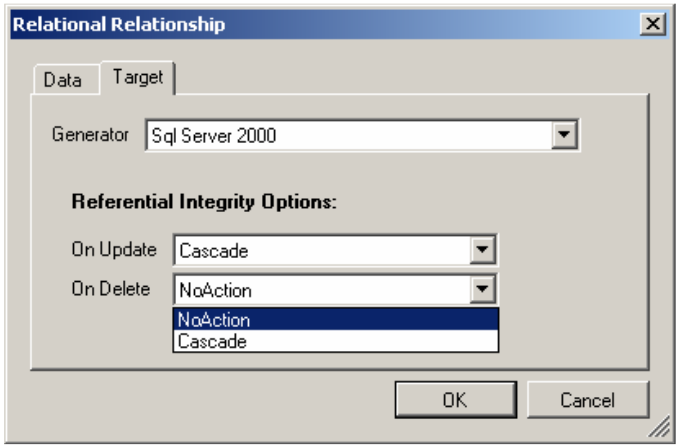

Figure 27 - Relationship settings for the MS SQL Server 2000

6.3.2 Relationship settings for the PostgreSQL database

The generator for the PostgreSQL database also lets you specify actions for the referential integrity. However, this database defines more actions than the SQL Server 2000 generator.

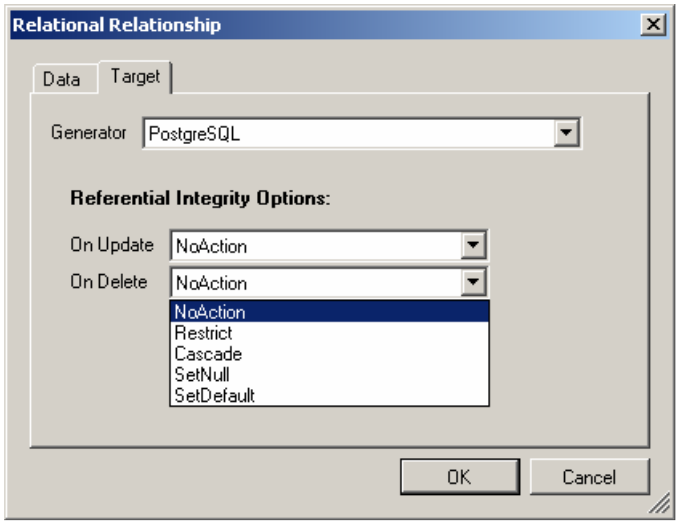

Figure 28 - Relationship settings for the Postgre SQL database

6.4 Generating SQL code

After you have finished specifying physical settings you can generate SQL code for an implemented target system. You can do that by selecting the menu item *Tools / Generate*.

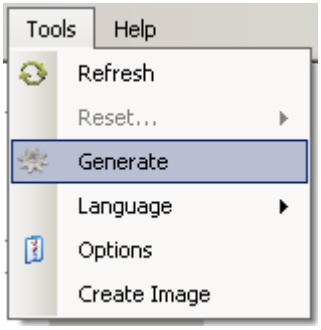

Figure 29 - Generating source code

The generator dialog box appears and displays the available generators for the relational model.
There are two tab pages: Settings can be made for the code generation process with *Options* tab page. If you click the *Generate* button, the code will be generated for you and displayed on the *Source Code* tab page.

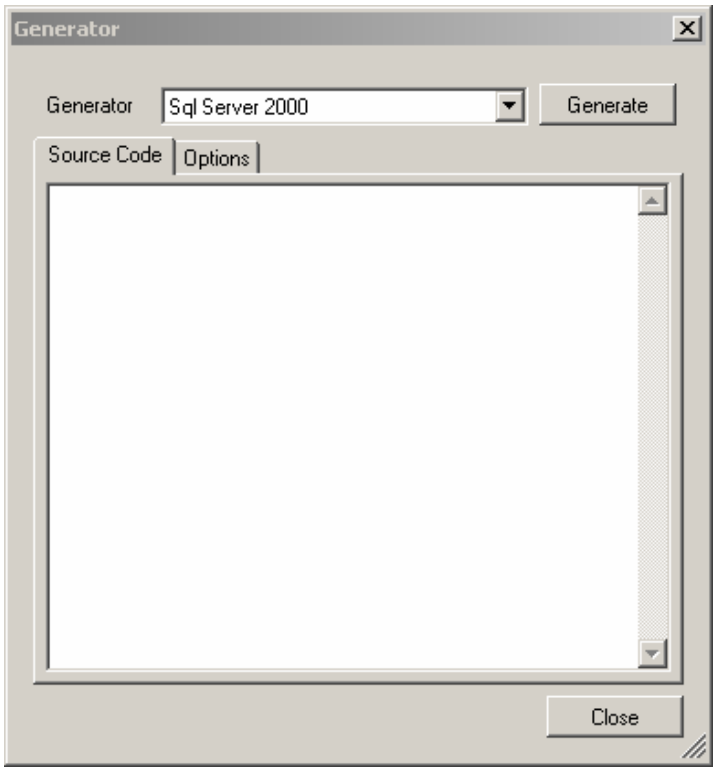

Figure 30 - Generating SQL code

6.4.1 Generating SQL code for the MS SQL Server 2000

The SQL Server 2000 generator allows you to make a few settings. You can specify if you want to generate the primary key constraint as column constraint or as table constraint. If a primary key contains more than one column, the constraint is always generated as a table constraint regardless what you have defined.
Furthermore, you can choose if you want to generate the foreign key constraints as a column constraint, as a table constraint or even with the ALTER TABLE-Statement. Again, if you choose the option *column constraint* and a foreign key constraint references more than one column, it is automatically generated as a table constraint. The same goes with the unique constraint.
The SQL Server 2000 generator creates fully functional SQL code. The only drawback is that the order of the tables doesn't correspond to the dependencies which the tables have with each other. If you want to have a working statement for sure, we recommend using the options *Foreign keys with ALTER-TABLE-Statements*.

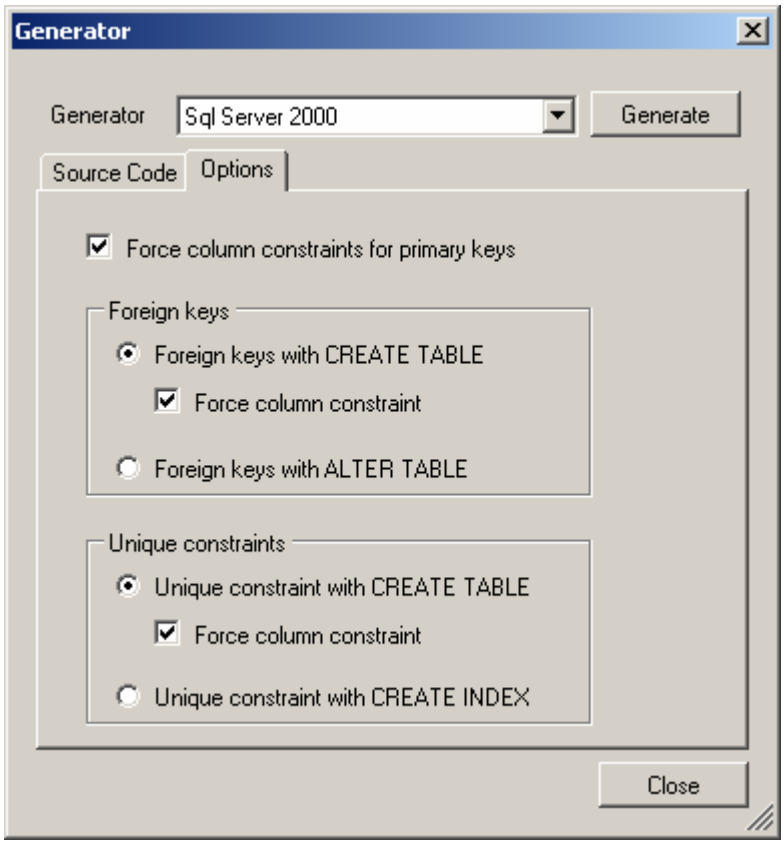

Figure 31 - Generator options for the SQL Server 2000 generator

6.4.2 Generating SQL code for the PostgreSql database

The PostgreSql database generator allows you to make the same settings as the SQL Server 2000 generator. There are the same things to consider. However, there is a small difference in the generated code. Can you make it out?

7 Manage the object-oriented layer

The other physical layer that dataMagus supports is the object-oriented one. After the generation process your schema will be displayed by using the ODLG notation.

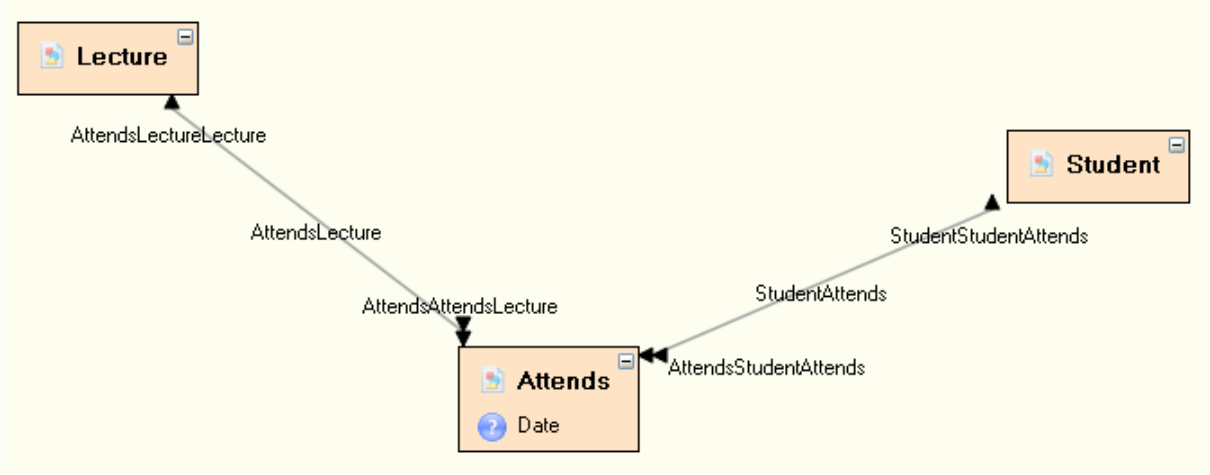

Figure 32 – Physical object-oriented model by using the ODLG notation

An object-oriented database works with classes, properties and associations. You will find all these elements in the navigation tree.

The handling of this layer is exactly the same as it is on the relational layer. You can change the properties of the physical data and generate code. By default, we only implement one generator for this layer: the ODL generator. This is a very simple generator because it doesn't specify any target-specific data.

You can make a few options for the source code generation such as whether you want to have inverse relationships, extensions or keys generated.

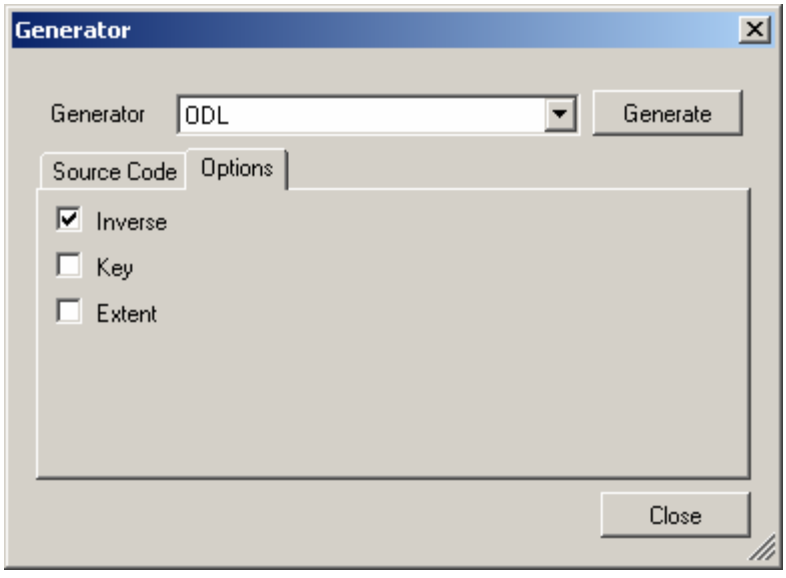

Figure 33 - Generator options for the ODL generator

8 Other functionality

8.1 Make a new model

If you want to start a new model, please use the menu item *File / New*. This menu item is only enabled in the logical layer.

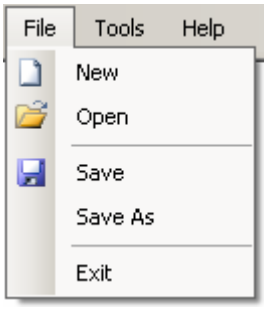

Figure 34 - New, Open and Save options

8.2 Save your model

If you want to save your mode, pleas use the menu item *File / Save* or *File / Save As*. This menu item is only enabled in the logical layer.

8.3 Load your model

You can open a saved model by using the menu item *File / Open*.

8.4 Make a picture

If you would like to export your model into a picture, please use the menu item *Tools / Create image*. There, you have to specify the size of your picture.

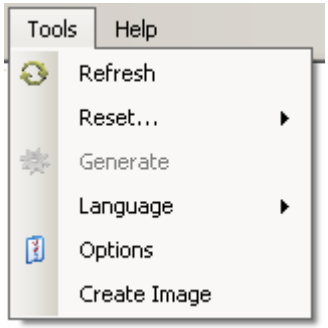

Figure 35 - Creating an image

Index of figures