# EvolveDB - Documentation for End-User

*Author:* Torben Eckwert, M.Sc.
*E-Mail:* torben.eckwert@zdh.thm.de

*Subject Area:* Computer Science

*Supervisors:* Prof. Dr. rer. nat. Michael Guckert
Prof. Dr. ing. Gabriele Taentzer

December 8, 2022 , Wetzlar

FORSCHUNGSCAMPUS MITTELHESSEN

# Contents

# 1 Introduction

EvolveDB is an Eclipse-based framework for schema evolution in MySQL databases. The user specifies the evolution steps by freely editing a database model extracted by reverse engineering. EvolveDB analyzes the differences between the status quo and the evolved model structures and generates a data migration script. This user manual includes installation instructions and introductory tutorials on how to use EvolveDB as an end user.

# 2 Prerequisites and Installation

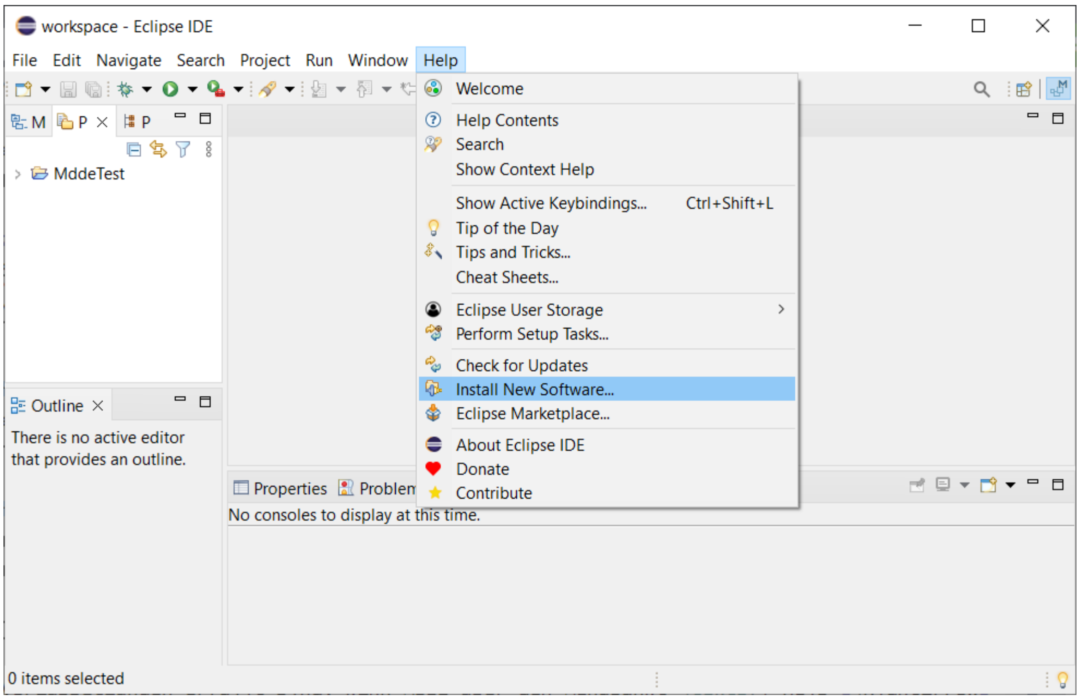
EvolveDB is a plug-in for recent versions of the Eclipse Modeling Tools (last tested version: 2022-06). The Eclipse feature is available from our GitHub repository.

**Important: Currently, EvolveDB requires Java SE 11. Newer or older Java Versions may cause unexpected behavior. The Java version can be changed by modifying the vm option in the *eclipse.ini*.**

Before installing EvolveDB, we need to install some additional Eclipse plugins. The following list contains the mandatory plugins.

- Sirius is an Eclipse project which allows you to easily create your own graphical modeling workbench by leveraging the Eclipse Modeling technologies, including EMF and GMF.
- Eclipse OCL is available via the eclipse marketplace.

If all requirements are fulfilled, EvolveDB can be installed via the menu item **Help ⇒ Install New Software...** (fig. 1).
We used the MySQL Server version 8.0.28 for this tutorial. When using older versions, compatibility problems may occur.

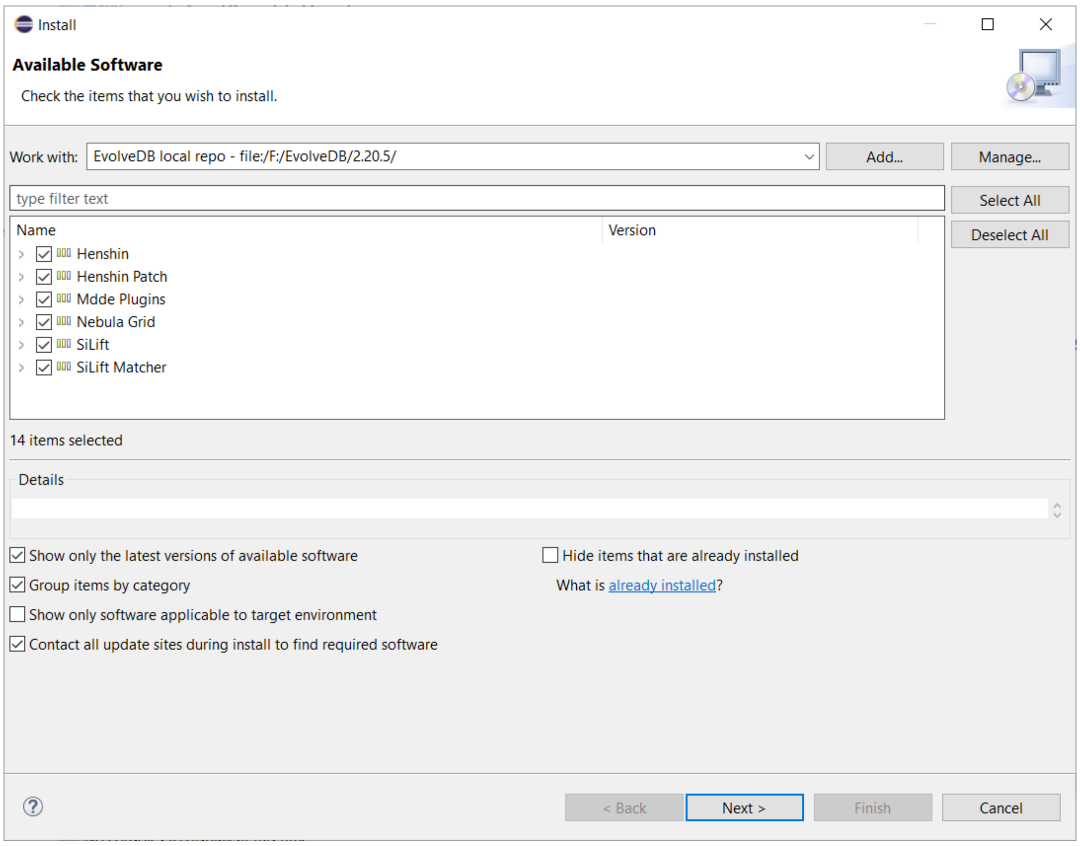


There is no active editor that provides an outline.

No consoles to display at this time.

Figure 1: Eclipse: Install New Software...

The repository contains multiple catgories (fig. 2). For the following tutorial, all elements must be installed.


Install

**Available Software**

Check the items that you wish to install.

Work with: EvolveDB local repo - file:/F:/EvolveDB/2.20.5/

type filter text

| Name | Version |
|---|---|
| ☑ Henshin | |
| ☑ Henshin Patch | |
| ☑ Mdde Plugins | |
| ☑ Nebula Grid | |
| ☑ SiLift | |
| ☑ SiLift Matcher | |

14 items selected

Details

☑ Show only the latest versions of available software
☑ Group items by category
☐ Show only software applicable to target environment
☑ Contact all update sites during install to find required software
☐ Hide items that are already installed
What is already installed?

Figure 2: Eclipse: EvolveDB local repository

## 3 EvolveDB

The schema evolution with EvolveDB is a model-driven reengineering process that includes three phases, reverse engineering, restructuring, and forward engineering. In the first step, we start by creating a representation of a (MySQL) database schema through reverse engineering. In the restructuring phase we edit this model resulting in a second model version. Afterwards EvolveDB analyzes the differences between the status quo and the evolved model structures. From the delta between the two model versions, SQL migration scripts are generated, which can be used to migrate the schema and the associated data.

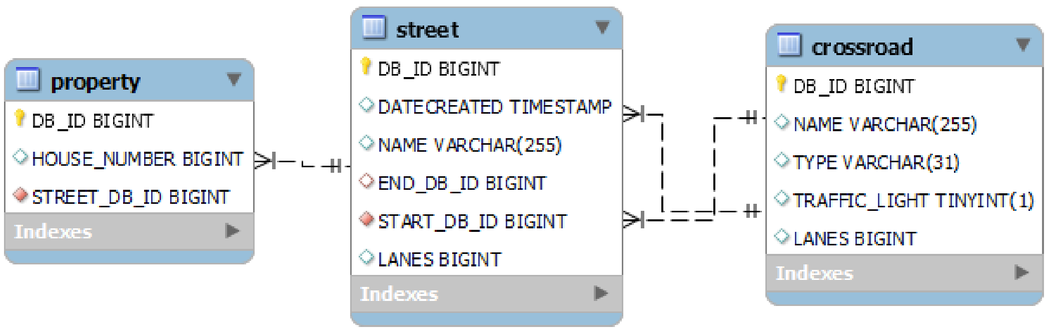


Figure 3: MySQL database schema

The use of EvolveDB as a tool for schema evolution is demonstrated with a running example. The starting point is an already existing MySQL database. (fig. 3) The schema describes traffic routes (table *street*). *Streets* always have a starting point. These starting points are intersections or traffic circles (table *crossroad*). Several roads can start or end at one node. Furthermore, properties (table *property*) can be located at traffic routes. During this tutorial, we are going to make the following changes to the database schema and the associated data:

- In the table *street* we change the type of the column *DATECREATED* changed from *TIMESTAMP* to *DATETIME*.
- We add two new columns, *LONGITUDE : Integer* and *LATITUDE : Integer*, to table *crossroad*.
- We reduce the column size of the *NAME* column in table *street* from 255 to 40.
- We change the single-valued association between property and street into a multi-valued association. (Requires a new cross-reference table.)
- We set 1 as new default value for column *lanes* in table *crossroad*.

## 3.1 Reverse Engineering

In the first step we extract a database model by reverse engineering. For this purpose, open the Eclipse Project Explorer and create a new project. Then select the new project and open the context menu with a right click. Select **EvolveDB** ⇒ **Create MDDE Model...** (Fig. 4)

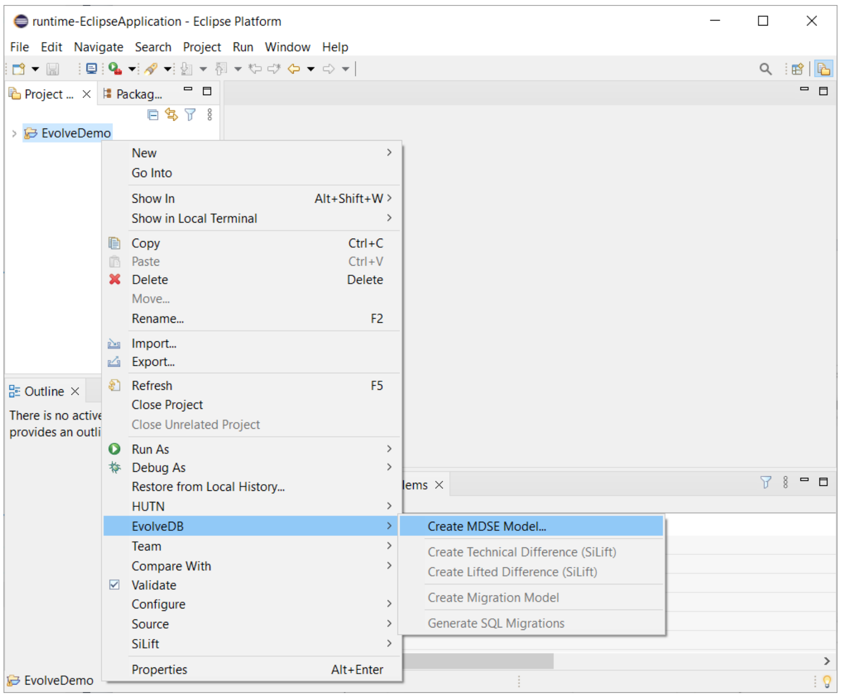
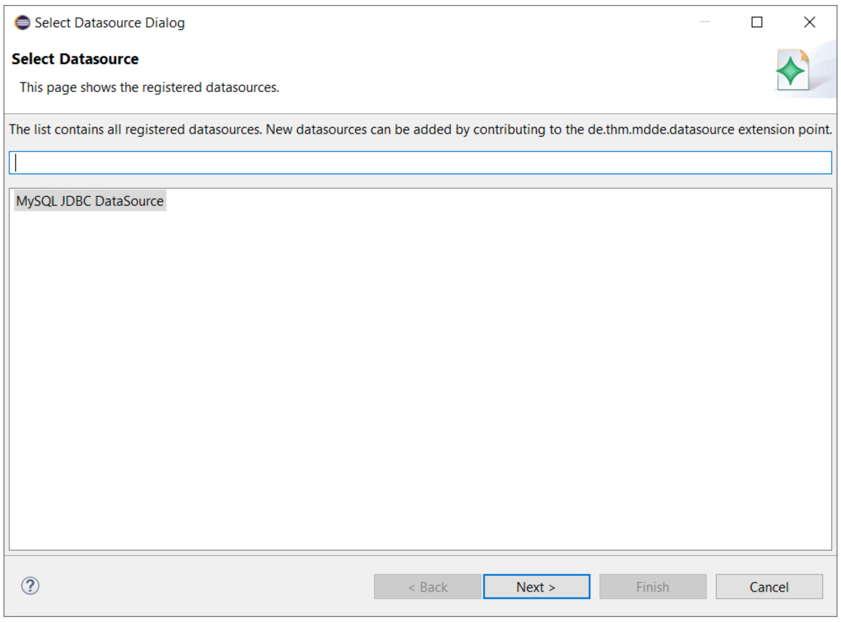


Figure 4: Start reverse engineering

A wizard dialog opens that has several pages. On the first page, we have to select the data source. (fig. 5)


Select Datasource Dialog

**Select Datasource**

This page shows the registered datasources.

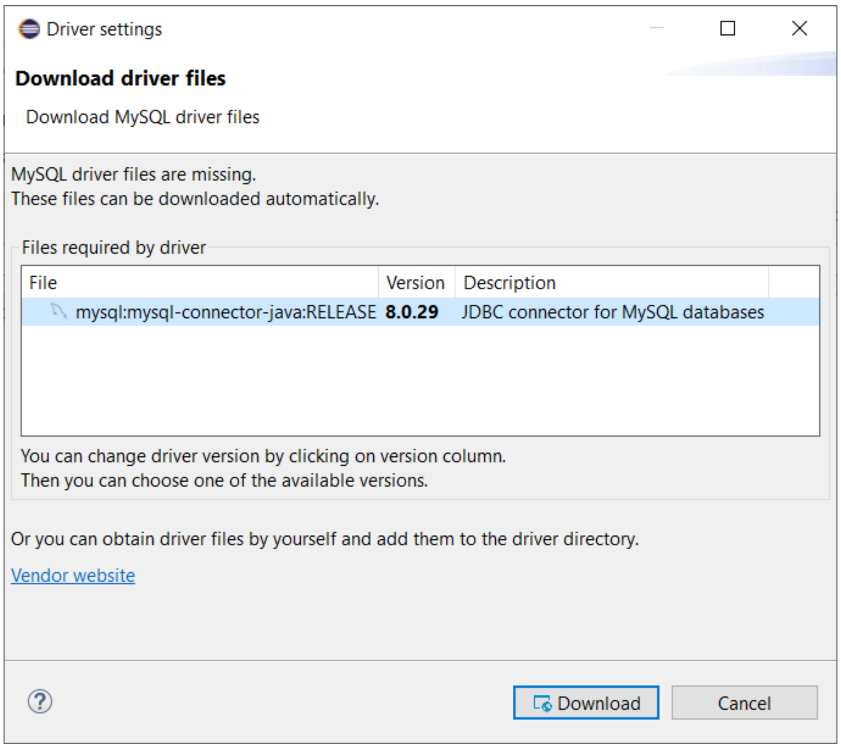
The list contains all registered datasources. New datasources can be added by contributing to the de.thm.mdde.datasource extension point.

MySQL JDBC DataSource

< Back | Next > | Finish | Cancel

Figure 5: Select data source

EvolveDB does not include database drivers. When used for the first time, we must download the JDBC driver. The driver files can be downloaded automatically. Or you can obtain driver files by yourself and add them to the driver directory. The default driver location is the eclipse install directory. The location can be changed in the eclipse preferences. If no suitable driver can be found, EvolveDB opens a download driver dialog. (fig. 6)



MySQL driver files are missing.
These files can be downloaded automatically.

Files required by driver

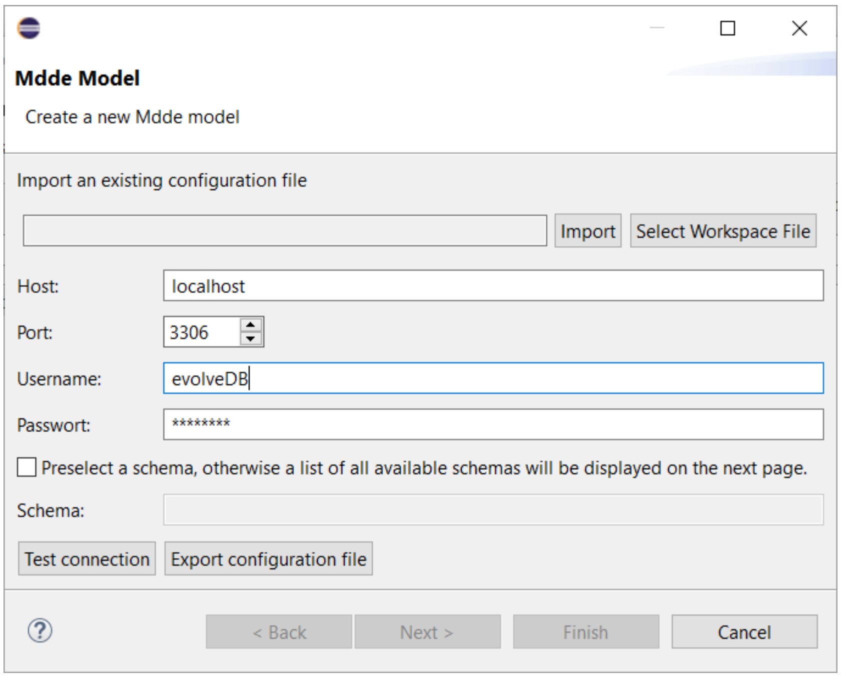
| File | Version | Description |
|---|---|---|
| mysql:mysql-connector-java:RELEASE | **8.0.29** | JDBC connector for MySQL databases |

You can change driver version by clicking on version column.
Then you can choose one of the available versions.

Or you can obtain driver files by yourself and add them to the driver directory.

Figure 6: Download JDBC driver

After loading the driver class, we have to insert the connection information. (fig. 7) Furthermore, it is possible to save the connection information (without password) as an XML configuration file. This configuration file can be imported when creating a new model so that the connection information does not have to be entered again. (fig. 9)) After entering the required information, the connection must be tested. We can continue if the connection is successful. The next page shows a list with all available database schemes if we did not preselect a schema on the first page. (fig. 8) On the last page of the dialog, the new model's name and location must be specified. The dialog can then be closed with *Finish*, and the newly created model is opened in the editor. In addition, a new folder with the name *genModel* and a copy of the model are created in the storage location. The second model is required for the comparison process after the restructuring phase and should not be changed.



**Mdde Model**

Create a new Mdde model

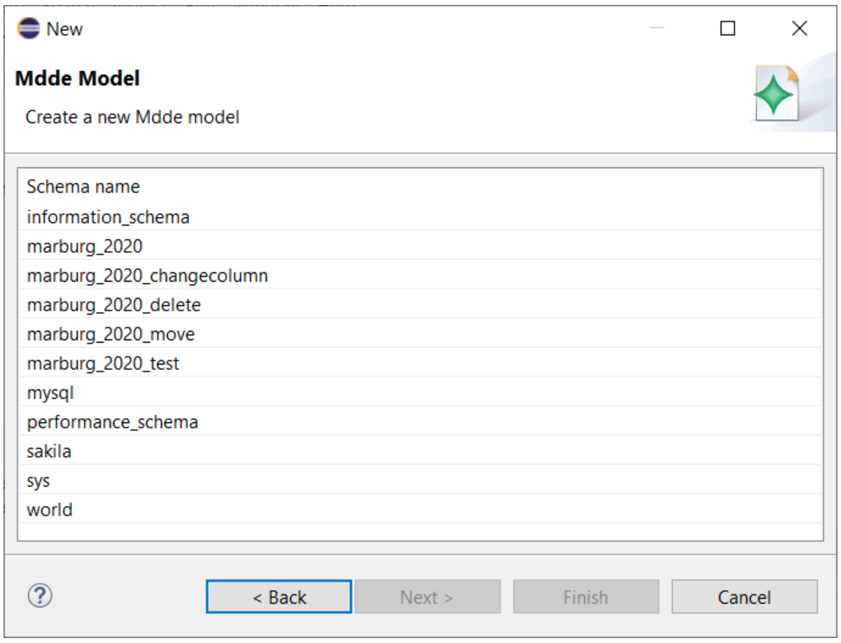
Import an existing configuration file

| | |
|---|---|
| Host: | localhost |
| Port: | 3306 |
| Username: | evolveDB |
| Passwort: | ******** |

☐ Preselect a schema, otherwise a list of all available schemas will be displayed on the next page.

Schema:

Figure 7: Insert connection data



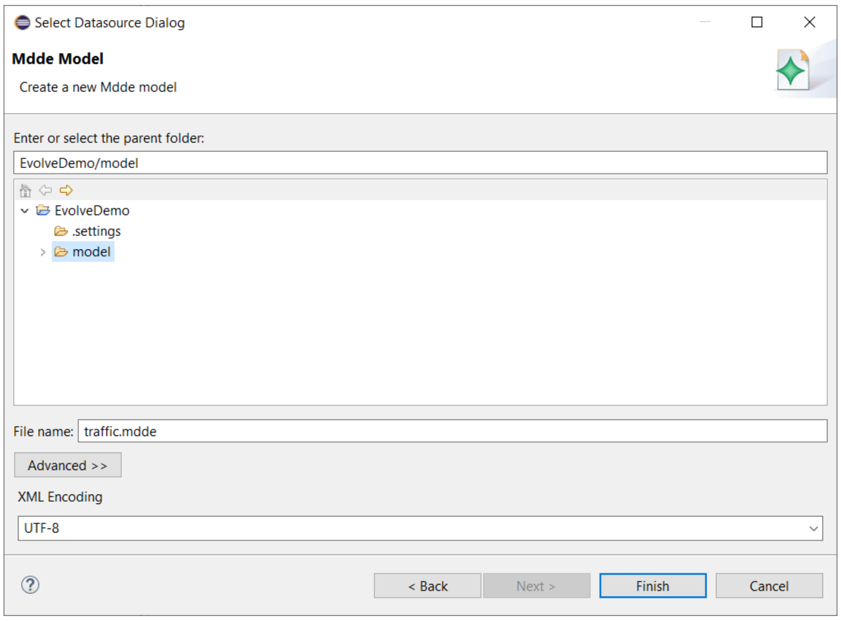
New

**Mdde Model**

Create a new Mdde model

| Schema name |
|---|
| information_schema |
| marburg_2020 |
| marburg_2020_changecolumn |
| marburg_2020_delete |
| marburg_2020_move |
| marburg_2020_test |
| mysql |
| performance_schema |
| sakila |
| sys |
| world |

Figure 8: Select database schema


Select Datasource Dialog

**Mdde Model**

Create a new Mdde model

Enter or select the parent folder:

EvolveDemo/model

EvolveDemo
.settings
model

File name: traffic.mdde

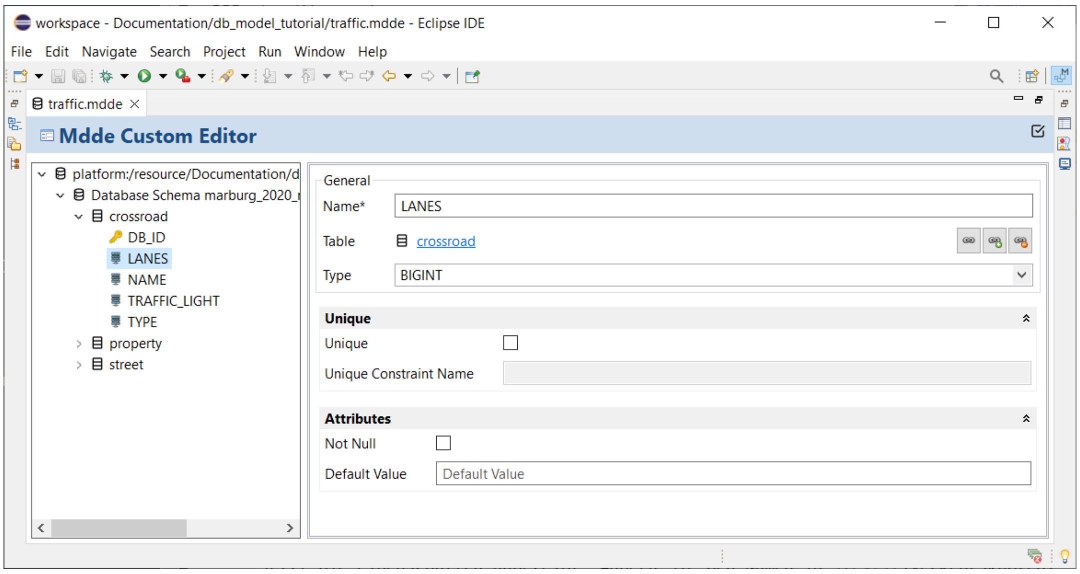
XML Encoding

UTF-8

Figure 9: Choose file location

## 3.2 Restructuring

In the previous step, we created a database model by reverse engineering. The new model was automatically opened in the Eclipse editor (fig. 10). The editor consists of two parts. On the left is a tree-based editor that allows navigating in the model, and on the right is a view with the selected element's properties. Each element in our model corresponds to an element in the database. By expanding an element, the child elements become visible. Now we can start editing the model.


workspace - Documentation/db_model_tutorial/traffic.mdde - Eclipse IDE
File Edit Navigate Search Project Run Window Help
traffic.mdde
Mdde Custom Editor
platform:/resource/Documentation/d
Database Schema marburg_2020_
crossroad
DB_ID
LANES
NAME
TRAFFIC_LIGHT
TYPE
property
street
General
Unique
Attributes

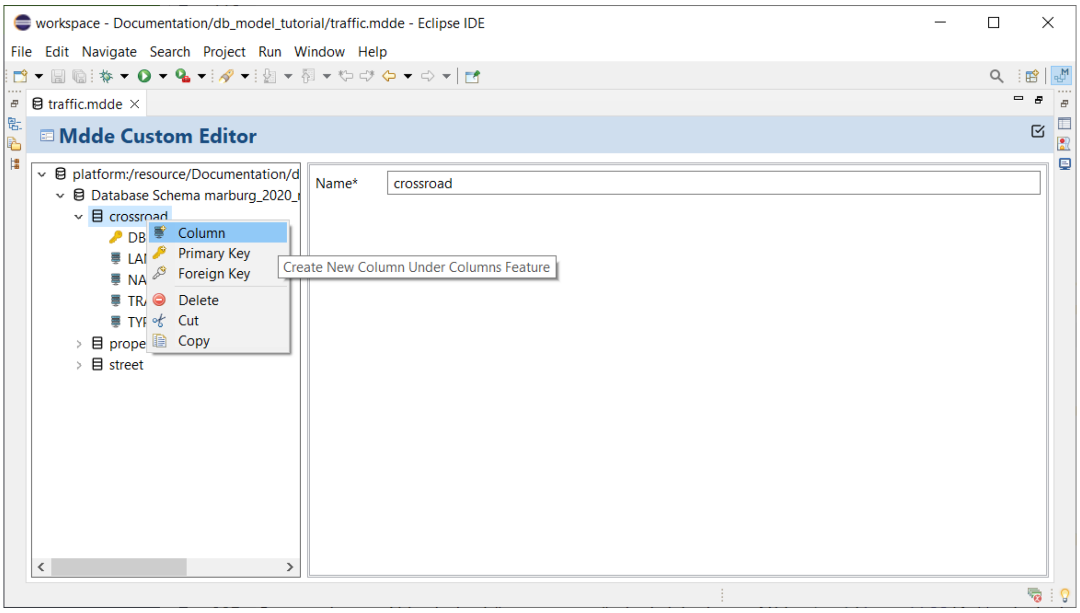
Name* LANES
Table crossroad
Type BIGINT
Unique ☐
Unique Constraint Name
Not Null ☐
Default Value Default Value

Figure 10: MDDE model editor

### 3.2.1 Change Column Type

Select the column *DATECREATED* in table *street*. The attributes of the column are now displayed in the right side of the editor. Change the type from *TIMESTAMP* to *DATETIME*.

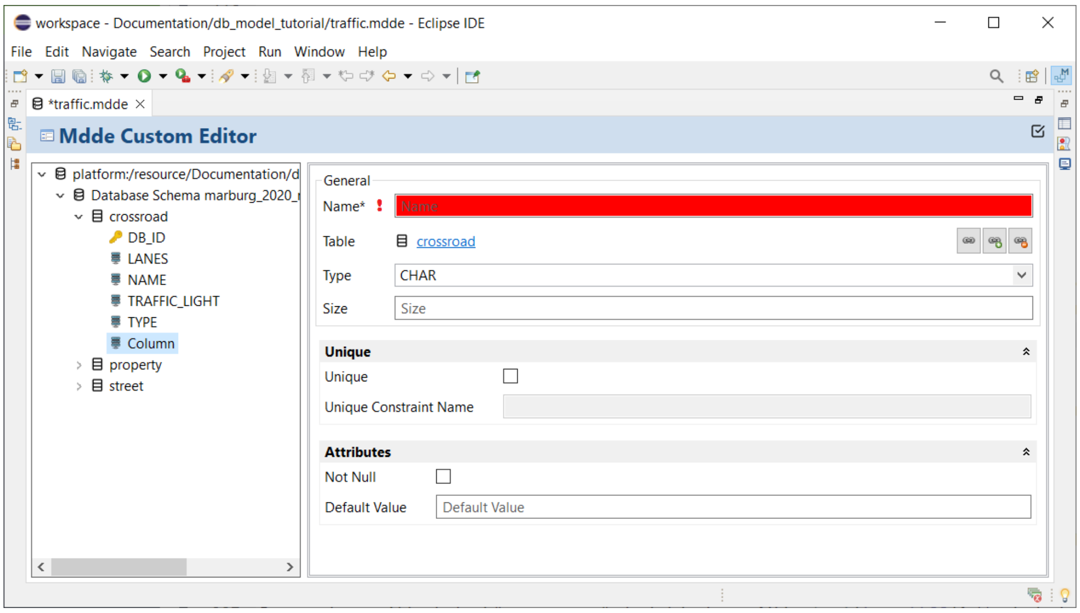
### 3.2.2 Add New Columns


workspace - Documentation/db_model_tutorial/traffic.mdde - Eclipse IDE
File Edit Navigate Search Project Run Window Help
traffic.mdde
Mdde Custom Editor
platform:/resource/Documentation/d
Database Schema marburg_2020_
crossroad
DB
LAI
NA
TR/
TYF
prope
street
Column
Primary Key
Foreign Key
Delete
Cut
Copy
Create New Column Under Columns Feature

Name* crossroad

Figure 11: Context menu

In this step we are going to add the two new columns for longitude and latitude. Select the *crossroad* table and call the context menu with a right-click (fig. 11). Create a new *column*. The new column will then be opened (fig. 12). Mandatory fields are marked in red. Name the new attribute *LATITUDE* and set the type to *Integer*. Since columns with the *Integer* type do not have a *size* attribute, the attribute is hidden. The other attributes can remain unchanged. (fig. 13) Repeat the process for the *LONGITUDE* column.


workspace - Documentation/db_model_tutorial/traffic.mdde - Eclipse IDE
File Edit Navigate Search Project Run Window Help
*traffic.mdde
Mdde Custom Editor

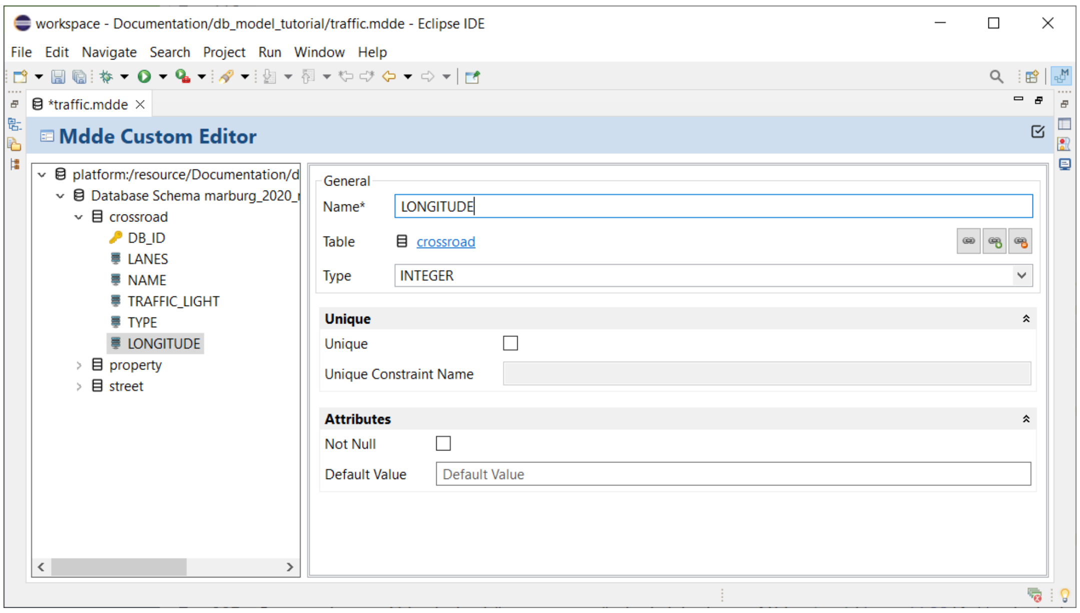
platform:/resource/Documentation/d
Database Schema marburg_2020_
crossroad
DB_ID
LANES
NAME
TRAFFIC_LIGHT
TYPE
Column
property
street

General
Name* | Name
Table | crossroad
Type | CHAR
Size | Size

Unique
Unique | ☐
Unique Constraint Name |

Attributes
Not Null | ☐
Default Value | Default Value

Figure 12: Mandatory fields are marked in red


workspace - Documentation/db_model_tutorial/traffic.mdde - Eclipse IDE
File Edit Navigate Search Project Run Window Help
*traffic.mdde
Mdde Custom Editor

platform:/resource/Documentation/d
Database Schema marburg_2020_
crossroad
DB_ID
LANES
NAME
TRAFFIC_LIGHT
TYPE
LONGITUDE
property
street

General
Name* | LONGITUDE
Table | crossroad
Type | INTEGER

Unique
Unique | ☐
Unique Constraint Name |

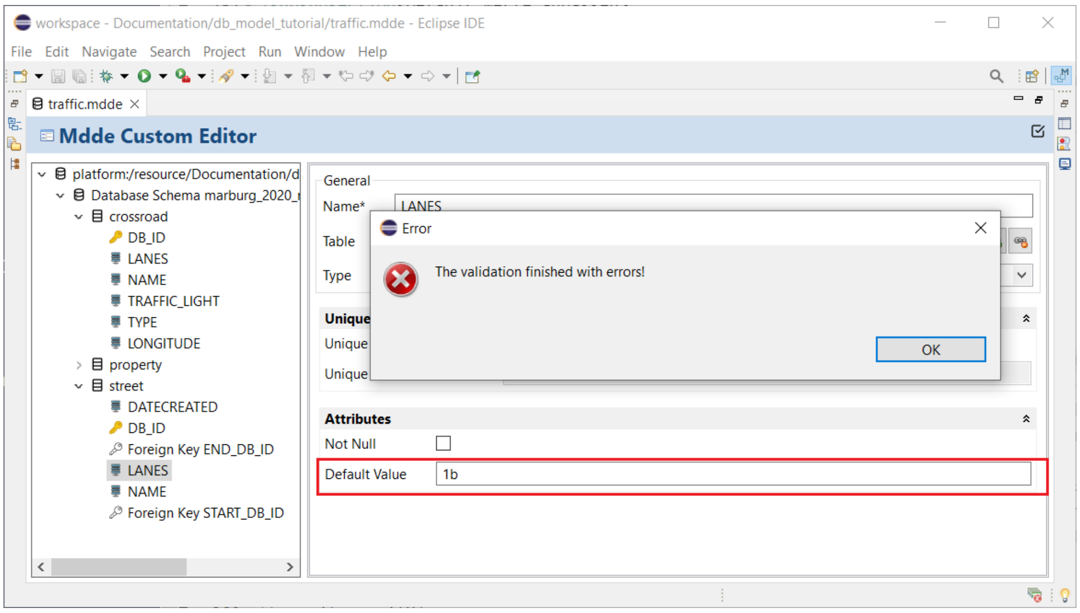
Attributes
Not Null | ☐
Default Value | Default Value

Figure 13: The *Size* attribute is not visible

### 3.2.3 Change Default Values

Open the *crossroad* table in the tree viewer and select the column *LANES*. Set the default value to 1. Basically, this change is no different from a name change or a type change. Therefore, you may wonder at this point why this change is listed and explained in this tutorial. The reason is one of the significant strengths of EvolveDB. We explain the details in the following section.

**Explanation: Default-Value, Size and Validation** Each column in a MySQL database can have a default value. However, this default value must match the column type. If we consider the column LANES as an example, the default value may only consist of digits because the type is a numeric data type. However, this is not the only restriction. At the same time, the default value must not exceed the value range of the data type. The *Integer* data type has a fixed size. Therefore, EvolveDB hides the size attribute for columns with the *Integer* data type. Columns of type Integer can contain values from the range between -2147483648 and 2147483647. When we specify a value outside the value range or enter invalid characters, we receive an error message when validating the model (fig. 14). The validation can be triggered manually using the button in the top right corner of the editor.



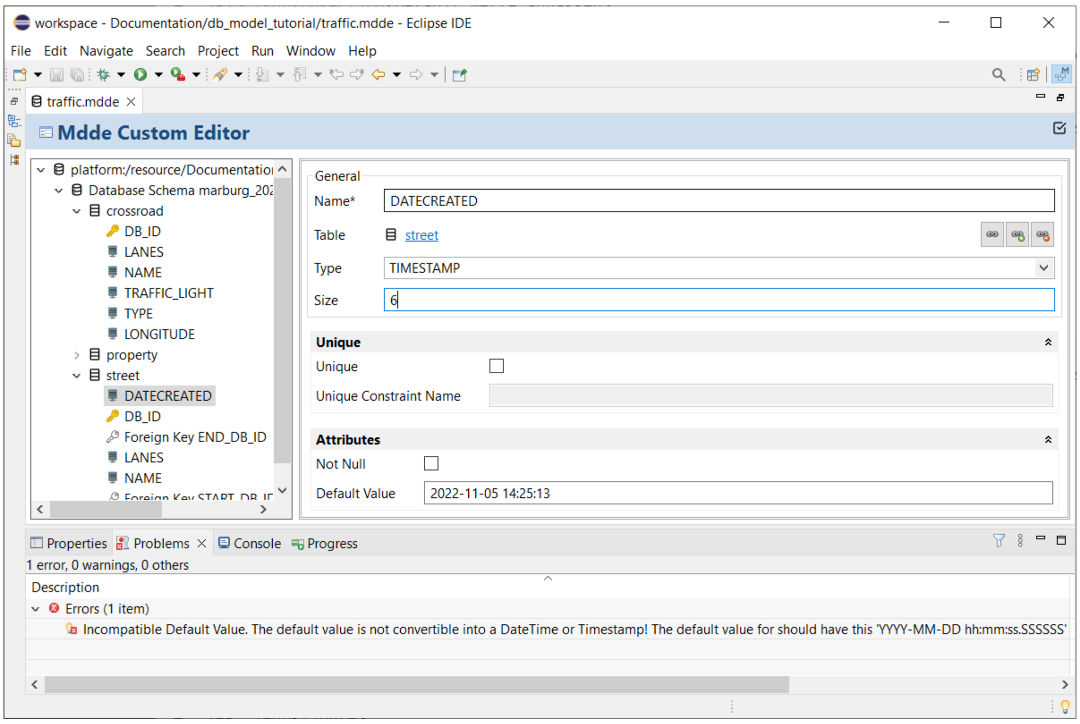
Name* LANES

The validation finished with errors!

Not Null ☐

Default Value 1b

Figure 14: The default value is invalid.


workspace - Documentation/db_model_tutorial/traffic.mdde - Eclipse IDE
File Edit Navigate Search Project Run Window Help
traffic.mdde
Mdde Custom Editor
Properties
Problems
Console
Progress

platform:/resource/Documentatio
Database Schema marburg_202
crossroad
DB_ID
LANES
NAME
TRAFFIC_LIGHT
TYPE
LONGITUDE
property
street
DATECREATED
DB_ID
Foreign Key END_DB_ID
LANES
NAME

General
Name* DATECREATED
Table street
Type TIMESTAMP
Size 6

Unique
Unique ☐
Unique Constraint Name

Attributes
Not Null ☐
Default Value 2022-11-05 14:25:13

1 error, 0 warnings, 0 others
Description
Errors (1 item)
Incompatible Default Value. The default value is not convertible into a DateTime or Timestamp! The default value for should have this 'YYYY-MM-DD hh:mm:ss.SSSSSS'

Figure 15: Default-value for a timestamp column

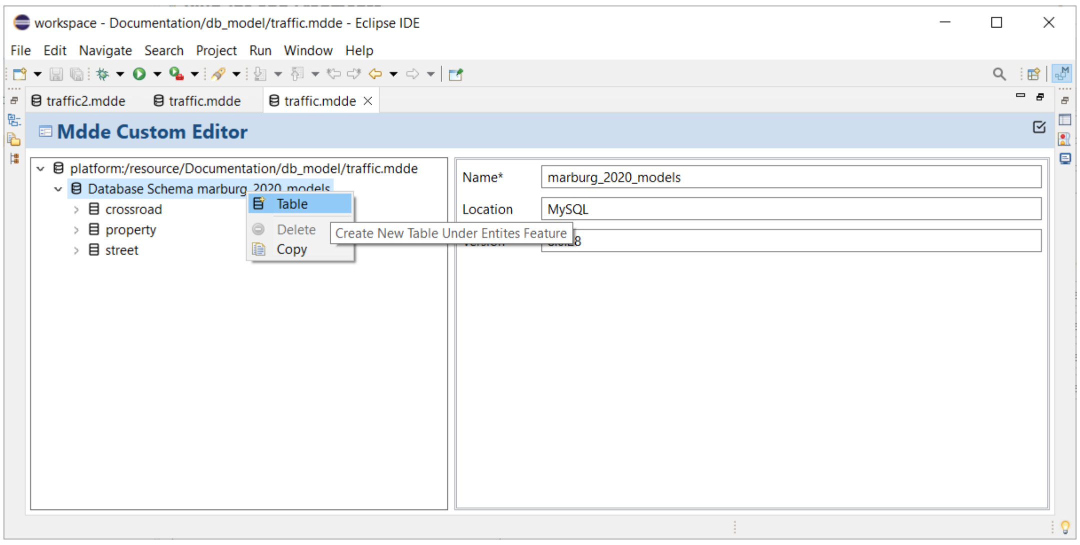
Let us consider another example. MySQL allows fractional seconds for the *TIME*, *DATETIME* and *TIMESTAMP* data types with an accuracy of up to microseconds (6 digits). When validating the default value of a column with one of these three types, the default value must match the *Size* attribute (fig. 15). In summary, a simple change to the default value or size attribute requires extensive validation. These validations ensure that only permissible values can be entered.

### 3.2.4 Reduce Column Size

Select the column *NAME* in table *street*. The column has the datatype *VARCHAR*. Unlike numeric data types like *INTEGER*, *VARCHAR* has no fixed size. For this reason, the size attribute is visible, and we can select a value in the range between 0 and 255. Currently, 255 is selected. Decrease the value to 40.

### 3.2.5 Change the Multiplicity of an Association

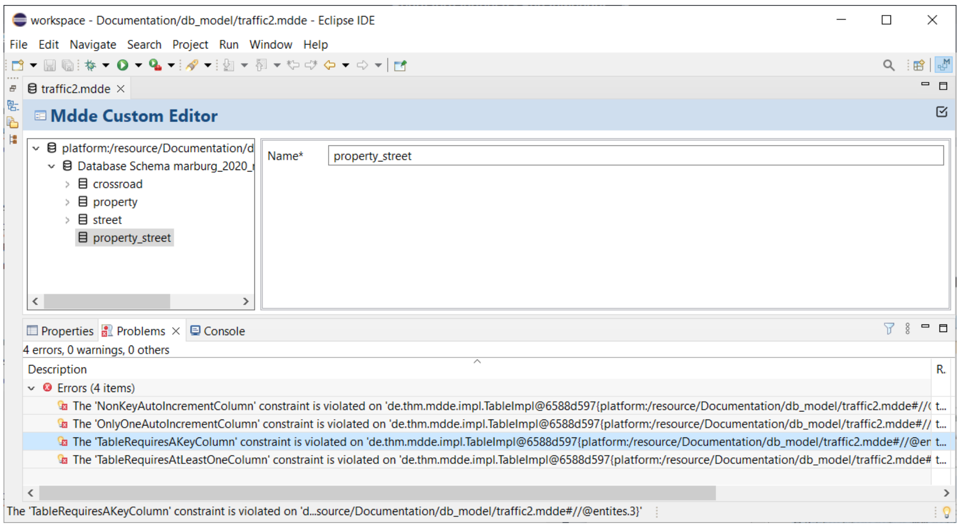
Right-clicking on the database schema within the tree-based editor opens the context menu. Via the context menu, we can add new tables(fig. 16). In the previous sections, the extensive validations have already been discussed. However, not only individual columns or attribute values are validated, but also entire tables and the model as a whole. After we add the new table *property_street*, the validation of the model leads to several errors (fig. 17). The model is invalid, because we have not added any columns yet.


workspace - Documentation/db_model/traffic.mdde - Eclipse IDE
File Edit Navigate Search Project Run Window Help
traffic2.mdde
traffic.mdde
traffic.mdde
Mdde Custom Editor
platform:/resource/Documentation/db_model/traffic.mdde
Database Schema marburg_2020_models
crossroad
property
street
Table
Delete
Copy
Create New Table Under Entites Feature

Name* marburg_2020_models

Location MySQL

Figure 16: Add new table

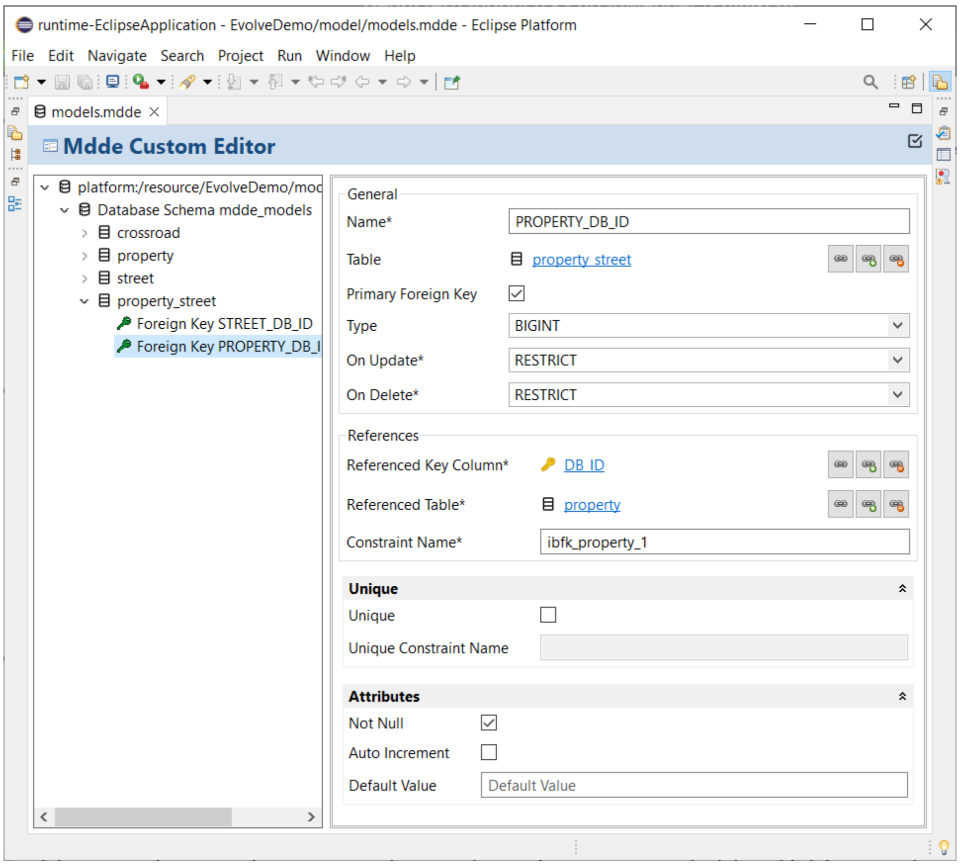

workspace - Documentation/db_model/traffic2.mdde - Eclipse IDE
File Edit Navigate Search Project Run Window Help
traffic2.mdde
Mdde Custom Editor
platform:/resource/Documentation/d
Database Schema marburg_2020_
crossroad
property
street
property_street
Properties
Problems
Console
Description
R.

Name* property_street

```
4 errors, 0 warnings, 0 others
Errors (4 items)
The 'NonKeyAutoIncrementColumn' constraint is violated on 'de.thm.mdde.impl.TableImpl@6588d597{platform:/resource/Documentation/db_model/traffic2.mdde#//( t...
The 'OnlyOneAutoIncrementColumn' constraint is violated on 'de.thm.mdde.impl.TableImpl@6588d597{platform:/resource/Documentation/db_model/traffic2.mdde#// t...
The 'TableRequiresAKeyColumn' constraint is violated on 'de.thm.mdde.impl.TableImpl@6588d597{platform:/resource/Documentation/db_model/traffic2.mdde#//@en t...
The 'TableRequiresAtLeastOneColumn' constraint is violated on 'de.thm.mdde.impl.TableImpl@6588d597{platform:/resource/Documentation/db_model/traffic2.mdde# t...
The 'TableRequiresAKeyColumn' constraint is violated on 'd...source/Documentation/db_model/traffic2.mdde#//@entites.3}'
```

Figure 17: Model validation

Tables must have at least one column and a primary key. By double-clicking on the error, the model element that caused the error is selected. Now we add the two Foreign Keys to the table. The two foreign keys reference the tables *street* and *property*. (It is also possible to move the *STREET_DB_ID* to table *property_street*.) Then the old foreign key *street_db_id* in the table *property* can be deleted. Figure 18 shows the the result. Once the validation of the model is successful, the comparison of the model versions can be continued in the next step.


runtime-EclipseApplication - EvolveDemo/model/models.mdde - Eclipse Platform

File Edit Navigate Search Project Run Window Help

models.mdde

Mdde Custom Editor

- platform:/resource/EvolveDemo/moc
  - Database Schema mdde_models
    - crossroad
    - property
    - street
    - property_street
      - Foreign Key STREET_DB_ID
      - Foreign Key PROPERTY_DB_I

General

Name*: PROPERTY_DB_ID
Table: property street
Primary Foreign Key: ✓
Type: BIGINT
On Update*: RESTRICT
On Delete*: RESTRICT

References

Referenced Key Column*: DB ID
Referenced Table*: property
Constraint Name*: ibfk_property_1

Unique

Unique: ☐
Unique Constraint Name:

Attributes

Not Null: ✓
Auto Increment: ☐
Default Value: Default Value

Figure 18: Table *property_street* with foreign keys

## 3.3 Model Comparison

EvolveDB analyzes the differences between the two model versions. At first, a sequence of atomic differences is automatically derived. Then the identified model evolution steps are (semi-)automatically mapped to predefined migration operations. These migration operations can finally be applied to the original database schema to transform it into the target schema. For the comparison process, EvolveDB uses the *SiLift* framework. *SiLift* is a generic model comparison environment that can semantically lift identified low-level differences of EMF-based models into representations of user-level edit operations.

### 3.3.1 SiLift

SiLift's approach can best be compared to a four-step pipeline, as shown in Figure 19. The two model versions, e.g. models.mdde and models2.mdde serve as input:

1. **Matching:** The task of a matcher is to identify the corresponding elements from model A and model B, i.e., the elements that match in both models. SiLift provides four different matchers by default. One of them is EMFCompare.

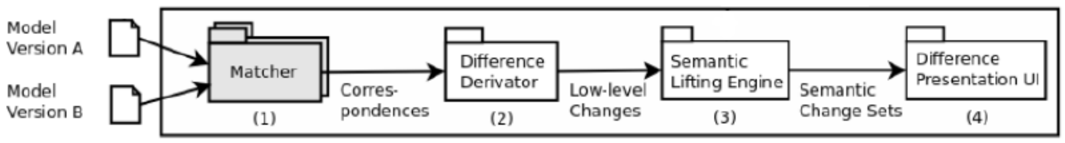
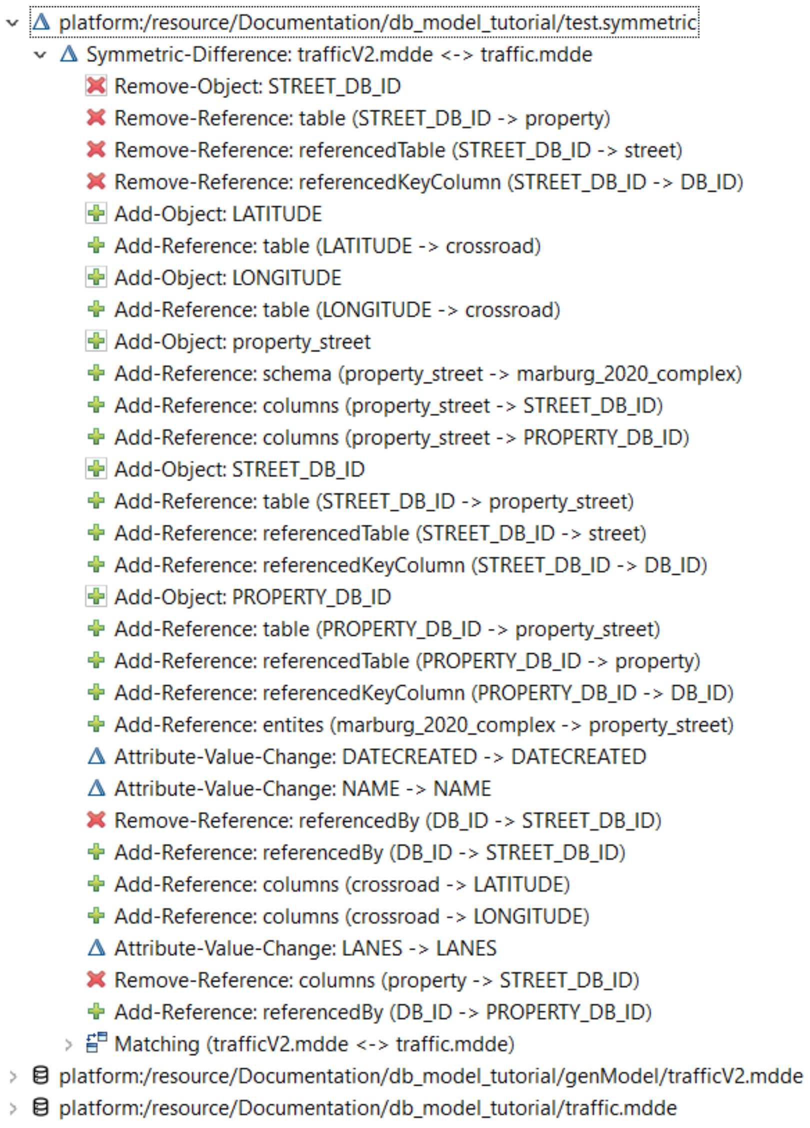


Figure 19: SiLift processing pipeline

2. **Difference Derivation:** Based on the correspondences, the difference derivator calculates a technical difference (low-level difference) between the model versions. All objects and references for which no correspondence exists must either have been added to model B or removed from model A.



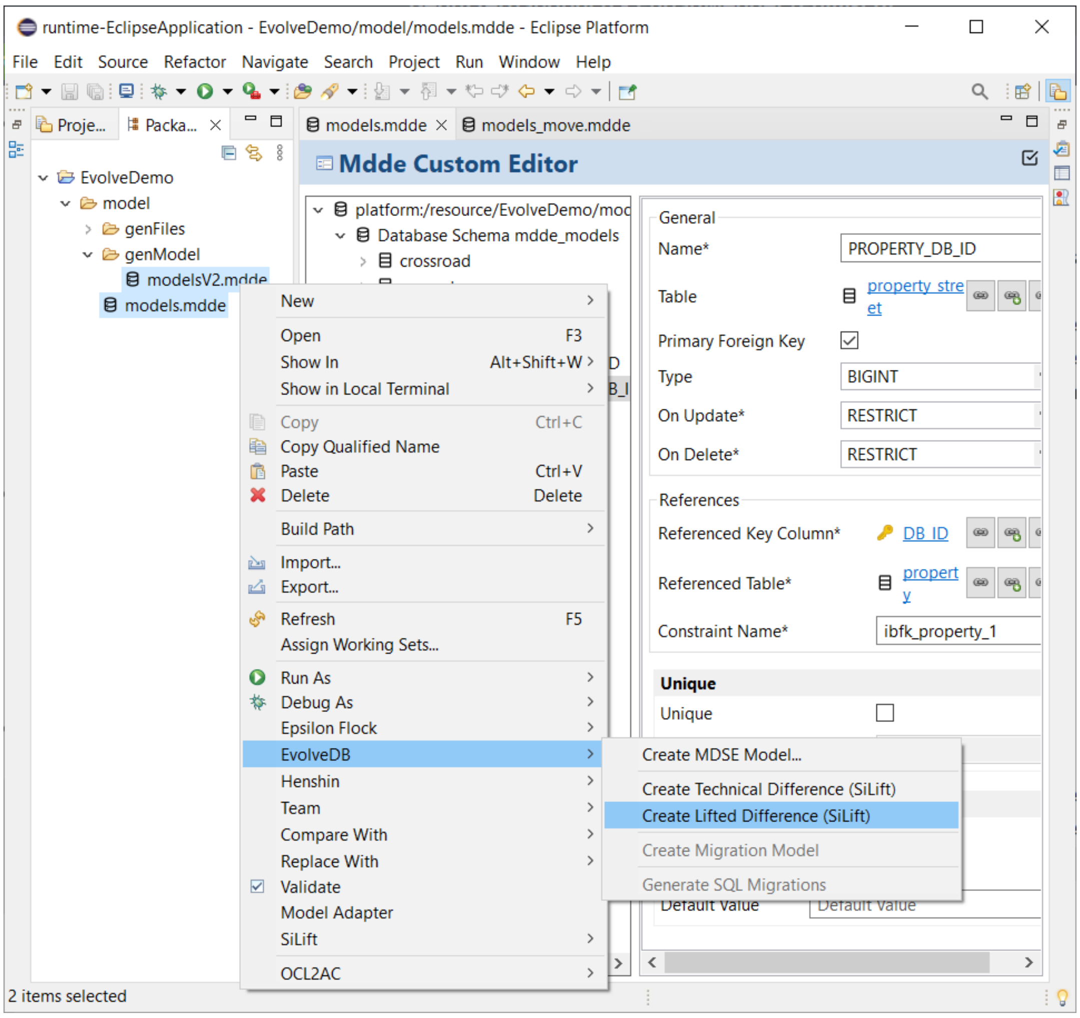
```
platform:/resource/Documentation/db_model_tutorial/test.symmetric
  Symmetric-Difference: trafficV2.mdde <-> traffic.mdde
    Remove-Object: STREET_DB_ID
    Remove-Reference: table (STREET_DB_ID -> property)
    Remove-Reference: referencedTable (STREET_DB_ID -> street)
    Remove-Reference: referencedKeyColumn (STREET_DB_ID -> DB_ID)
    Add-Object: LATITUDE
    Add-Reference: table (LATITUDE -> crossroad)
    Add-Object: LONGITUDE
    Add-Reference: table (LONGITUDE -> crossroad)
    Add-Object: property_street
    Add-Reference: schema (property_street -> marburg_2020_complex)
    Add-Reference: columns (property_street -> STREET_DB_ID)
    Add-Reference: columns (property_street -> PROPERTY_DB_ID)
    Add-Object: STREET_DB_ID
    Add-Reference: table (STREET_DB_ID -> property_street)
    Add-Reference: referencedTable (STREET_DB_ID -> street)
    Add-Reference: referencedKeyColumn (STREET_DB_ID -> DB_ID)
    Add-Object: PROPERTY_DB_ID
    Add-Reference: table (PROPERTY_DB_ID -> property_street)
    Add-Reference: referencedTable (PROPERTY_DB_ID -> property)
    Add-Reference: referencedKeyColumn (PROPERTY_DB_ID -> DB_ID)
    Add-Reference: entites (marburg_2020_complex -> property_street)
    Attribute-Value-Change: DATECREATED -> DATECREATED
    Attribute-Value-Change: NAME -> NAME
    Remove-Reference: referencedBy (DB_ID -> STREET_DB_ID)
    Add-Reference: referencedBy (DB_ID -> STREET_DB_ID)
    Add-Reference: columns (crossroad -> LATITUDE)
    Add-Reference: columns (crossroad -> LONGITUDE)
    Attribute-Value-Change: LANES -> LANES
    Remove-Reference: columns (property -> STREET_DB_ID)
    Add-Reference: referencedBy (DB_ID -> PROPERTY_DB_ID)
  > Matching (trafficV2.mdde <-> traffic.mdde)
> platform:/resource/Documentation/db_model_tutorial/genModel/trafficV2.mdde
> platform:/resource/Documentation/db_model_tutorial/traffic.mdde
```

Figure 20: Technical difference between models.mdde and models2.mdde

3. **Semantic Lifting:** The previously calculated technical difference contains all changes between the models. Then these changes are semantically lifted. For this purpose, the low-level changes are grouped into semantic change sets with the help of recognition rules. Each semantic change set represents an editing operation performed by the user. Often, user editing operations consist of more than one low-level change, because even basic edit operations that appear simple from a users point of view may lead to many low-level changes. While atomic rules cover the creation, deletion, moving of elements, and the changing of attribute values, the complex editing rules are usually composed of atomic and other complex rules.

4. **Difference Presentation UI:** The result is a symmetric difference model which contains the matching, the atomic changes and the resulting semantic change sets.

### 3.3.2 Model Compoarison with EvolveDB



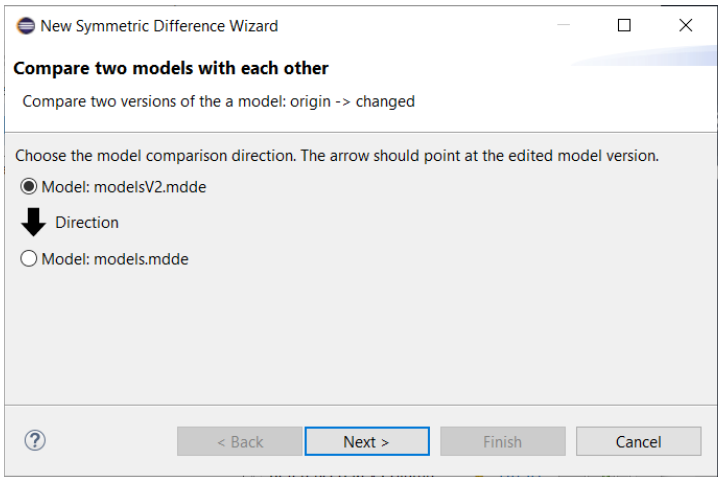
General
Name* PROPERTY_DB_ID
Table property_street
Primary Foreign Key ✓
Type BIGINT
On Update* RESTRICT
On Delete* RESTRICT

References
Referenced Key Column* DB_ID
Referenced Table* property
Constraint Name* ibfk_property_1

Unique
Unique ☐
Default Value Default Value

Figure 21: Start comparison

We start comparing the two model versions from the previous chapter. Therefore, we select the two **mdde** files in the Package or Project Explorer and open the context menu with a right click. We choose **EvolveDB** ⇒ **Create Lifted Difference (SiLift)** (fig. 21) A multi-page wizard dialog opens. On the first page we must select the model comparison direction. The arrow has to point at the edited model version(fig. 22).



New Symmetric Difference Wizard

**Compare two models with each other**

Compare two versions of the a model: origin -> changed

Choose the model comparison direction. The arrow should point at the edited model version.

(●) Model: modelsV2.mdde

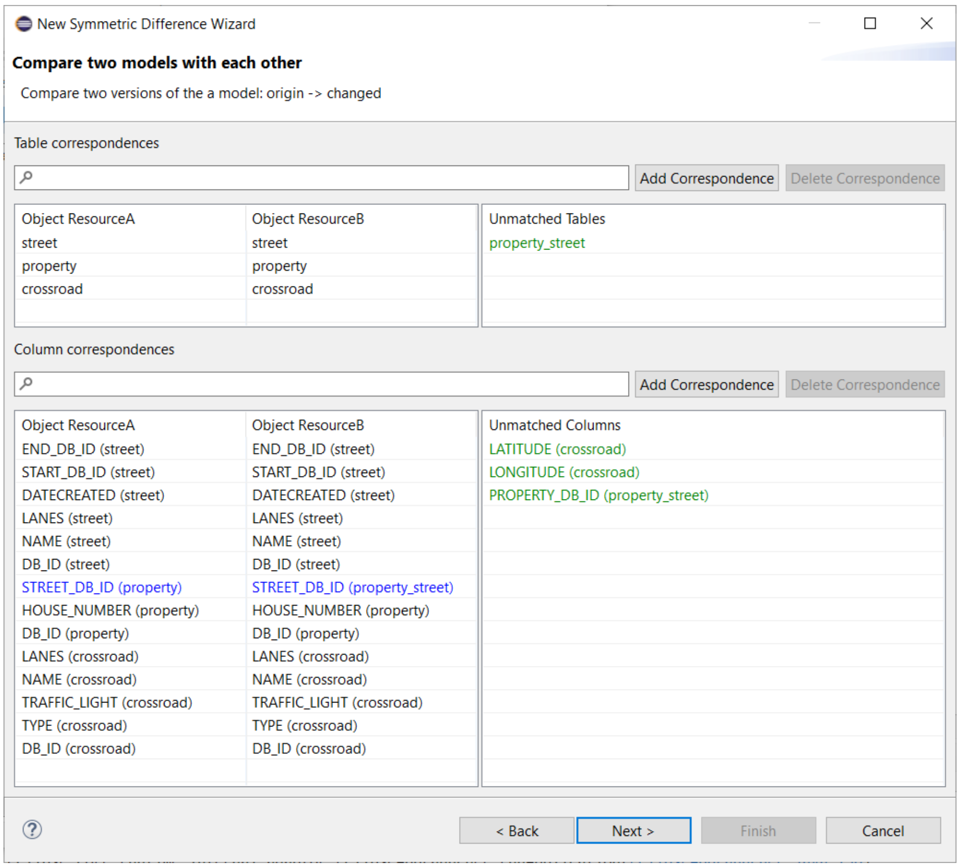
Direction

( ) Model: models.mdde

Figure 22: Choose model comparison direction

On the second page (fig. 23) the matching identified by SiLift is presented. The upper table shows the corresponding tables from model A and model B. Unmatched tables which have no correspondence must either have been added to model B (marked in green) or removed from model A (marked in red). The tables at the bottom show the corresponding column. If a column was moved or renamed, the column is marked blue. For example, the foreign key *STREET_DB_ID* was moved to the table *property_street*. If an identified correspondence is not correct, we can delete or add correspondences.

A file name for the symmetric difference model must be specified on the last page of the dialog (fig. 24). Furthermore, we can create a so-called migration model. The migration model will be required later in the process. For this reason, it makes sense to create it directly. The migration model will be stored at the same location as the symmetric difference model.



New Symmetric Difference Wizard

**Compare two models with each other**

Compare two versions of the a model: origin -> changed

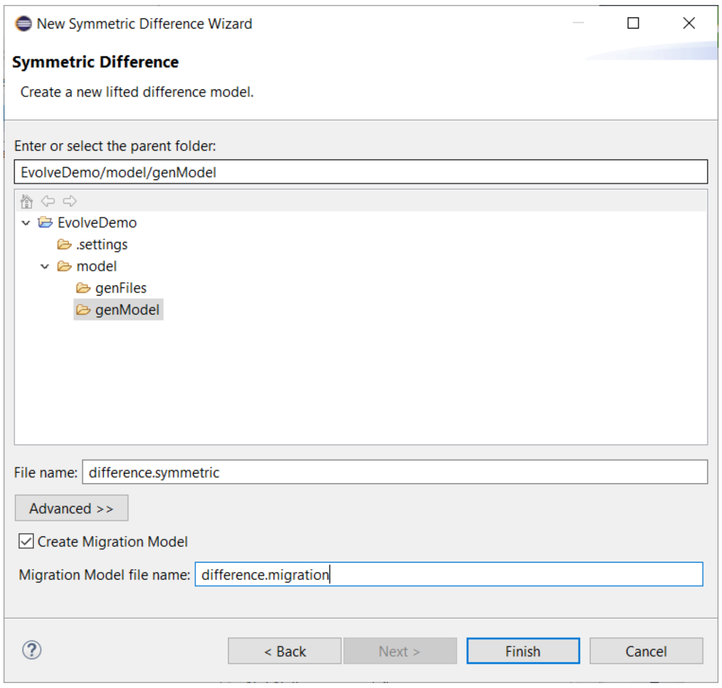
Table correspondences

| Object ResourceA | Object ResourceB | Unmatched Tables |
|---|---|---|
| street | street | property_street |
| property | property | |
| crossroad | crossroad | |

Column correspondences

| Object ResourceA | Object ResourceB | Unmatched Columns |
|---|---|---|
| END_DB_ID (street) | END_DB_ID (street) | LATITUDE (crossroad) |
| START_DB_ID (street) | START_DB_ID (street) | LONGITUDE (crossroad) |
| DATECREATED (street) | DATECREATED (street) | PROPERTY_DB_ID (property_street) |
| LANES (street) | LANES (street) | |
| NAME (street) | NAME (street) | |
| DB_ID (street) | DB_ID (street) | |
| STREET_DB_ID (property) | STREET_DB_ID (property_street) | |
| HOUSE_NUMBER (property) | HOUSE_NUMBER (property) | |
| DB_ID (property) | DB_ID (property) | |
| LANES (crossroad) | LANES (crossroad) | |
| NAME (crossroad) | NAME (crossroad) | |
| TRAFFIC_LIGHT (crossroad) | TRAFFIC_LIGHT (crossroad) | |
| TYPE (crossroad) | TYPE (crossroad) | |
| DB_ID (crossroad) | DB_ID (crossroad) | |

Figure 23: Choose model comparison direction



New Symmetric Difference Wizard

**Symmetric Difference**

Create a new lifted difference model.

Enter or select the parent folder: EvolveDemo/model/genModel

- EvolveDemo
  - .settings
  - model
    - genFiles
    - genModel

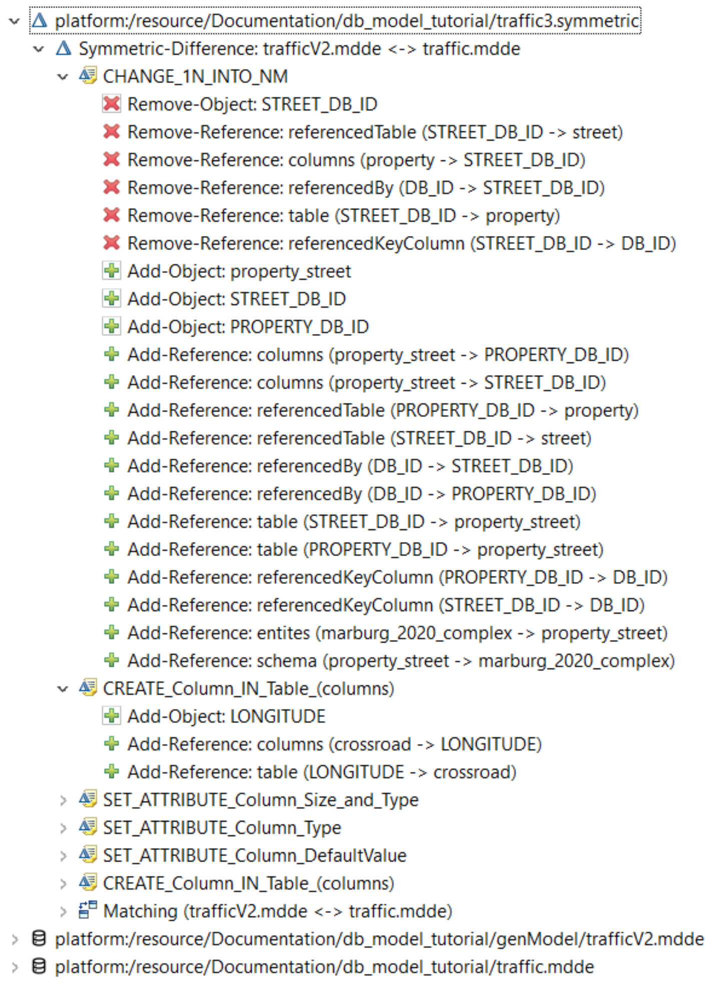
File name: difference.symmetric

☑ Create Migration Model

Migration Model file name: difference.migration

Figure 24: Choose file location

The symmetric difference model contains the matching, the atomic changes and the resulting semantic change sets. Figure 25 shows the symmetric difference model for our example opened in the symmetric difference model editor. All atomic changes are displayed when we expand one of the semantic change sets. In our example (fig. 25) the complex operator *CHANGE_1N_INTO_NM*, representing the multiplicity change, consists of 21 atomic changes.



```
platform:/resource/Documentation/db_model_tutorial/traffic3.symmetric
  Symmetric-Difference: trafficV2.mdde <-> traffic.mdde
    CHANGE_1N_INTO_NM
      Remove-Object: STREET_DB_ID
      Remove-Reference: referencedTable (STREET_DB_ID -> street)
      Remove-Reference: columns (property -> STREET_DB_ID)
      Remove-Reference: referencedBy (DB_ID -> STREET_DB_ID)
      Remove-Reference: table (STREET_DB_ID -> property)
      Remove-Reference: referencedKeyColumn (STREET_DB_ID -> DB_ID)
      Add-Object: property_street
      Add-Object: STREET_DB_ID
      Add-Object: PROPERTY_DB_ID
      Add-Reference: columns (property_street -> PROPERTY_DB_ID)
      Add-Reference: columns (property_street -> STREET_DB_ID)
      Add-Reference: referencedTable (PROPERTY_DB_ID -> property)
      Add-Reference: referencedTable (STREET_DB_ID -> street)
      Add-Reference: referencedBy (DB_ID -> STREET_DB_ID)
      Add-Reference: referencedBy (DB_ID -> PROPERTY_DB_ID)
      Add-Reference: table (STREET_DB_ID -> property_street)
      Add-Reference: table (PROPERTY_DB_ID -> property_street)
      Add-Reference: referencedKeyColumn (PROPERTY_DB_ID -> DB_ID)
      Add-Reference: referencedKeyColumn (STREET_DB_ID -> DB_ID)
      Add-Reference: entites (marburg_2020_complex -> property_street)
      Add-Reference: schema (property_street -> marburg_2020_complex)
    CREATE_Column_IN_Table_(columns)
      Add-Object: LONGITUDE
      Add-Reference: columns (crossroad -> LONGITUDE)
      Add-Reference: table (LONGITUDE -> crossroad)
    SET_ATTRIBUTE_Column_Size_and_Type
    SET_ATTRIBUTE_Column_Type
    SET_ATTRIBUTE_Column_DefaultValue
    CREATE_Column_IN_Table_(columns)
    Matching (trafficV2.mdde <-> traffic.mdde)
platform:/resource/Documentation/db_model_tutorial/genModel/trafficV2.mdde
platform:/resource/Documentation/db_model_tutorial/traffic.mdde
```

Figure 25: Lifted difference between traffic.mdde and traffic2.mdde

If we recall the changes we did previously, the semantic change sets can be assigned to the editing operations as follows:

- **SET_ATTRIBUTE_Column_Size_and_Type**: Reduce the column *size* of *NAME* in table *street* from 255 to 40.
- **CREATE_Column_IN_Table_(columns)**: Create the new column *LONGITUDE*: in table *crossroad*.

- **CREATE_Column_IN_Table_(columns)** Create the new column *LATITUDE* in table *crossroad*.
- **CHANGE_1N_INTO_NM**: Change the single-valued association between property and street into a multi-valued association. (Add a new cross-reference table.) This this operator includes both the creation of the new table and the deletion of the old foreign key.
- **SET_ATTRIBUTE_Column_Type**: Change the column type of*DATECREATED* from *TIMESTAMP* to *DATETIME*.
- **SET_ATTRIBUTE_Column_DefaultValue**: Change the default value for column *LANES*.

All currently supported operators are listed in table 1.

| **Name** | **Classification** |
|---|---|
| Rename Element | VALUE CHANGE |
| Make ForeignKey a PrimaryForeignKey | VALUE CHANGE |
| Set foreign key on update constraint | VALUE CHANGE |
| Set foreign key on delete constraint | VALUE CHANGE |
| Set foreign key constraint name | VALUE CHANGE |
| Make column unique | VALUE CHANGE |
| Set column unique constraint name | VALUE CHANGE |
| Set column type | VALUE CHANGE |
| Set column size | VALUE CHANGE |
| Set column not null | VALUE CHANGE |
| Set column default value | VALUE CHANGE |
| Set column auto-increment attribute | VALUE CHANGE |
| Drop primary key | REDUCE |
| Drop table | REDUCE |
| Drop foreign key | REDUCE |
| Drop column | REDUCE |
| Create primary key | INCREASE |
| Create foreign key | INCREASE |
| Create table | INCREASE |
| Create column | INCREASE |
| Create many-to-many table | STRUCTURAL CHANGE |
| Move column | STRUCTURAL CHANGE |
| Dissolve many-to-many table | STRUCTURAL CHANGE |
| Join table | STRUCTURAL CHANGE |

Table 1: Supported edit operations

## 3.4 Forward Engineering

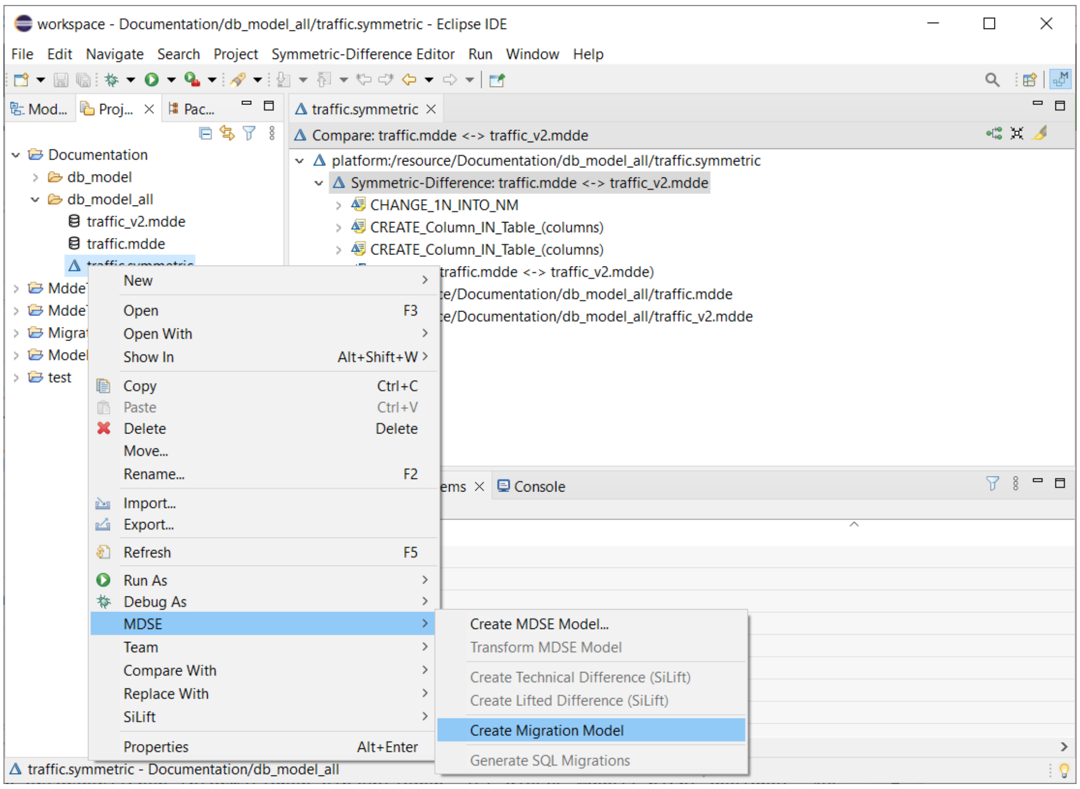
In the next step, we will transform the edit operations into a SQL migration script. An editing operation can be non-breaking, breaking & resolvable, or breaking & unresolvable.

- **non-breaking**: Non-breaking changes can be resolved automatically. Most additions belong to this group.
- **breaking and resolvable**: Breaking and resolvable changes break the conformance of existing data, although they can be automatically adapted.
- **breaking and unresolvable**: Breaking and unresolvable changes break the conformity of existing data which cannot be automatically adjusted and require user intervention. These migrations often require the user to specify how the data migration should be performed. For example, if the column size is reduced, it is necessary to specify how to deal with too large elements.

Changes in the first category are not relevant for the migration problem because data migration is not necessary. Changes placed into the second category can be resolved without user intervention but require data migration. The third category of changes requires user attention. The user has to provide additional information necessary to migrate the affected data.

### 3.4.1 Migration Model

The symmetric difference model we created in the previous section is only used to represent the difference between the two model versions. The difference model is unsuitable for adding the additional information required for the migration. For this reason, we convert the *difference.symmetric* model into a migration model (If this has not yet been done in the previous step). Therefore, we select the model in the Project or Package Explorer and open the menu with a right-click. Then we choose **EvolveDB** ⇒ **Create Migration Model...** (Abb. 26) The migration model is stored in the same folder as the difference model. If the model is not visible immediately, the project or the folder must be refreshed once.

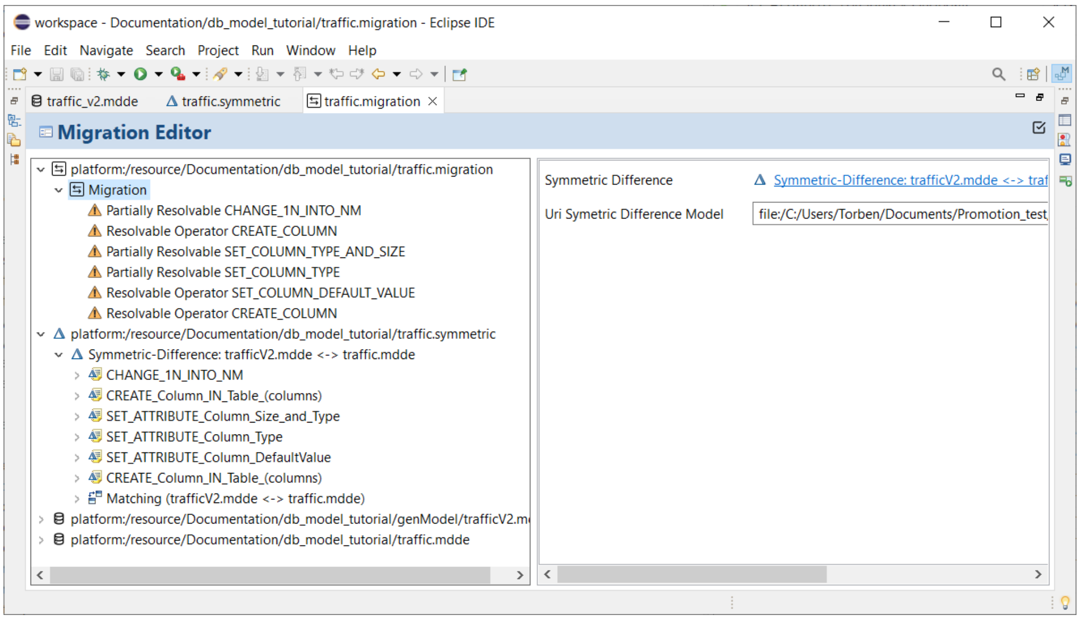


Documentation
db_model
db_model_all
traffic_v2.mdde
traffic.mdde
Mdde
Mdde
Migra
Model
test

Compare: traffic.mdde <-> traffic_v2.mdde
platform:/resource/Documentation/db_model_all/traffic.symmetric
Symmetric-Difference: traffic.mdde <-> traffic_v2.mdde
CHANGE_1N_INTO_NM
CREATE_Column_IN_Table_(columns)
CREATE_Column_IN_Table_(columns)
traffic.mdde <-> traffic_v2.mdde)
e/Documentation/db_model_all/traffic.mdde
e/Documentation/db_model_all/traffic_v2.mdde

traffic.symmetric - Documentation/db_model_all

Figure 26: Create migration model


workspace - Documentation/db_model_tutorial/traffic.migration - Eclipse IDE
File Edit Navigate Search Project Run Window Help
traffic_v2.mdde
traffic.symmetric
traffic.migration
Migration Editor

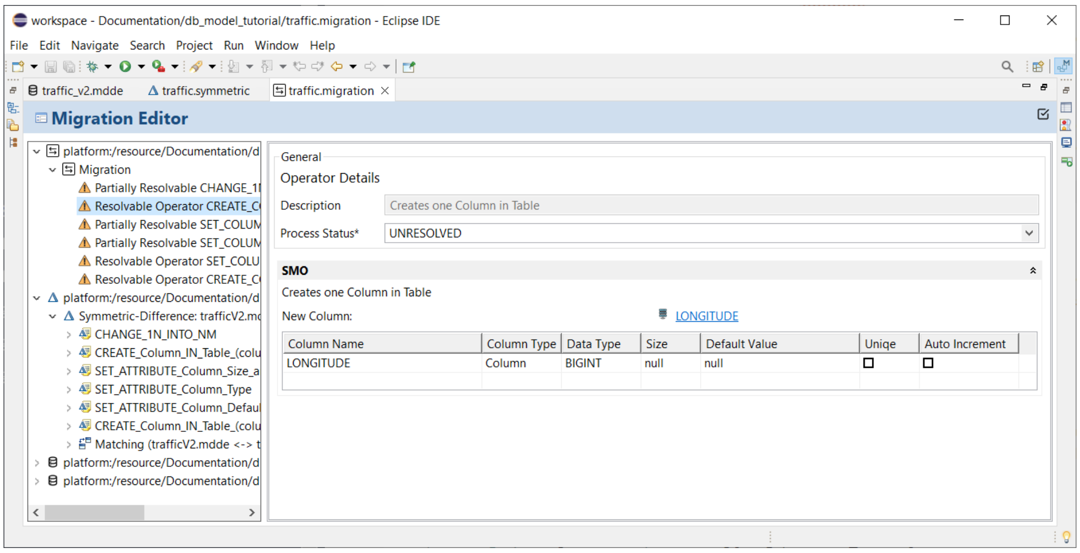
platform:/resource/Documentation/db_model_tutorial/traffic.migration
Migration
Partially Resolvable CHANGE_1N_INTO_NM
Resolvable Operator CREATE_COLUMN
Partially Resolvable SET_COLUMN_TYPE_AND_SIZE
Partially Resolvable SET_COLUMN_TYPE
Resolvable Operator SET_COLUMN_DEFAULT_VALUE
Resolvable Operator CREATE_COLUMN
platform:/resource/Documentation/db_model_tutorial/traffic.symmetric
Symmetric-Difference: trafficV2.mdde <-> traffic.mdde
CHANGE_1N_INTO_NM
CREATE_Column_IN_Table_(columns)
SET_ATTRIBUTE_Column_Size_and_Type
SET_ATTRIBUTE_Column_Type
SET_ATTRIBUTE_Column_DefaultValue
CREATE_Column_IN_Table_(columns)
Matching (trafficV2.mdde <-> traffic.mdde)
platform:/resource/Documentation/db_model_tutorial/genModel/trafficV2.m
platform:/resource/Documentation/db_model_tutorial/traffic.mdde

| | |
|---|---|
| Symmetric Difference | Symmetric-Difference: trafficV2.mdde <-> traf |
| Uri Symetric Difference Model | file:/C:/Users/Torben/Documents/Promotion_test |

Figure 27: Migration model editor

The new model should be opened automatically in the Migration Editor. This editor also consists of two areas. On the left side, a tree-based navigation area can be used to navigate through the model. On the right side, the currently selected element is displayed. The migration model references all models created so far, so four models are listed in the navigation view. These four models are the two versions of the traffic model, the difference model, and the migration model (fig. 27).

During the model transformation, the operators were classified according to the classification from the previous section. Operators marked as *Resolvable* belong to the group of *non-breaking* operators. These operators do not affect already existing data. For example, the *CREATE_COLUMN* operator belongs to this group. Operators marked as *Partially Resolvable* belong to the group of *breaking and resolvable* Operators. Operators of the group *breaking and unresolvable* are prefixed with *Not automatically resolvable*.


workspace - Documentation/db_model_tutorial/traffic.migration - Eclipse IDE

File Edit Navigate Search Project Run Window Help

traffic_v2.mdde | traffic.symmetric | traffic.migration

**Migration Editor**

- platform:/resource/Documentation/d
  - Migration
    - Partially Resolvable CHANGE_1I
    - Resolvable Operator CREATE_C
    - Partially Resolvable SET_COLUM
    - Partially Resolvable SET_COLUM
    - Resolvable Operator SET_COLU
    - Resolvable Operator CREATE_C
- platform:/resource/Documentation/d
  - Symmetric-Difference: trafficV2.mc
    - CHANGE_1N_INTO_NM
    - CREATE_Column_IN_Table_(colu
    - SET_ATTRIBUTE_Column_Size_a
    - SET_ATTRIBUTE_Column_Type
    - SET_ATTRIBUTE_Column_Defaul
    - CREATE_Column_IN_Table_(colu
    - Matching (trafficV2.mdde <-> t
- platform:/resource/Documentation/d
- platform:/resource/Documentation/d

General

Operator Details

Description: Creates one Column in Table

Process Status*: UNRESOLVED

**SMO**

Creates one Column in Table

New Column: LONGITUDE

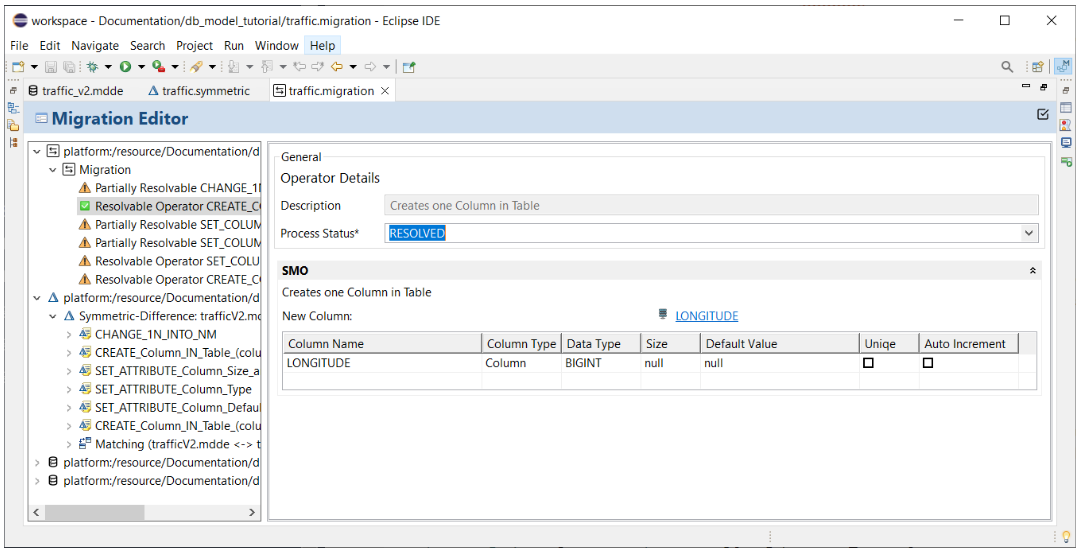
| Column Name | Column Type | Data Type | Size | Default Value | Uniqe | Auto Increment |
|---|---|---|---|---|---|---|
| LONGITUDE | Column | BIGINT | null | null | ☐ | ☐ |

Figure 28: Resolvable operator Create Column

If we select one of the operators, the description opens in the right area of the editor(fig. 28). Every operator should be reviewed by the developer. This precaution may be necessary as not all change operations are covered yet. Therefore, every operator has a so-called process status. We can change the status by selecting a value from the drop-down menu below the operator's description. The three possible values are:

- **UNRESOLVED**: Unresolved is the default value. After creating the migration model, all operators are initially assigned this value. This status means that the operator has not been checked and confirmed by the user yet.
- **RESOLVED**: This status means that the operator has been checked by the user and should be taken into account when subsequently generating the migration scripts.
- **IGNORE**: If an operator is incorrect, it is not necessary to repeat the comparison or the creation of the migration model. Instead, the status can be set to ignore. Operators with this status will not be considered afterwards.

All operators with the status *UNRESOLVED*, are marked with a warning in the tree-based editor. As soon as we choose another status, a green checkmark appears instead of the warning (fig. 29).


workspace - Documentation/db_model_tutorial/traffic.migration - Eclipse IDE
File Edit Navigate Search Project Run Window Help
traffic_v2.mdde
traffic.symmetric
traffic.migration
Migration Editor

platform:/resource/Documentation/d
Migration
Partially Resolvable CHANGE_1I
Resolvable Operator CREATE_C
Partially Resolvable SET_COLUM
Partially Resolvable SET_COLUM
Resolvable Operator SET_COLU
Resolvable Operator CREATE_C
platform:/resource/Documentation/d
Symmetric-Difference: trafficV2.m
CHANGE_1N_INTO_NM
CREATE_Column_IN_Table_(colu
SET_ATTRIBUTE_Column_Size_a
SET_ATTRIBUTE_Column_Type
SET_ATTRIBUTE_Column_Defau
CREATE_Column_IN_Table_(colu
Matching (trafficV2.mdde <-> t
platform:/resource/Documentation/d
platform:/resource/Documentation/d

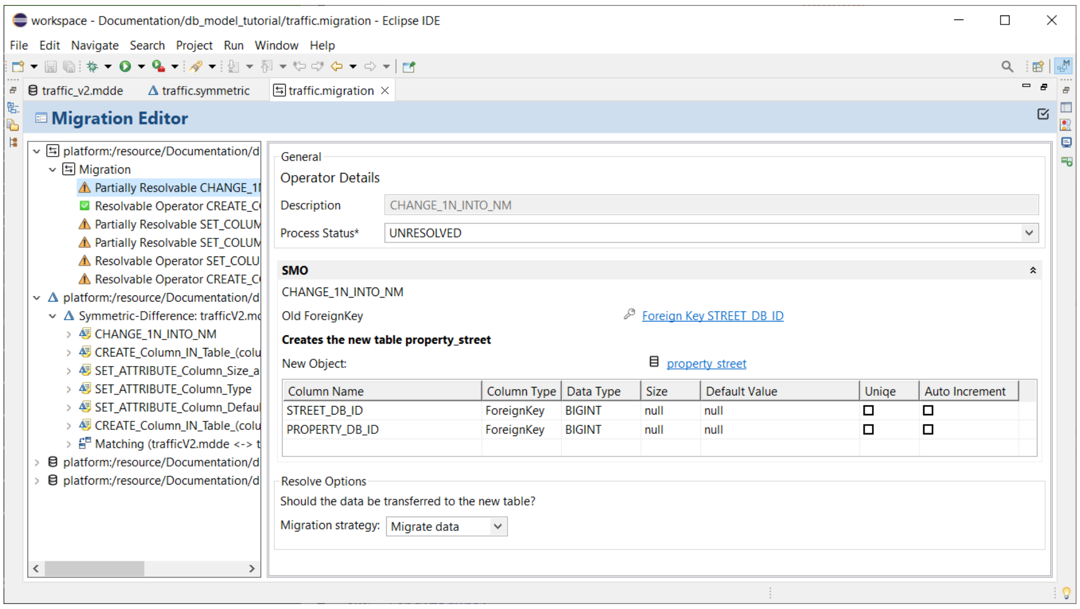
General
Operator Details
Description: Creates one Column in Table
Process Status*: RESOLVED

SMO
Creates one Column in Table
New Column: LONGITUDE

| Column Name | Column Type | Data Type | Size | Default Value | Uniqe | Auto Increment |
|---|---|---|---|---|---|---|
| LONGITUDE | Column | BIGINT | null | null | ☐ | ☐ |

Figure 29: Operator was marked as resolved

### 3.4.2 Breaking & Resolvable


workspace - Documentation/db_model_tutorial/traffic.migration - Eclipse IDE
File Edit Navigate Search Project Run Window Help
traffic_v2.mdde
traffic.symmetric
traffic.migration
Migration Editor

platform:/resource/Documentation/d
Migration
Partially Resolvable CHANGE_1I
Resolvable Operator CREATE_C
Partially Resolvable SET_COLUM
Partially Resolvable SET_COLUM
Resolvable Operator SET_COLU
Resolvable Operator CREATE_C
platform:/resource/Documentation/d
Symmetric-Difference: trafficV2.m
CHANGE_1N_INTO_NM
CREATE_Column_IN_Table_(colu
SET_ATTRIBUTE_Column_Size_a
SET_ATTRIBUTE_Column_Type
SET_ATTRIBUTE_Column_Defau
CREATE_Column_IN_Table_(colu
Matching (trafficV2.mdde <-> t
platform:/resource/Documentation/d
platform:/resource/Documentation/d

General
Operator Details
Description: CHANGE_1N_INTO_NM
Process Status*: UNRESOLVED

SMO
CHANGE_1N_INTO_NM
Old ForeignKey: Foreign Key STREET_DB_ID
**Creates the new table property_street**
New Object: property_street

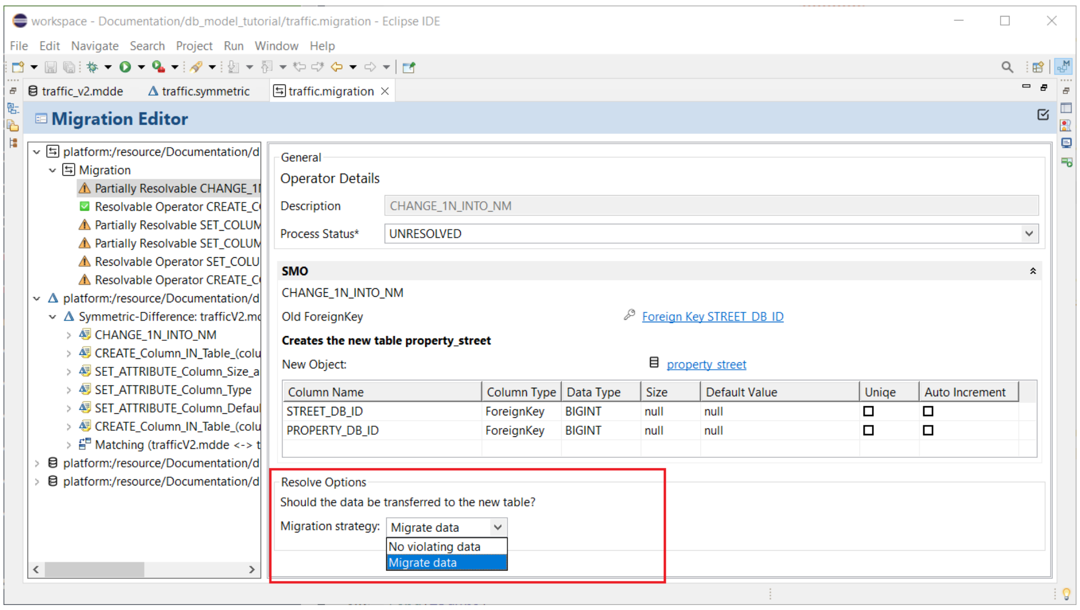
| Column Name | Column Type | Data Type | Size | Default Value | Uniqe | Auto Increment |
|---|---|---|---|---|---|---|
| STREET_DB_ID | ForeignKey | BIGINT | null | null | ☐ | ☐ |
| PROPERTY_DB_ID | ForeignKey | BIGINT | null | null | ☐ | ☐ |

Resolve Options
Should the data be transferred to the new table?
Migration strategy: Migrate data

Figure 30: Partially Resolvable Operator

All operators requiring data migration usually belong to the group *Breaking & Resolvable*. For these operators, it is necessary to specify how and whether EcolveDB should migrate the data. For this purpose, EvolveDB provides different migration strategies. These strategies depend on the change operator. To make this more precise, we will look at three examples below.

**CHANGE_1N_INTO_NM:** Changing the multiplicity between property and street requires several operations (fig. 31). First, we must create a new table with the two foreign keys. Second, existing data should be transferred to the new table. Finally, we can delete the old foreign key. EvolveDB allows us to choose between two different migration strategies. We can choose a migration strategy by selecting a value from the dropdown box (fig. 31). In this case, the possibilities are simple because we only have two options. If *Migrate Data* is selected, the data will be transferred to the new table. Otherwise, the data will not be migrated and could be lost.


workspace - Documentation/db_model_tutorial/traffic.migration - Eclipse IDE
File Edit Navigate Search Project Run Window Help
traffic_v2.mdde
traffic.symmetric
traffic.migration
Migration Editor

platform:/resource/Documentation/d
Migration
Partially Resolvable CHANGE_1I
Resolvable Operator CREATE_C
Partially Resolvable SET_COLUM
Partially Resolvable SET_COLUM
Resolvable Operator SET_COLU
Resolvable Operator CREATE_C
platform:/resource/Documentation/d
Symmetric-Difference: trafficV2.mc
CHANGE_1N_INTO_NM
CREATE_Column_IN_Table_(colu
SET_ATTRIBUTE_Column_Size_a
SET_ATTRIBUTE_Column_Type
SET_ATTRIBUTE_Column_Defaul
CREATE_Column_IN_Table_(colu
Matching (trafficV2.mdde <-> t
platform:/resource/Documentation/d
platform:/resource/Documentation/d

General

Operator Details

Description: CHANGE_1N_INTO_NM

Process Status*: UNRESOLVED

SMO

CHANGE_1N_INTO_NM

Old ForeignKey: Foreign Key STREET_DB_ID

**Creates the new table property_street**

New Object: property_street

| Column Name | Column Type | Data Type | Size | Default Value | Uniqe | Auto Increment |
|---|---|---|---|---|---|---|
| STREET_DB_ID | ForeignKey | BIGINT | null | null | ☐ | ☐ |
| PROPERTY_DB_ID | ForeignKey | BIGINT | null | null | ☐ | ☐ |

Resolve Options

Should the data be transferred to the new table?

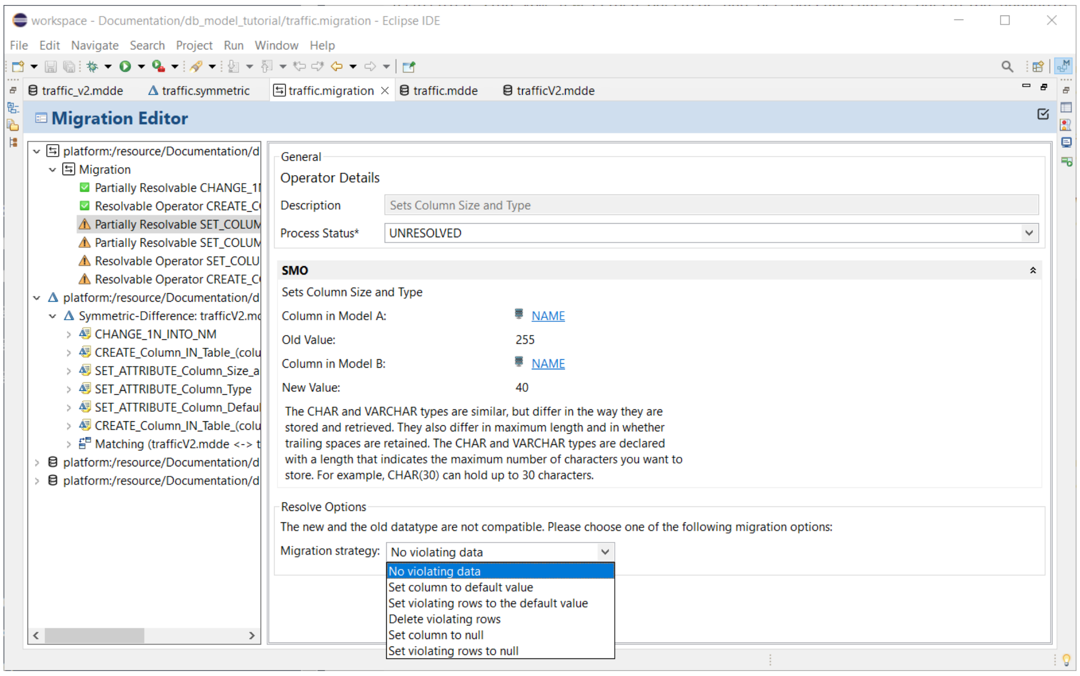
Migration strategy: Migrate data

No violating data
Migrate data

Figure 31: Migration strategy

**SET_COLUMN_SIZE:** In this example, we reduced the size of *VARCHAR* column from 255 to 40 (fig. 32). If the column contains values longer than the new size, the migration will abort with an error. The following enumeration lists all possible migration strategies:

- **No violating data**: no values invalidate the new constraint.
- If the column has a default value and has no unique constraint:
  1. **Set column to default value**: all values are set to the default value.
  2. **Set violating rows to the default value**: all values longer than 40 are set to the default value.
- If the column is nullable:
  1. **Set column to null**: all values are set to null.


workspace - Documentation/db_model_tutorial/traffic.migration - Eclipse IDE

File Edit Navigate Search Project Run Window Help

traffic_v2.mdde | traffic.symmetric | traffic.migration | traffic.mdde | trafficV2.mdde

Migration Editor

- platform:/resource/Documentation/d
  - Migration
    - Partially Resolvable CHANGE_1I
    - Resolvable Operator CREATE_C
    - Partially Resolvable SET_COLUM
    - Partially Resolvable SET_COLUM
    - Resolvable Operator SET_COLU
    - Resolvable Operator CREATE_C
- platform:/resource/Documentation/d
  - Symmetric-Difference: trafficV2.m
    - CHANGE_1N_INTO_NM
    - CREATE_Column_IN_Table_(colu
    - SET_ATTRIBUTE_Column_Size_a
    - SET_ATTRIBUTE_Column_Type
    - SET_ATTRIBUTE_Column_Defau
    - CREATE_Column_IN_Table_(colu
    - Matching (trafficV2.mdde <-> t
- platform:/resource/Documentation/d
- platform:/resource/Documentation/d

General

Operator Details

Description: Sets Column Size and Type

Process Status*: UNRESOLVED

SMO

Sets Column Size and Type

| | |
|---|---|
| Column in Model A: | NAME |
| Old Value: | 255 |
| Column in Model B: | NAME |
| New Value: | 40 |

The CHAR and VARCHAR types are similar, but differ in the way they are stored and retrieved. They also differ in maximum length and in whether trailing spaces are retained. The CHAR and VARCHAR types are declared with a length that indicates the maximum number of characters you want to store. For example, CHAR(30) can hold up to 30 characters.

Resolve Options

The new and the old datatype are not compatible. Please choose one of the following migration options:

Migration strategy: No violating data

- No violating data
- Set column to default value
- Set violating rows to the default value
- Delete violating rows
- Set column to null
- Set violating rows to null

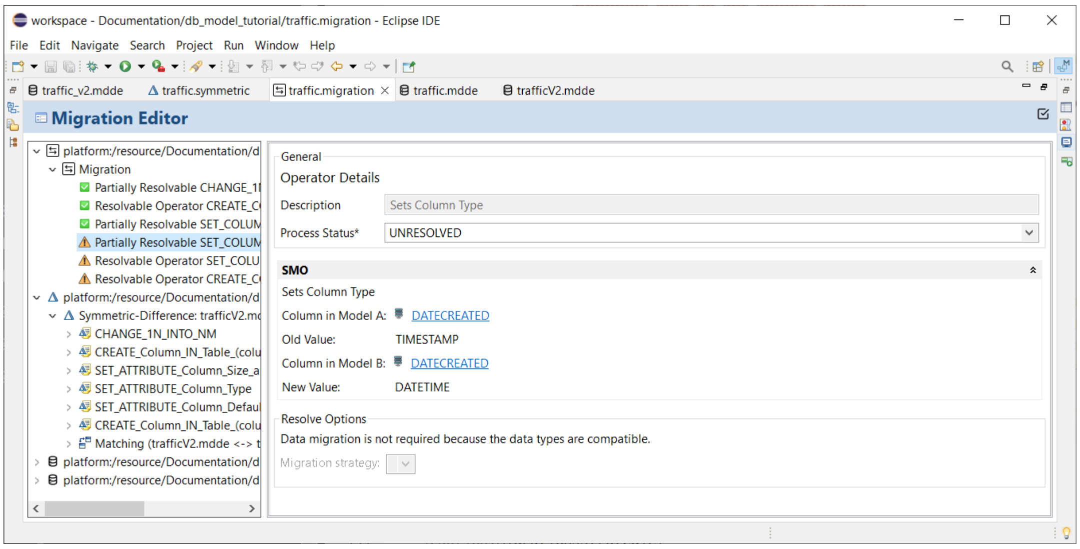
Figure 32: Migration strategy

2. **Set violating rows to null**: all values longer than 40 are set to null.

**SET_COLUMN_TYPE:** In the third example, we consider the type change. Type changes also belong to the second group (*Breaking & Resolvable*) because existing data could be incompatible with the target type. EvolveDB automatically checks if the old and the new types are compatible. Depending on the result, different migration strategies are offered. In our example shown (fig. 33), we changed the column type from *TIMESTAMP* to *DATETIME*. Because both data types are compatible, we do not need to choose a migration strategy in this case.

### 3.4.3 Not Automatically Resolvable

-TBD-


workspace - Documentation/db_model_tutorial/traffic.migration - Eclipse IDE

File Edit Navigate Search Project Run Window Help

traffic_v2.mdde | traffic.symmetric | traffic.migration | traffic.mdde | trafficV2.mdde

Migration Editor

- platform:/resource/Documentation/d
  - Migration
    - Partially Resolvable CHANGE_1I
    - Resolvable Operator CREATE_C
    - Partially Resolvable SET_COLUM
    - Partially Resolvable SET_COLUM
    - Resolvable Operator SET_COLU
    - Resolvable Operator CREATE_C
- platform:/resource/Documentation/d
  - Symmetric-Difference: trafficV2.mc
    - CHANGE_1N_INTO_NM
    - CREATE_Column_IN_Table_(colu
    - SET_ATTRIBUTE_Column_Size_a
    - SET_ATTRIBUTE_Column_Type
    - SET_ATTRIBUTE_Column_Defaul
    - CREATE_Column_IN_Table_(colu
    - Matching (trafficV2.mdde <-> t
- platform:/resource/Documentation/d
- platform:/resource/Documentation/d

General

Operator Details

Description: Sets Column Type

Process Status*: UNRESOLVED

SMO

Sets Column Type

Column in Model A: DATECREATED

Old Value: TIMESTAMP

Column in Model B: DATECREATED

New Value: DATETIME

Resolve Options

Data migration is not required because the data types are compatible.

Migration strategy:

Figure 33: Migration strategy

## 3.5 Generate Migration Scripts

After choosing a migration strategy and setting the process status, we continue generating the migration script. Select the migration model in Package Explorer and open the context menu with a right-click. Choose **MDSE ⇒ Generate SQL Migrations** (fig. 34). We can also start the generation via the toolbar in the Migration Editor.

A wizard dialog opens. We have to choose the generator on the first page of the dialog. Since the SQL commands can differ slightly depending on the underlying database or the database version, EvolveDB offers an extension point for generators. In the default installation, only the MySQL-compatible generator is included (fig. 35).

On the second page of the dialog, we must select the location for the SQL script (fig. 36). Currently, we can only select a project in the workspace as the storage location. With *Finish* the generation process starts. If the file is not visible immediately afterwards, the project must be *refreshed*.

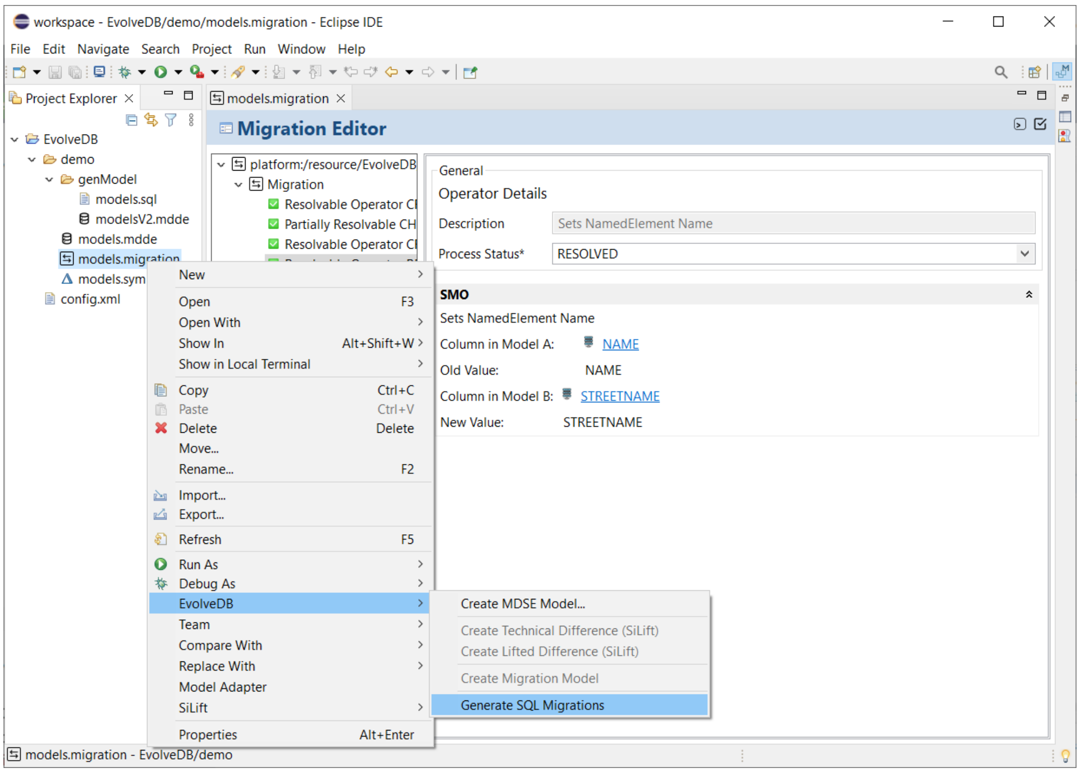


Figure 34: Create SQL migration script

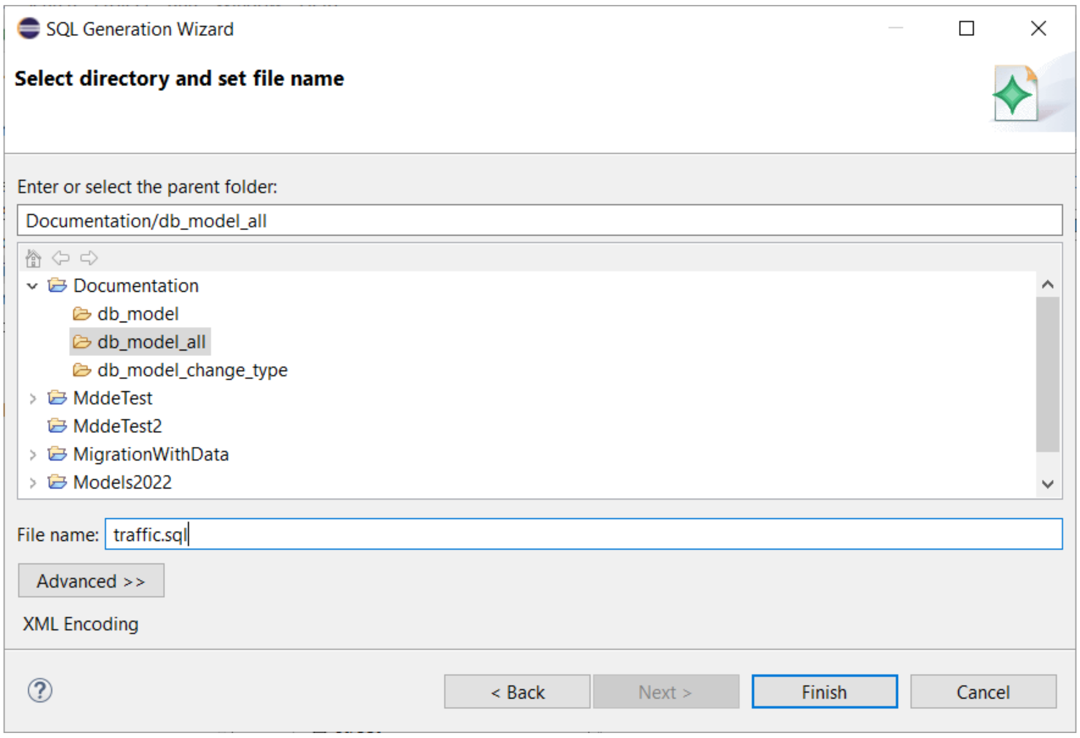
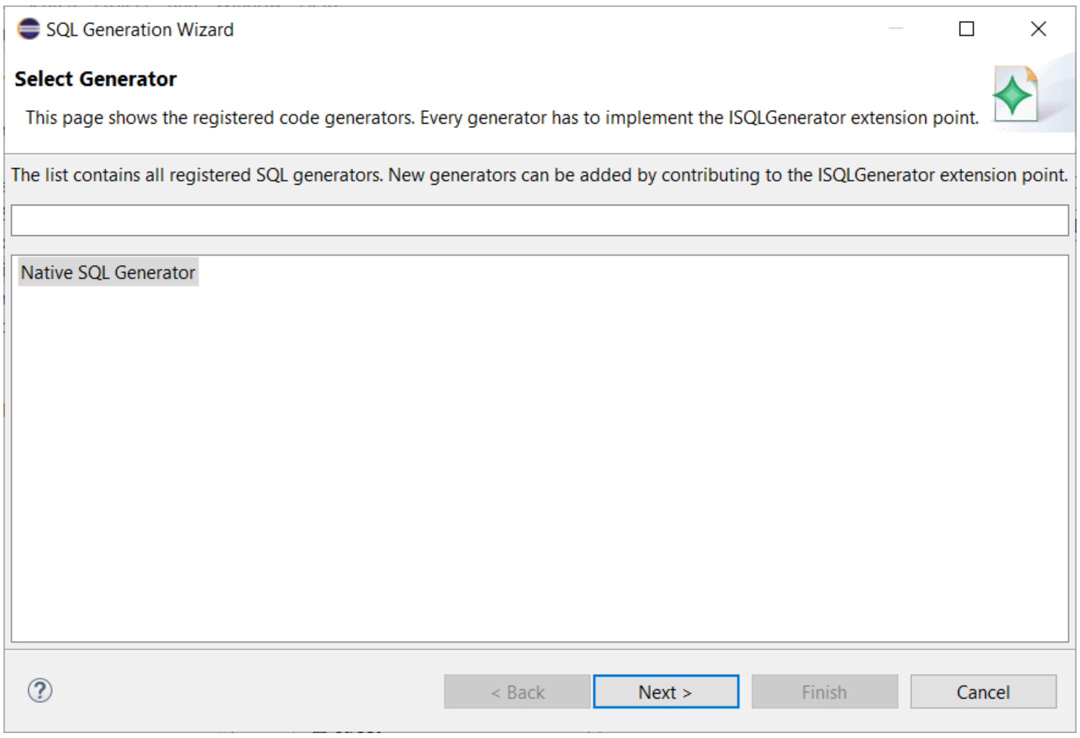


Figure 36: Choose file location



**Select Generator**

This page shows the registered code generators. Every generator has to implement the ISQLGenerator extension point.

The list contains all registered SQL generators. New generators can be added by contributing to the ISQLGenerator extension point.

Native SQL Generator

Figure 35: Select generator

# 4 Migration Scripts

In this chapter we will have a look at the generated migration script. Listing 1 shows the generated script when we use the model from our running example (fig. 27).

```
USE marburg_2020_models3;
START TRANSACTION;

-- Creates an history table for deleted and updated values
CREATE TABLE IF NOT EXISTS `marburg_2020_models3`.`
    mdde_history` (
  `DB_ID` BIGINT NOT NULL AUTO_INCREMENT,
  `TABLENAME` VARCHAR(255) NOT NULL,
  `COLUMN_DB_ID` BIGINT NOT NULL,
  `COLUMN_NAME` VARCHAR(255) NOT NULL,
  `VALUE` BLOB NULL,
  `CHANGEDATE` DATETIME NOT NULL,
  PRIMARY KEY (`DB_ID`));

-- Add the new column LONGITUDE in Table crossroad
ALTER TABLE `crossroad`
ADD COLUMN `LONGITUDE` BIGINT;
```

```

-- Change default value of lanes
ALTER TABLE ‘crossroad‘ CHANGE COLUMN ‘LANES‘ ‘LANES‘ BIGINT
    NULL DEFAULT 1 ;

-- Add the new column LATITUDE in Table crossroad
ALTER TABLE ‘crossroad‘
ADD COLUMN ‘LATITUDE‘ BIGINT;

-- Change column type of datecreated
ALTER TABLE ‘street‘ CHANGE COLUMN ‘DATECREATED‘ ‘
   DATECREATED‘ DATETIME(0) NULL  ;

-- Create Table property_street
CREATE TABLE IF NOT EXISTS property_street  (
‘STREET_DB_ID‘ BIGINT NOT NULL,
‘PROPERTY_DB_ID‘ BIGINT NOT NULL
,PRIMARY KEY(‘STREET_DB_ID‘ ,‘PROPERTY_DB_ID‘ ),
CONSTRAINT ‘property_crossroad_ibfk2‘
FOREIGN KEY (‘STREET_DB_ID‘)
REFERENCES ‘street‘(‘DB_ID‘)
ON DELETE RESTRICT
ON UPDATE RESTRICT,
CONSTRAINT ‘property_crossroad_ibfk3‘
FOREIGN KEY (‘PROPERTY_DB_ID‘)
REFERENCES ‘property‘(‘DB_ID‘)
ON DELETE RESTRICT
ON UPDATE RESTRICT
);

BEGIN;
SET @safe_mode = @@SQL_SAFE_UPDATES;
SET SQL_SAFE_UPDATES = 0;

-- Migrate data to the new table
INSERT INTO ‘property_street‘ (PROPERTY_DB_ID , STREET_DB_ID
   )
SELECT DB_ID, STREET_DB_ID FROM property WHERE STREET_DB_ID
   IS NOT NULL;

SET SQL_SAFE_UPDATES = @safe_mode;
COMMIT;
-- If executing the script fails, we suggest a rollback.

-- Drop foreign key in property
ALTER TABLE `property` DROP FOREIGN KEY `ibfk_street`;
ALTER TABLE `property` DROP COLUMN `STREET_DB_ID`;

-- Find violating rows
SET @sql_mode = @@SESSION.sql_mode;
set @@SESSION.sql_mode = '';
DROP TEMPORARY TABLE IF EXISTS my_temp_id_table;
CREATE TEMPORARY TABLE my_temp_id_table
   SELECT DB_ID from street v where LENGTH(`NAME`) > 40;
set @@SESSION.sql_mode = @sql_mode;

BEGIN;
SET @safe_mode = @@SQL_SAFE_UPDATES;
SET SQL_SAFE_UPDATES = 0;

-- Insert violating values into the history table
INSERT INTO `marburg_2020_models3`.`mdde_history`
(`TABLENAME`,
`COLUMN_DB_ID`,
`COLUMN_NAME`,
`VALUE`,
`CHANGEDATE`)
SELECT 'street', DB_ID, 'NAME', NAME, now() from street s
   where LENGTH(`NAME`) > 40;

-- Set violating rows to the default value
UPDATE `street` SET `NAME` = 'unnamed␣road' where DB_ID in (
   Select DB_ID from  my_temp_id_table);

SET SQL_SAFE_UPDATES = @safe_mode;
COMMIT;
-- If executing the script fails, we suggest a rollback.

DROP TEMPORARY TABLE IF EXISTS my_temp_id_table;

-- Change column size of name
ALTER TABLE `street` CHANGE COLUMN `NAME` `NAME` VARCHAR(40)
    NULL DEFAULT 'unnamed␣road';
COMMIT;
-- If executing the script fails, we suggest a rollback.
```

Listing 1: Generated SQL code

If we apply the script to the original schema in figure 3, we obtain the migrated schema shown in figure 37. The new columns *LATITUDE* and *LONGITUDE* have been added. The original foreign key relationship between the *property* and *street* tables was moved to the new cross-reference table *property_street*. Furthermore, the type of the *DATECREATED* column was changed to *datetime* and the size of the *NAME* column was reduced to 40.

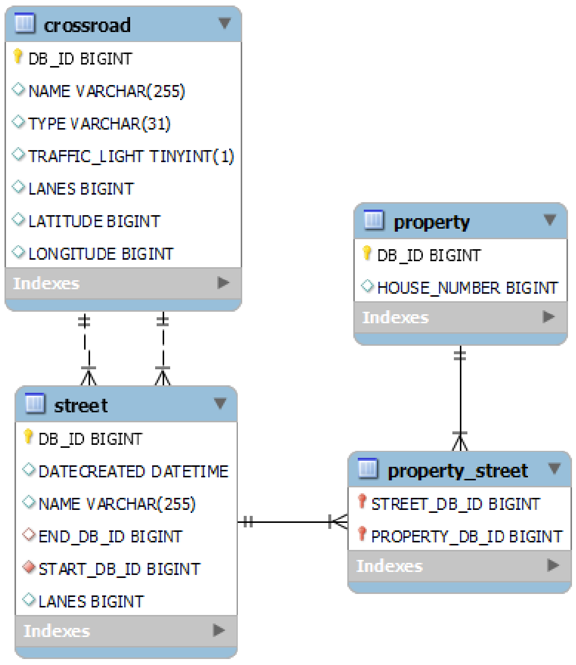
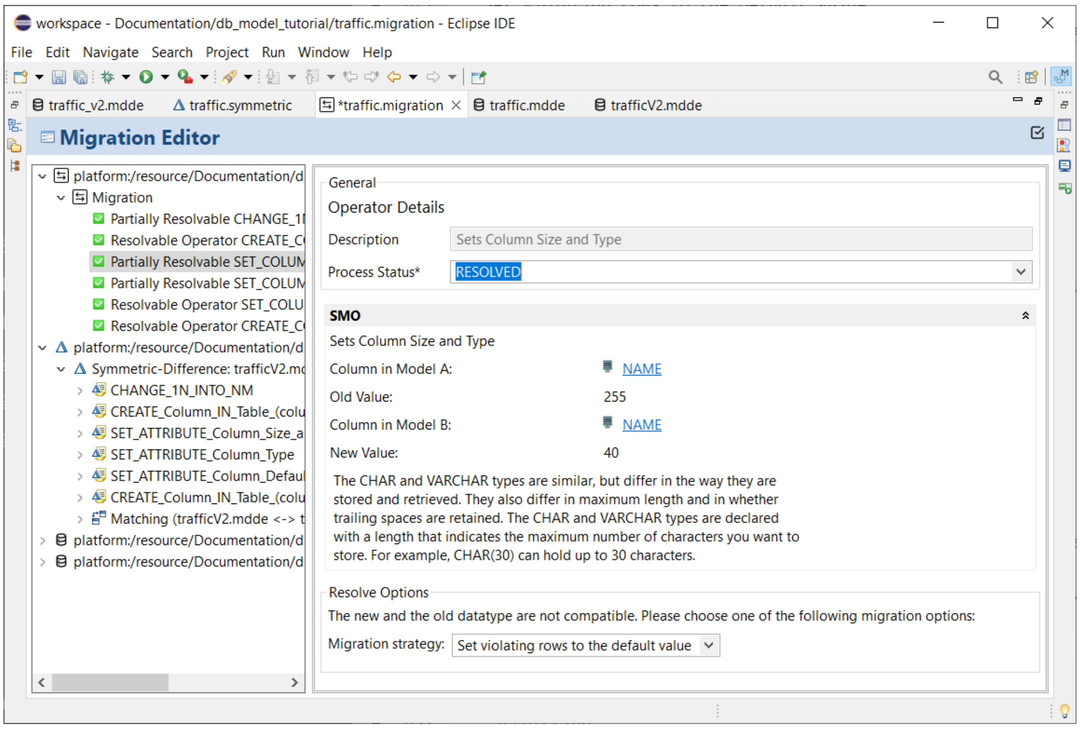


Figure 37: Das migrierte Datenbankschema


workspace - Documentation/db_model_tutorial/traffic.migration - Eclipse IDE

File Edit Navigate Search Project Run Window Help

traffic_v2.mdde | traffic.symmetric | *traffic.migration | traffic.mdde | trafficV2.mdde

**Migration Editor**

- platform:/resource/Documentation/d
  - Migration
    - Partially Resolvable CHANGE_1I
    - Resolvable Operator CREATE_C
    - Partially Resolvable SET_COLUM
    - Partially Resolvable SET_COLUM
    - Resolvable Operator SET_COLU
    - Resolvable Operator CREATE_C
- platform:/resource/Documentation/d
  - Symmetric-Difference: trafficV2.mc
    - CHANGE_1N_INTO_NM
    - CREATE_Column_IN_Table_(colu
    - SET_ATTRIBUTE_Column_Size_a
    - SET_ATTRIBUTE_Column_Type
    - SET_ATTRIBUTE_Column_Defaul
    - CREATE_Column_IN_Table_(colu
    - Matching (trafficV2.mdde <-> t
- platform:/resource/Documentation/d
- platform:/resource/Documentation/d

General

Operator Details

Description: Sets Column Size and Type

Process Status*: RESOLVED

SMO

Sets Column Size and Type

Column in Model A: NAME

Old Value: 255

Column in Model B: NAME

New Value: 40

The CHAR and VARCHAR types are similar, but differ in the way they are stored and retrieved. They also differ in maximum length and in whether trailing spaces are retained. The CHAR and VARCHAR types are declared with a length that indicates the maximum number of characters you want to store. For example, CHAR(30) can hold up to 30 characters.

Resolve Options

The new and the old datatype are not compatible. Please choose one of the following migration options:

Migration strategy: Set violating rows to the default value

Figure 38: Migrationsstrategie mit History Tabelle

## 4.1 History Table

As mentioned earlier, EvolveDB provides default migration options. For example, we migrated the data from the old foreign key into the new table. In our running example, we also considered a case 3.4.2 where we had to select a migration strategy. To avoid data loss, EvolveDB creates a history table. Any values that are overwritten by one of the migration options are inserted into the history table. In our example, we chose the option to set all violating values to the default value ’unnamed road’ (fig. 38).

```

-- Creates an history table for deleted and updated values
CREATE TABLE IF NOT EXISTS ‘marburg_2020_models3‘.‘
   mdde_history‘ (
  ‘DB_ID‘ BIGINT NOT NULL AUTO_INCREMENT,
  ‘TABLENAME‘ VARCHAR(255) NOT NULL,
  ‘COLUMN_DB_ID‘ BIGINT NOT NULL,
  ‘COLUMN_NAME‘ VARCHAR(255) NOT NULL,
  ‘VALUE‘ BLOB NULL,
  ‘CHANGEDATE‘ DATETIME NOT NULL,
  PRIMARY KEY (‘DB_ID‘));

-- Create temporary table
SET @sql_mode = @@SESSION.sql_mode;
```

```
14 set @@SESSION.sql_mode = '';
15 DROP TEMPORARY TABLE IF EXISTS my_temp_id_table;
16 CREATE TEMPORARY TABLE my_temp_id_table
17    SELECT DB_ID from street v where LENGTH(`NAME`) > 40;
18 set @@SESSION.sql_mode = @sql_mode;
19
20 -- Find violating rows
21 BEGIN;
22 SET @safe_mode = @@SQL_SAFE_UPDATES;
23 SET SQL_SAFE_UPDATES = 0;
24 INSERT INTO `marburg_2020_models3`.`mdde_history`
25 (`TABLENAME`,
26 `COLUMN_DB_ID`,
27 `COLUMN_NAME`,
28 `VALUE`,
29 `CHANGEDATE`)
30 SELECT 'street', DB_ID, 'NAME', NAME, now() from street s
      where LENGTH(`NAME`) > 40;
31
32 -- Set violating rows to the default value
33 UPDATE `street` SET `NAME` = 'unnamed␣road' where DB_ID in (
      Select DB_ID from my_temp_id_table);
34 SET SQL_SAFE_UPDATES = @safe_mode;
35 COMMIT;
36
37 -- If executing the script fails, we suggest a rollback.
38 DROP TEMPORARY TABLE IF EXISTS my_temp_id_table;
39
40 -- Change column type and size of name
41 ALTER TABLE `street` CHANGE COLUMN `NAME` `NAME` VARCHAR(40)
       NULL DEFAULT 'unnamed␣road';
```

Listing 2: Generated SQL code

The history table (see Fig. 39) is created between line three and line twelve. Subsequently, all columns whose contents are longer than 40 are transferred to the history table. Table 2 shows the history table after we performed the migration. One value was longer than 40 and was replaced by the default value *unnamed road*. Table 3 shows the content of table *street* after migration.

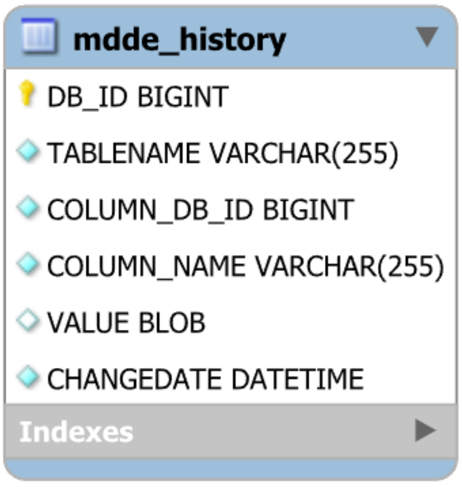


Figure 39: History table

| DB_ID | TABLENAME | COL..._ID | COL..._NAME | VALUE | CHANGEDATE |
|---|---|---|---|---|---|
| 1 | street | 2 | NAME | BLOB | 2022-05-20 15:41:57 |

Table 2: History table after executing the migration script.

| DB_ID | DATECREATED | NAME | END_DB_ID | START_DB_ID | LANES |
|---|---|---|---|---|---|
| 1 | 1976-05-20 15:41:57 | Bahnhofstrasse | 2 | 1 | 2 |
| 2 | 2019-08-12 13:25:57 | unnamed road | 2 | 3 | 3 |
| 3 | 1999-10-02 17:05:41 | Seltersweg | 3 | 1 | 1 |

Table 3: Street table after executing the migration script.