|  |
| --- |
| Evaluant |
| R&D Project |
| **Database Reverse Engineering (for Euss use)** |

|  |
| --- |
|  |

Sommaire

[1. Objective 3](#_Toc194416498)

[2. Description 3](#_Toc194416499)

[3. Resources 3](#_Toc194416500)

[3.1. Euss 3](#_Toc194416501)

[4. Getting database schema 3](#_Toc194416502)

[4.1. ADO.Net 2.0 3](#_Toc194416503)

[4.2. Getting connection string 4](#_Toc194416504)

[4.3. Selecting desired tables 4](#_Toc194416505)

[4.4. Database schema model 5](#_Toc194416506)

[5. Generating domain model 5](#_Toc194416507)

[5.1. Domain model 5](#_Toc194416508)

[5.2. Generation method 5](#_Toc194416509)

[6. Generating mapping file 5](#_Toc194416510)

[6.1. Mapping model 5](#_Toc194416511)

[6.2. Generation method 6](#_Toc194416512)

[7. Project structure 6](#_Toc194416513)

[8. Integration with the mapping designer 6](#_Toc194416514)

# Objective

The aim of this document is to present the project and describe its purpose for Euss (Evaluant Universal Storage Service). Moreover, as it can feed the Euss mapping designer project, we will define the interface between both projects.

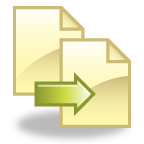
# Description

Reverse engineer a database means get the schema of an existing database. The schema describes tables with their fields, primary keys, relationships … Then, this schema can be analyzed in a specific purpose (i.e. code generation).

In our case, the database schema will be analyzed to be use by Euss.

It will generate two elements:

* The corresponding domain model (source code)
* The corresponding default mapping file



Domain model

Sql Server,  
Oracle,  
MsAccess,  
 …





Mapping file

Db schema

# Resources

## Euss

* Documentation: <http://euss.evaluant.com/>
* Code source : svn://langlay/euss

# Getting database schema

## ADO.Net 2.0

ADO.Net 2.0 provides a method GetSchema() in DbConnection class. This method grabs metadata associated with various metadata types. For each type of metadata, you can grab and filter more precise information.

This method can differ according to the Ado.net provider.

The DataReader class provides the GetSchemaTable() that return metadata corresponding to one table. It gives more precise information about columns (identity, expression, …)

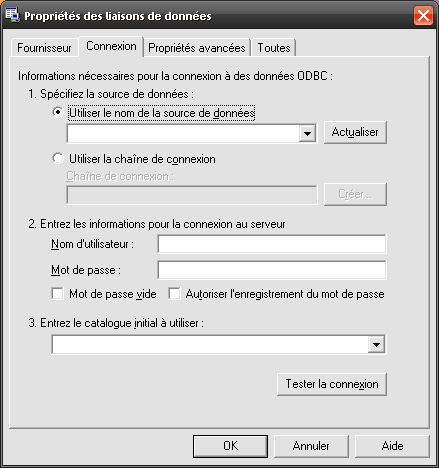
Needed metadata:

* Table names
* Column names and types
* Primary keys
* Relationships

## Getting connection string

Before reading database schema, we must have the database connection string. This last one can be asked to the user through the Data Link Property Editor.

It can be incorporated by referencing the COM Component “*Microsoft OLEDB Service Component 1.0 Type Library”*.

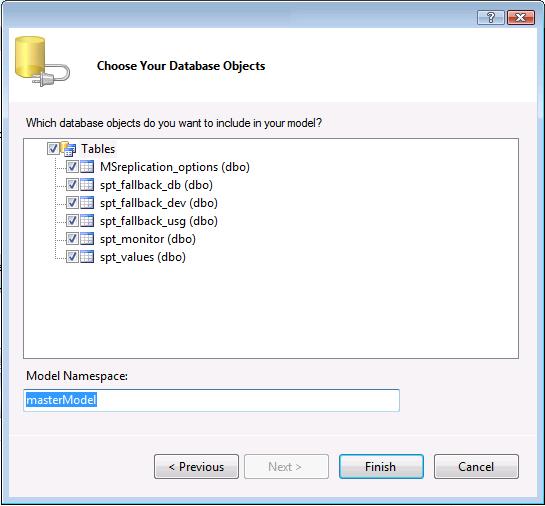


Note:

There is an example here: <http://www.developer.com/db/article.php/2241031>

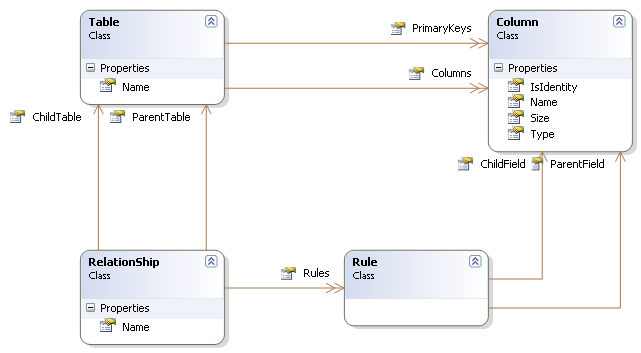
## Selecting desired tables

With the connection, we can list tables of the database and ask the user to choose those that will be managed by Euss.



## Database schema model

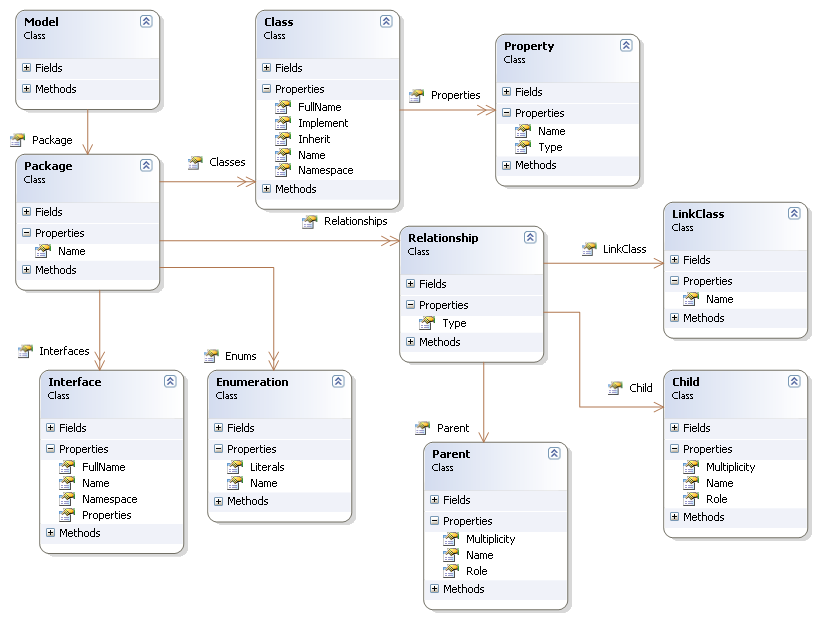
Once we are able to grab all needed metadata, we can build our database schema model that will be used in next step to generate the corresponding domain model and mapping.



# Generating domain model

## Domain model

The domain model is the one already defined in Euss (Evaluant.Uss.DomainModel namespace).



## Generation method

The domain model will be deduced from database schema by applying following mapping logic:

|  |  |
| --- | --- |
| Database model | Domain model |
| Table | Class |
| Column | Property (data type mapping) |
| Relationship | Reference |
| Single Column PK | Property Id |
| Relationship to index table (only 2 FK) | Reference many to many |
|  |  |

Once the domain model created, we can generate the corresponding source code by using CodeDom. We can then target a specific .Net language.

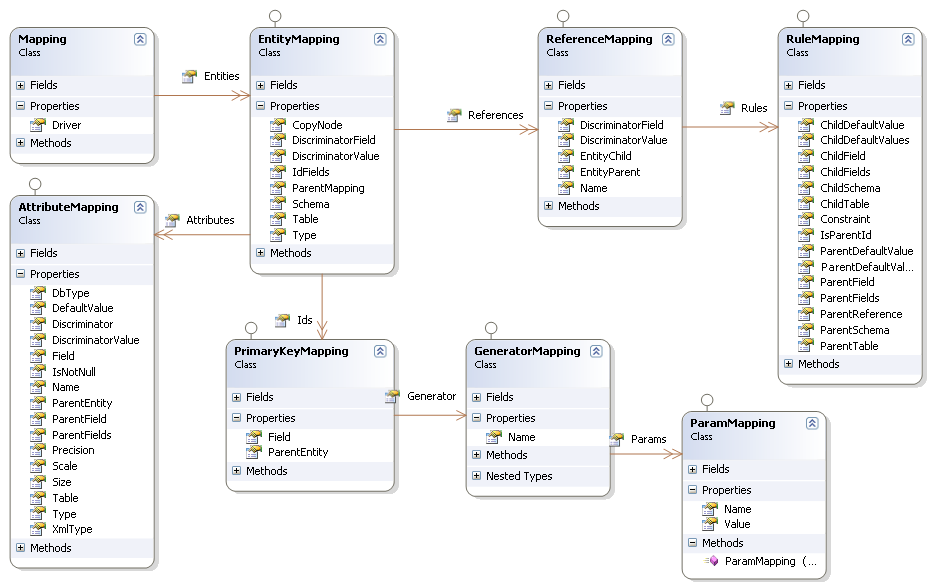
Note:

Inverse references can be generated too (in read only) for an easier OPath manipulation.

# Generating mapping file

## Mapping model

The domain model is the one already defined in Euss (Evaluant.Uss.SqlMapper namespace).



## Generation method

Once we get the database model and the domain model, we must define the mapping file that maps elements of each model.

Here are listed some specific elements:

|  |  |
| --- | --- |
| Database model | Mapping model |
| Multiple Column PK | Business Id |
|  |  |

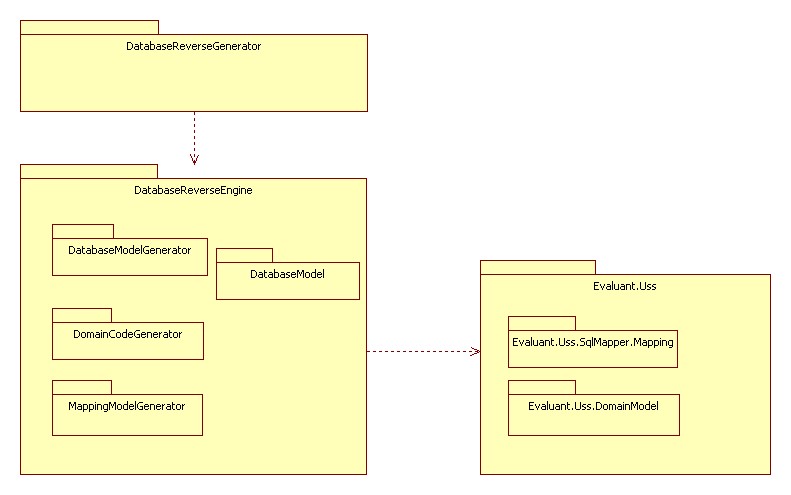
Note:

Inverse relationships can be generated too for an easier OPath manipulation.

# Project structure

This project can be divided in different parts:

* DatabaseReverseGenerator: Application project that contains windows forms.
* DatabaseReverseEngine: Library project that contains generation logics.
  + Database model
  + Domain model
  + Mapping model



# Integration with the mapping designer

## Interface

The interface between the database reverse engine and the mapping designer will be define with the two following models:

* Domain model (required)
* Database model (optional)

If the database model is provide to the designer, it means that a default mapping file should already be present. The designer will also be able to validate some mapping fields.