erwin Data Modeler

Feature Tour

Release 2021 R1

Legal Notices

Contact erwin

Understanding your Support

Review support maintenance programs and offerings.

Registering for Support

Access the erwin support site and click **Sign in** or **Sign up** to register for product support.

Accessing Technical Support

For your convenience, erwin provides easy access to "One Stop" support for all editions of erwin Data Modeler, and includes the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- erwin Support policies and guidelines
- Other helpful resources appropriate for your product

For information about other erwin products, visit http://erwin.com/products.

Provide Feedback

If you have comments or questions, or feedback about erwin product documentation, you can send a message to techpubs@erwin.com.

erwin Data Modeler News and Events

Visit www.erwin.com to get up-to-date news, announcements, and events. View video demos and read up on customer success stories and articles by industry experts.

Contents

Introduction

The Feature Tour guide walks Data Architects, Data Administrators, Application Administrators, Database Administrators, and Partners through the features introduced in erwin Data Modeler (DM) 2021 R1 release.

The features and enhancements introduced in this release are:

- erwin Mart Administrator UI Facelift
- NoSQL Modeling
- JSON and AVRO Support
- Oracle 12c R2, 18c,19c, and 21c
- Microsoft SQL Server 2019
- Microsoft Azure SQL
- MySQL
- Data Vault 2.0 Support
- Productivity and UI Enhancements
- JDBC Support

For additional information about a feature, in erwin Data Modeler, click **Help** > **Help Topics** on the toolbar or press **F1**.

erwin Mart Administrator UI Facelift

erwin Mart Administrator now comes with a brand new UI that follows Google's Material Design principles. The redesigned UI offers an improved user experience with its modern look and feel, dark and light modes, and graphical buttons and icons.

Apart from the overall facelift, the wiki-like editable Home page lets you add information, such as key text, process diagrams, important hyperlinks, resources, and much more. Also, the configurable Dashboard let's you add and view a pictorial presentation of your data and actions on the Mart. You can add charts for your data footprint, model overview and history, profile data, and session overview.

For more information, refer to the erwin Mart Online Help.

NoSQL Modeling

Along with relational databases, erwin Data Modeler (DM) now supports the following NoSQL, non-relational databases as target databases:

- MongoDB 4.x
- Cassandra 3.x
- Couchbase 6.x

These NoSQL databases support all the erwin DM features and functions. The following sections will take you through these features with MongoDB database as an example:

- Migrating a relational model to NoSQL model
- Reverse engineering models from database and script
- Forward engineering models to database
- Comparing changes using Complete Compare

MongoDB Support

erwin Data Modeler (DM) now supports MongoDB 4.x as a target database. This implementation supports the following objects:

- Databases
- Collection
  - Collation
- Index
- Relationships
- User IDs
  - Roles
- View

The following table lists the supported data types:

Numeric	String Literals	Date and Time	Other
• double • binary • int • integer • boolean • minKey • maxKey • long • decimal	• string	• date • timestamp	• object • array • null • objectId • regex • code

Cassandra Support

erwin Data Modeler (DM) now supports Cassandra 3.x/4.x as a target database. This implementation supports the following objects:

- Aggregate
- Function
- Keyspace
- Materialized View
  - Materialized View Column
- Role
- Table
  - Table Column
  - Index
- User Type

The following table lists the supported data types:

Category	Data Type	Supported Constants
Native	• ascii • bigint • blob • boolean • counter • date • decimal • double • float	• string • integer • blob • boolean • integer • integer, string • integer, float • integer, float • integer, float

	• inet • int • smallint • text • time • timestamp • timeuuid • tinyint • uuid • varint	• string • integer • integer • string • integer, string • integer, string • uuid • integer • uuid • integer
Collection	• list • map • set	
Tuple	• tuple	

Couchbase Support

erwin Data Modeler (DM) now supports Couchbase 6.x as a target database. This implementation supports the following objects:

- Bucket
- Document
  - Field
- Full Text Index
- Global Index
- User ID
- View

Following are the supported data types:

- MISSING
- NULL
- BOOLEAN
- NUMBER
- STRING
- ARRAY
- OBJECT
- BINARY

Migrating Relational Models to NoSQL Models

You can convert and migrate your relational models to NoSQL models in two ways:

- Changing the target database
- Deriving a model

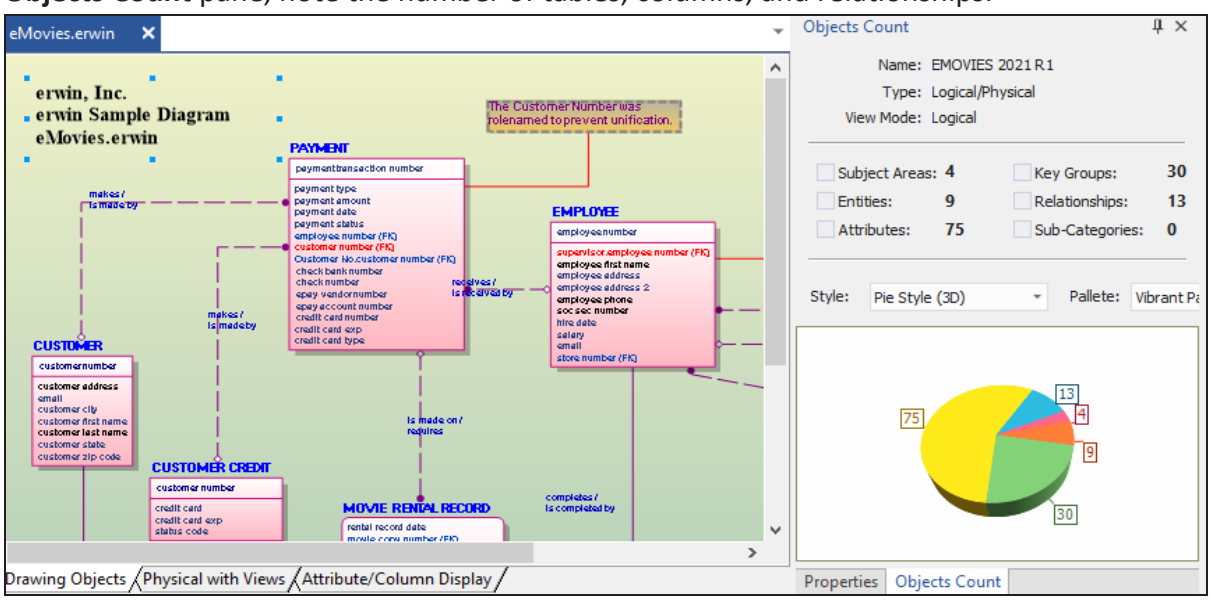
This topic walks you through the steps to migrate a SQL Server model to a MongoDB model. Similarly, you can migrate your relational models to Cassandra and Couchbase models.

Note: Ensure that you keep a backup of your original models.

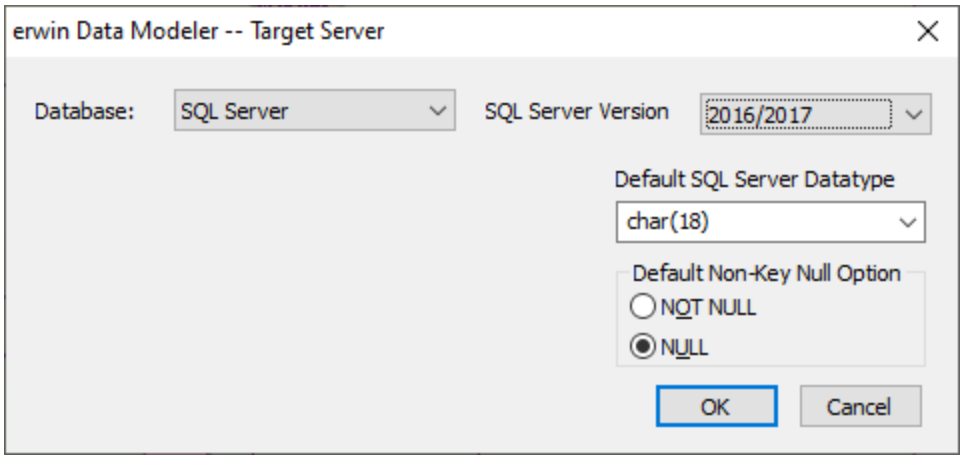
Migration by Changing the Target Database

To migrate by changing the target database, follow these steps:

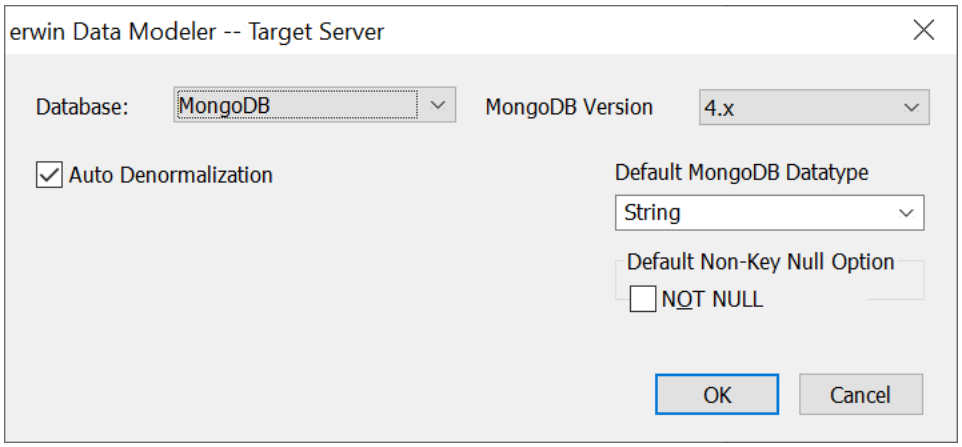
1. Open your relational model in erwin Data Modeler (DM).
   Note: Ensure that you are in the Physical mode.
   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.



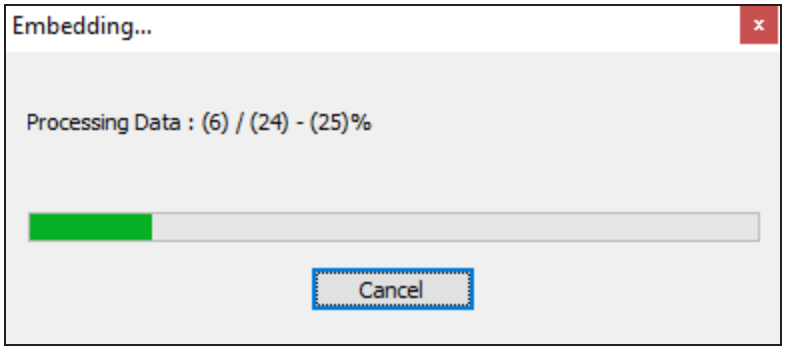
2. On the ribbon, click **Actions** > **Target Database** or on the status bar, click the database name.
   The erwin Data Modeler -- Target Server screen appears.



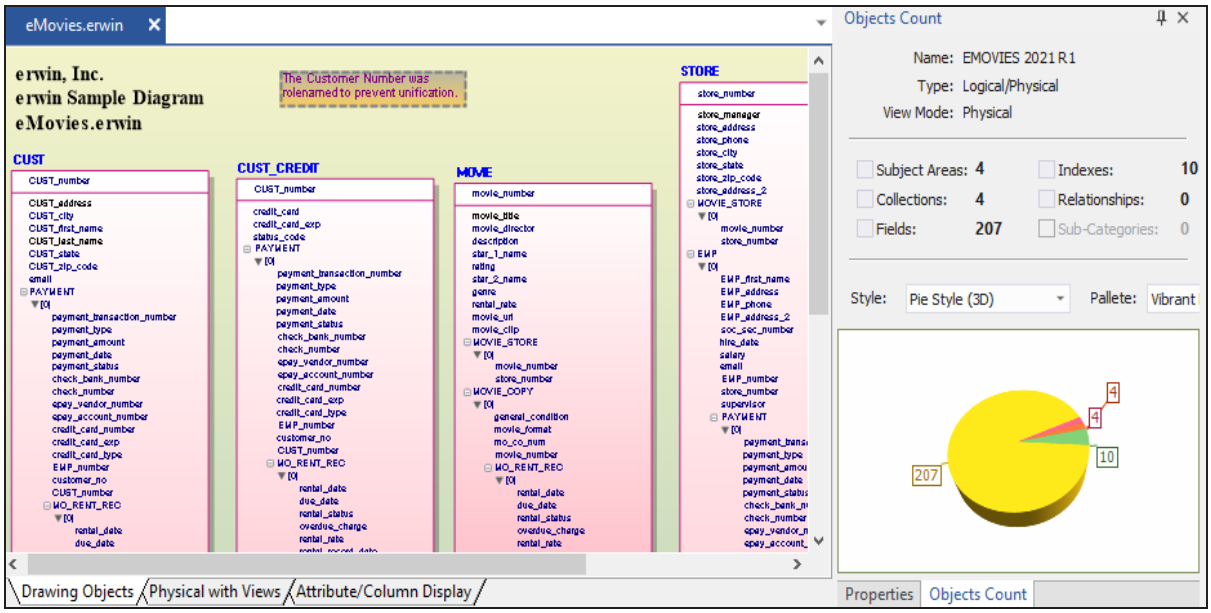
3. In the **Database** drop-down list, select MongoDB.
   By default, the Auto Denormalization check box is selected. Keep it selected.



4. Click **OK**.
   The conversion process starts.



Once the conversion is complete, the existing model in migrated to a NoSQL database.



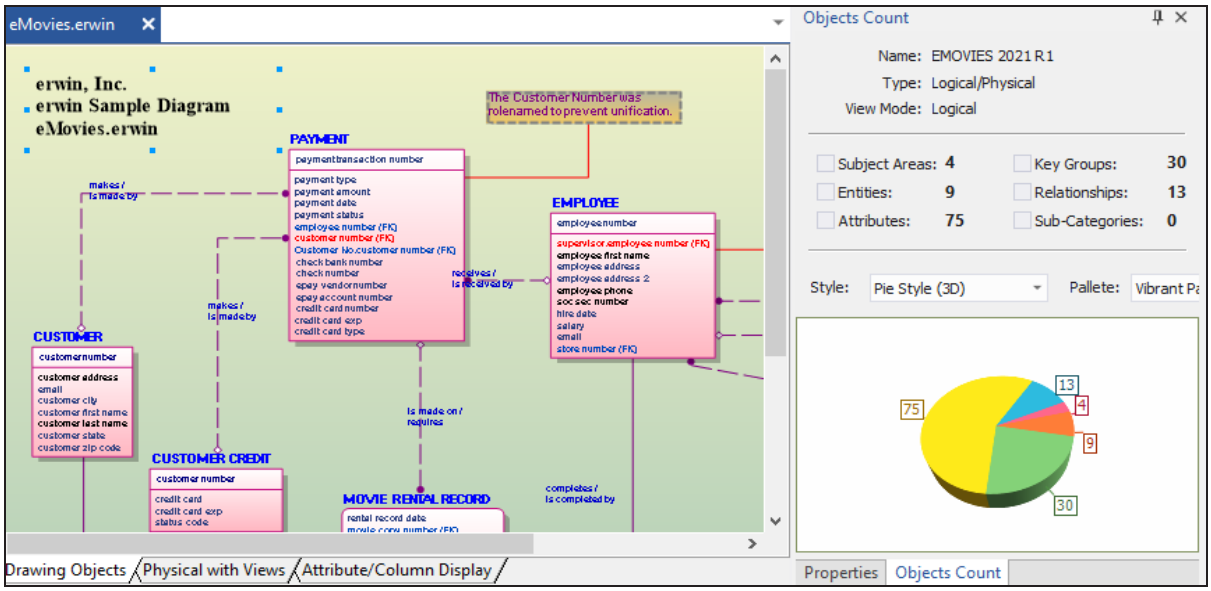
In the **Objects Count** pane, note that instead of tables and columns, we now have collections and fields. Also, the Relationships count has changed to 0. The migration process converts and merges multiple tables, columns, and relationships to the NoSQL format according to the database that you select.

Note: This migration method overwrites the existing model once you save it. Hence, we recommend that you keep a backup of your original model.

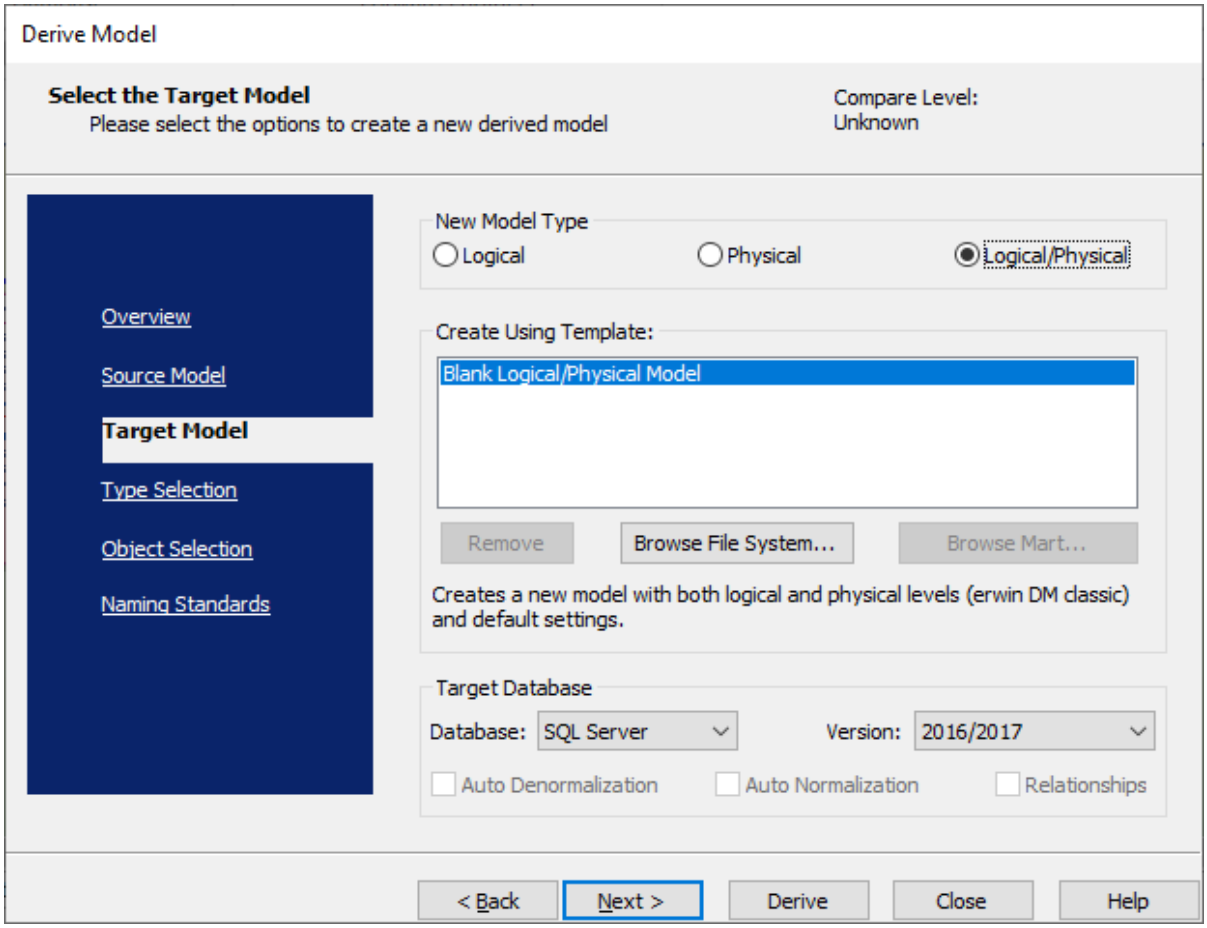
Migration by Deriving a Model

To migrate by deriving a model, follow these steps:

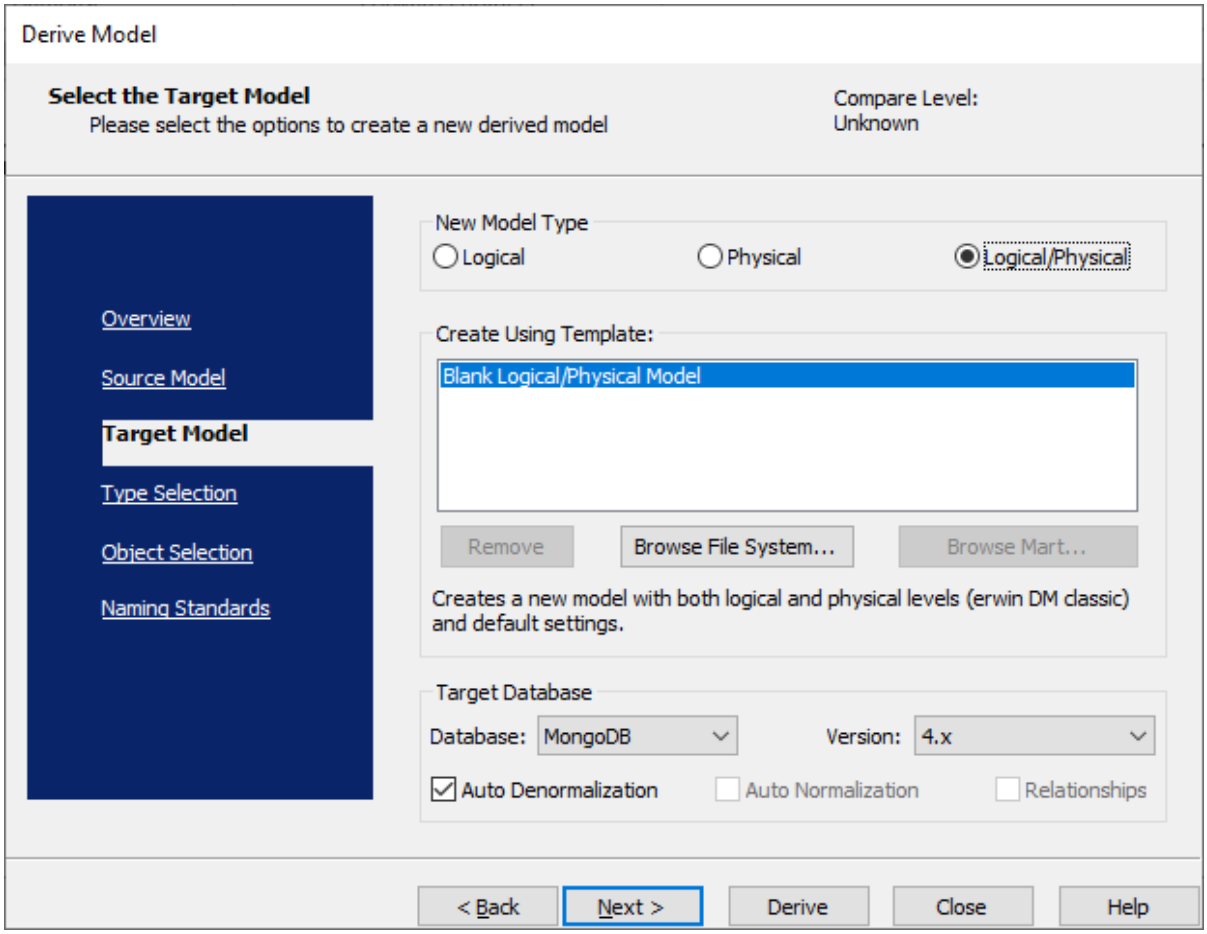
1. Open your relational model in erwin Data Modeler (DM).
   Note: Ensure that you are in the Physical mode.
   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.



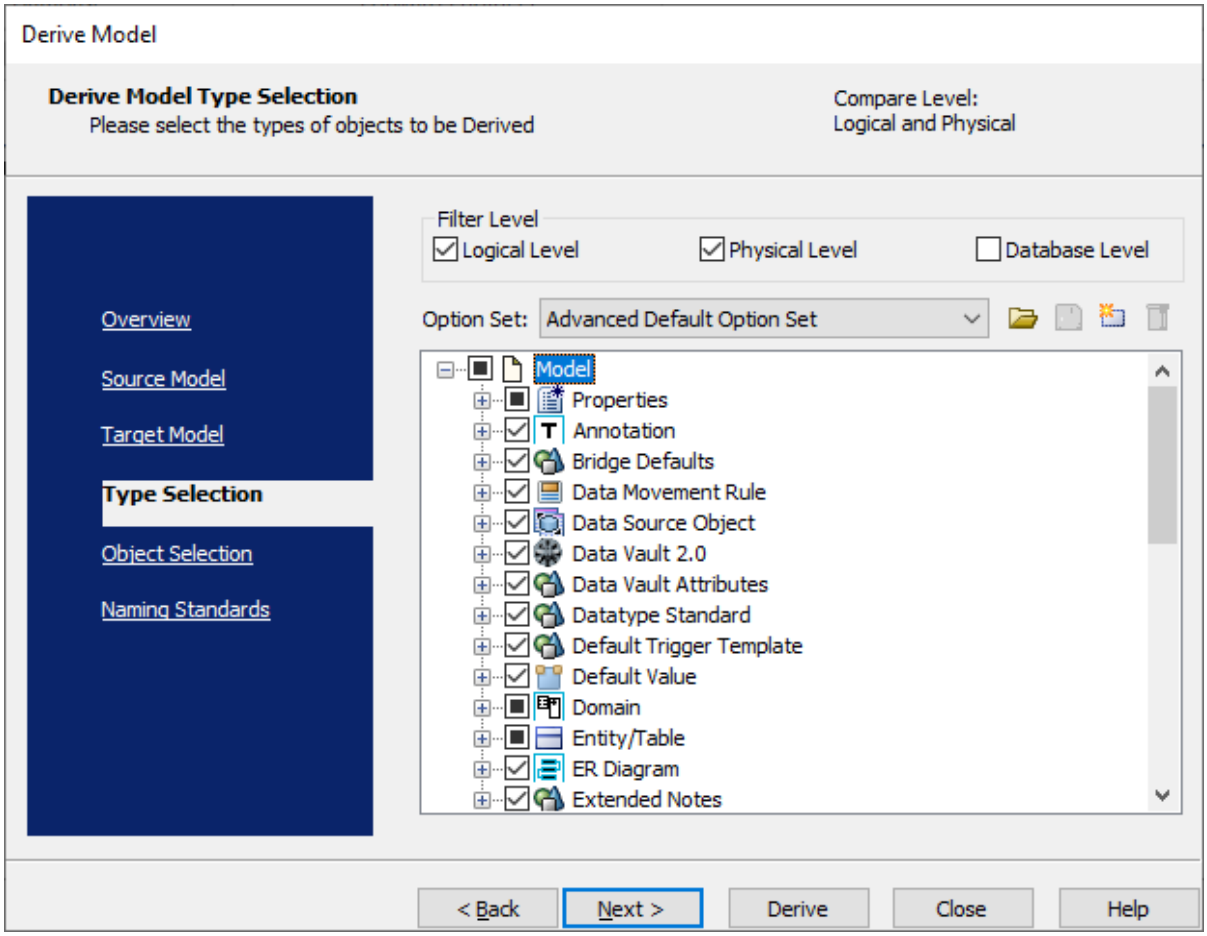
2. On the ribbon, click **Actions** > **Design Layers** > **Derive New Model**.
   The Derive Model screen appears. By default, the Source Model is set to your current model.



3. In the **Database** drop-down list, select **MongoDB**.
   By default, the Auto Denormalization check box is selected. Keep it selected.

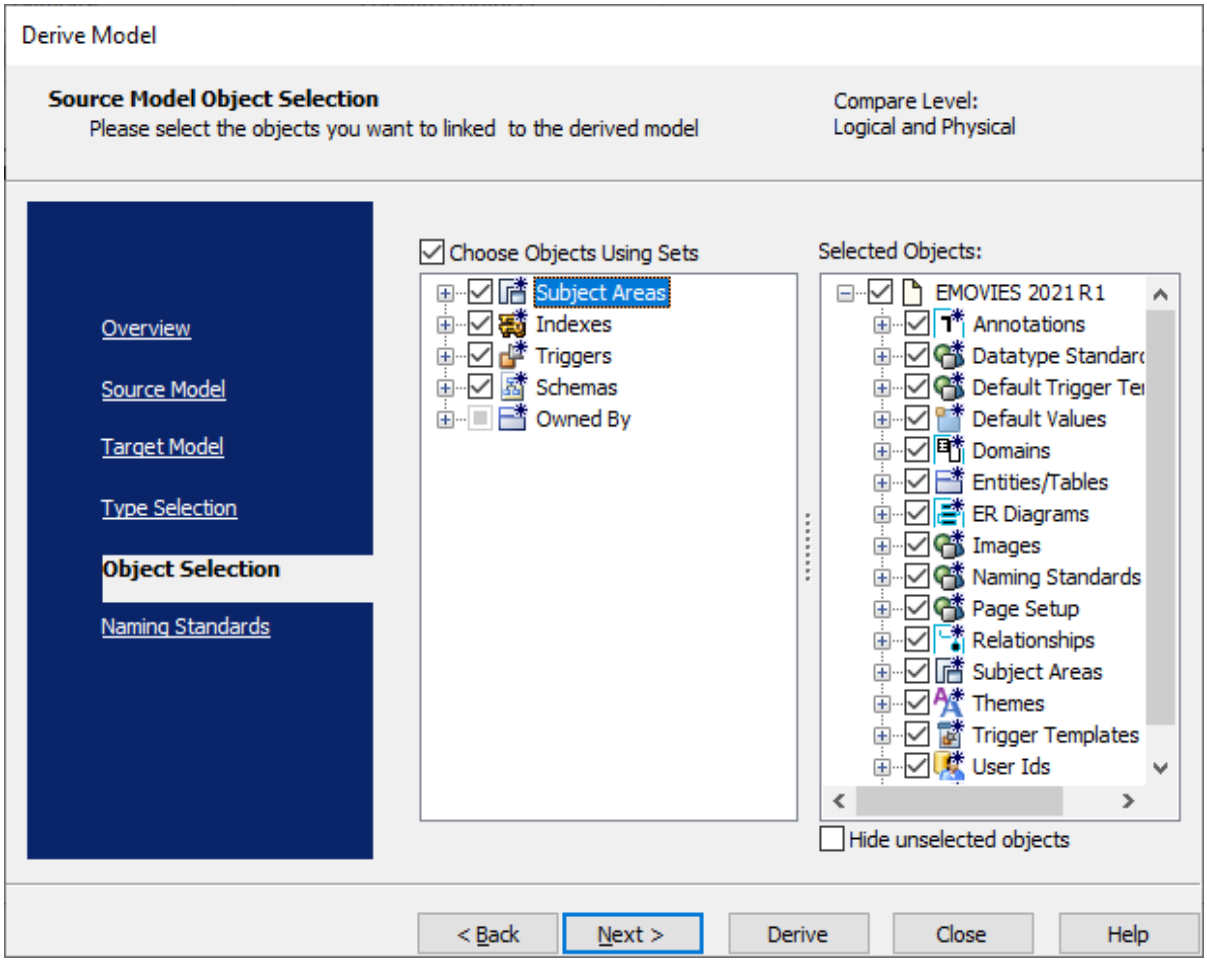


4. Click **Next**.
   Note: If the Type Resolution screen appears, click **Finish**.
   The Type Selection section appears.



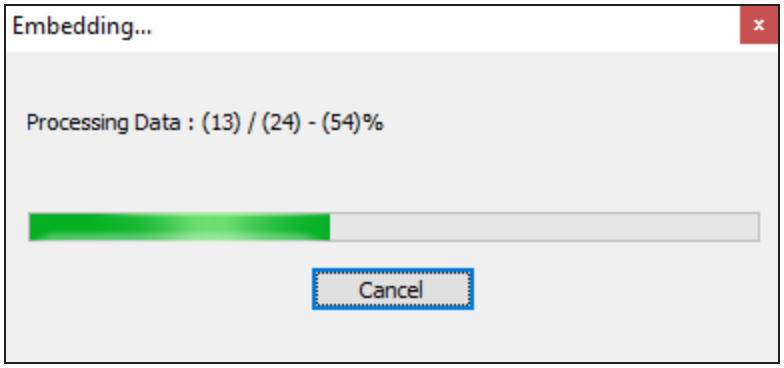
5. Select the types of objects that you want to derive into the target MongoDB model.

6. Click **Next**.
   The Object Selection section appears. Based on the object types you selected in step 5, it displays a list of objects.

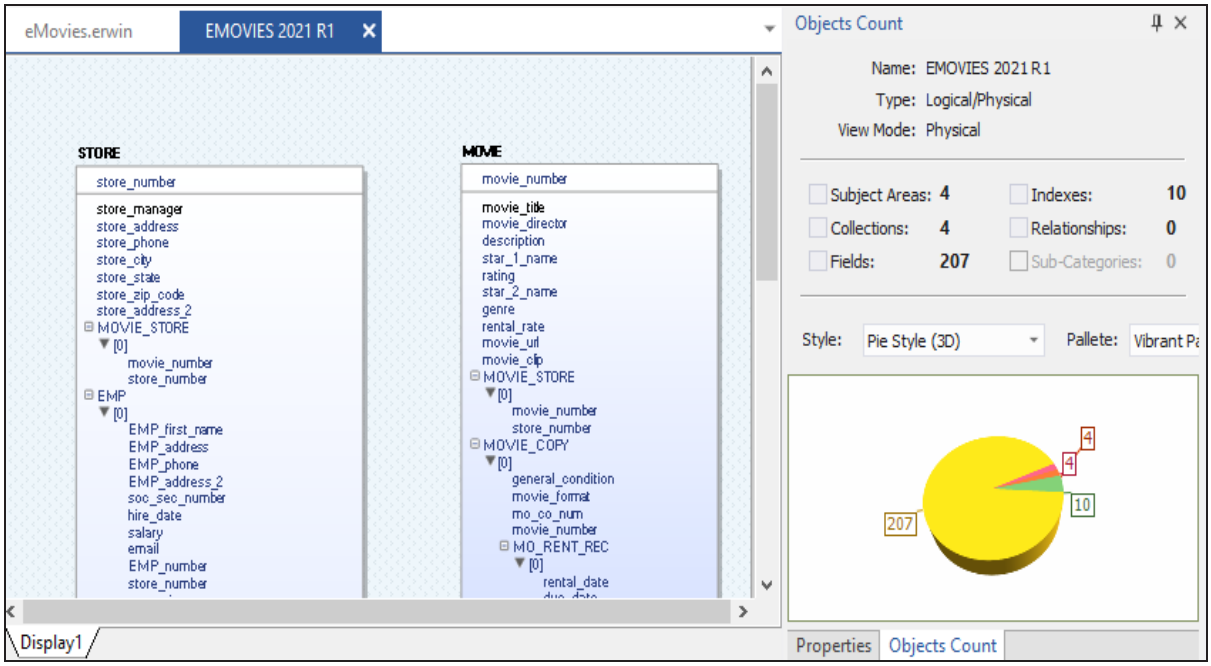


7. Select the objects that you want to derive into the target MongoDB model.

8. Click **Derive**.
   The model derivation process starts.



Once the conversion is complete, the existing model in migrated to a NoSQL database.



In the **Objects Count** pane, note that instead of tables and columns, we now have collections and fields. Also, the Relationships count has changed to 0. The migration process converts and merges multiple tables, columns, and relationships to the NoSQL format according to the database that you select.

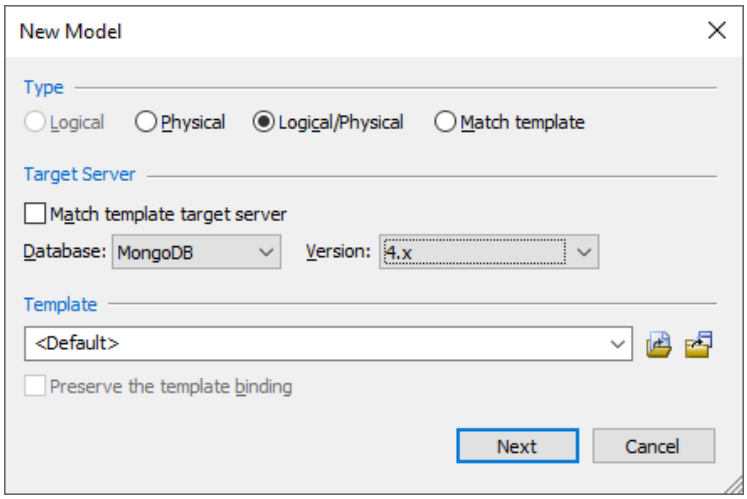
Reverse Engineering Models

You can create a data model from a database or a script using the Reverse Engineering process.

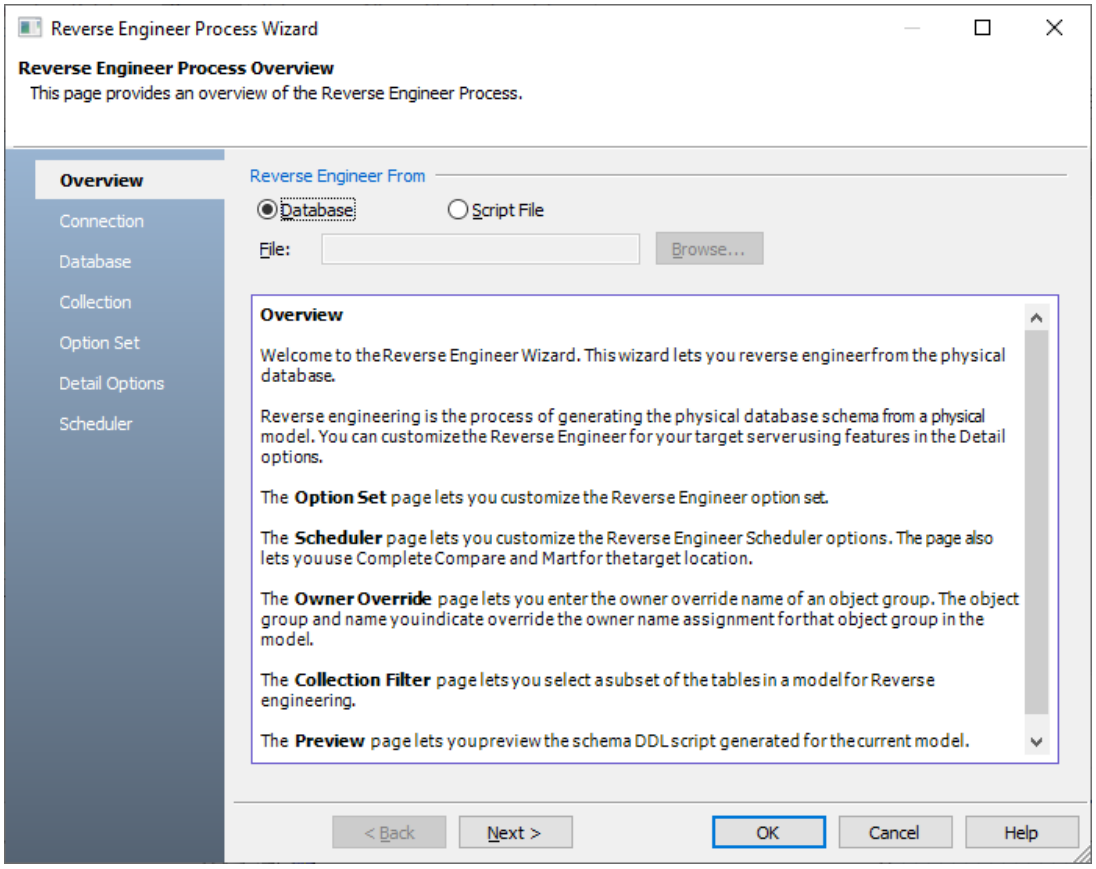
This topic walks you through the steps to reverse engineer a MongoDB model. Similarly, you can reverse engineer a model from your Cassandra Keyspace and Couchbase Bucket.

To reverse engineer a model:

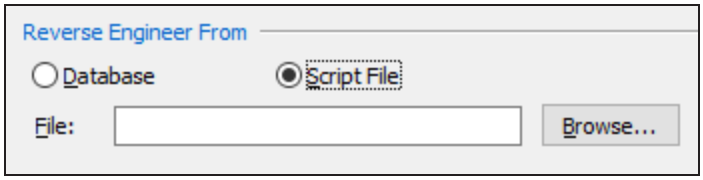
1. In erwin Data Modeler (DM), click **Actions** > **Reverse Engineer**.
   The New Model screen appears.



2. Click **Logical/Physical** and set **Database** to MongoDB.
3. Click **Next**.
   The Reverse Engineer Process Wizard appears.



4. Click one of the following options:
   - **Database**: Use this option to reverse engineer a model from your database.
   - **Script File**: Use this option to reverse engineer a model from a script. Selecting this option enables the File field. Click **Browse** and select the necessary script file.

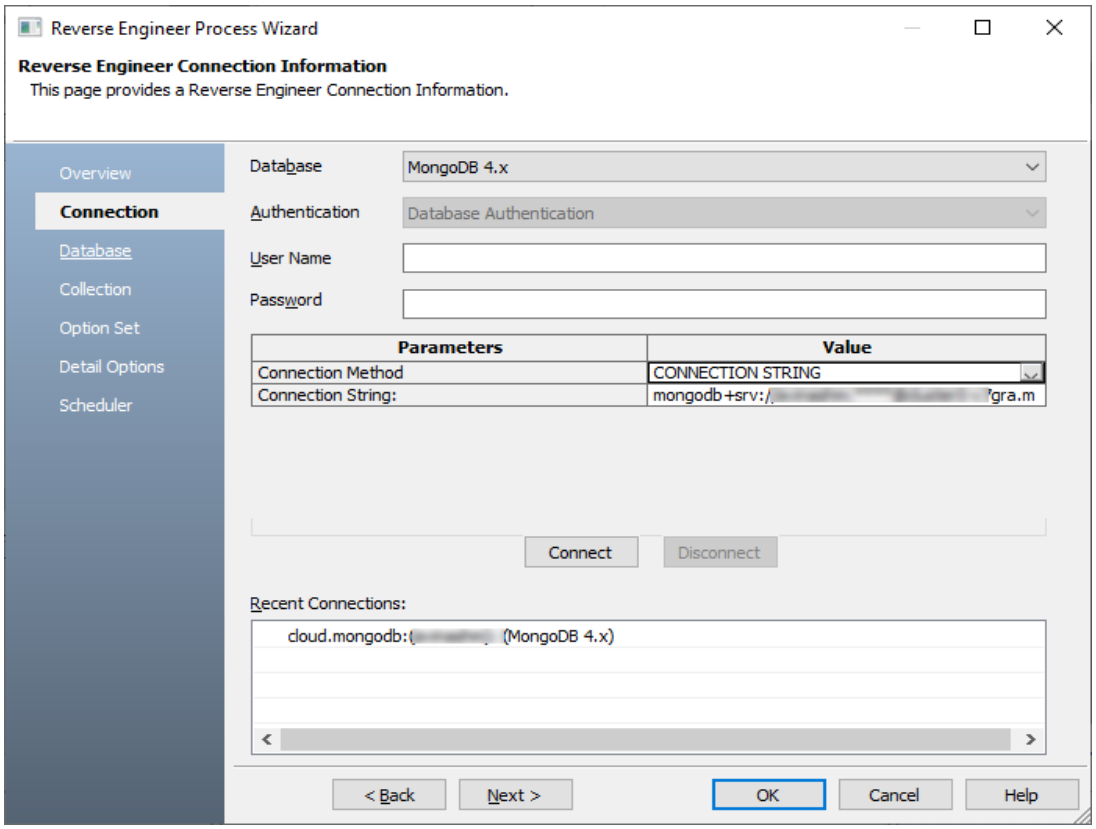
     

     Note: If you click **Script File**, jump to step 8 below and ensure that Document Count or Document % is not set to zero (0).
5. Click **Next**.

The Connection section appears. Use this section to connect to the database from which you want to reverse engineer the model. You can connect to the database directly or using a connection string. The following table explains the connection parameters:

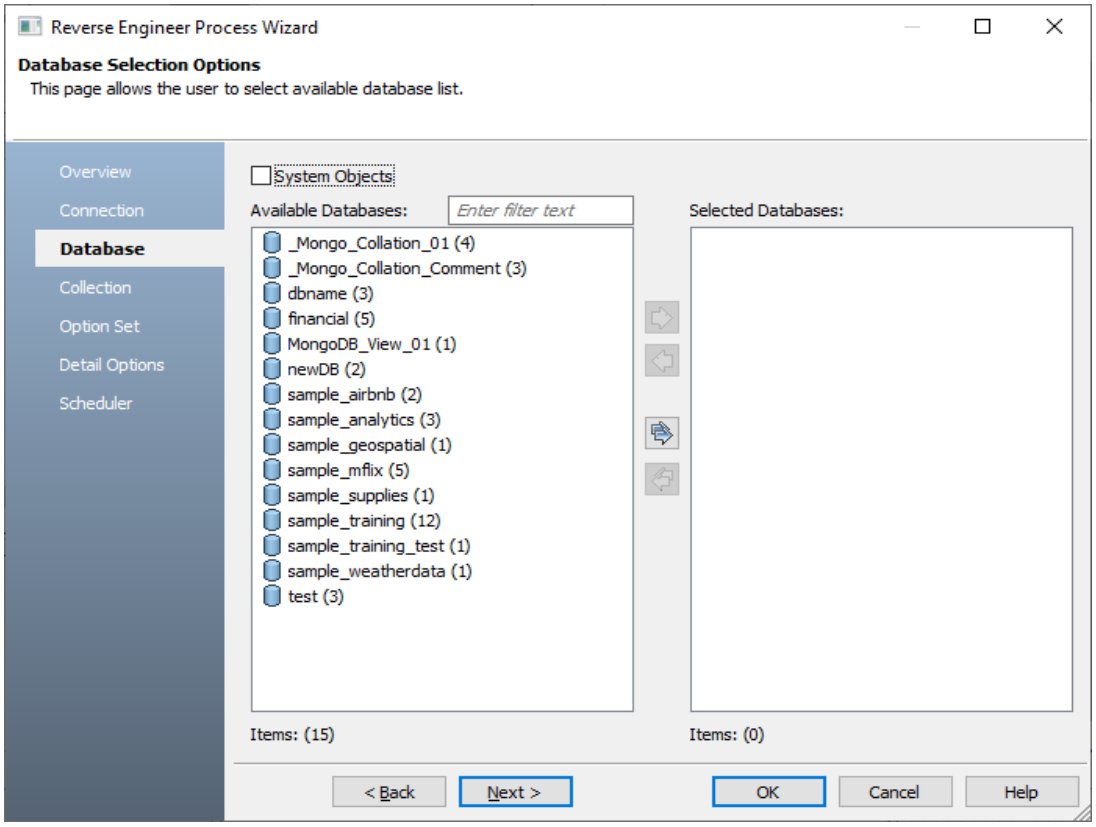
Connection Method	Parameters/Values
Connection String	Specify the MongoDB Connection String. For example: `mongodb+srv://<abcd>:****@<xyz>.mongodb.net/test?retryWrites=true&w=majority` Replace <abcd> with your username and <xyz> with host name. The host name parameter would change based on your MongoDB deployment; standalone, replica set, or a sharded cluster.
Direct	Specify the host name and port number of your MongoDB deployment. Also, specify the database that you want to connect to.

In the following image, for example, the connection is being established using a connection string.

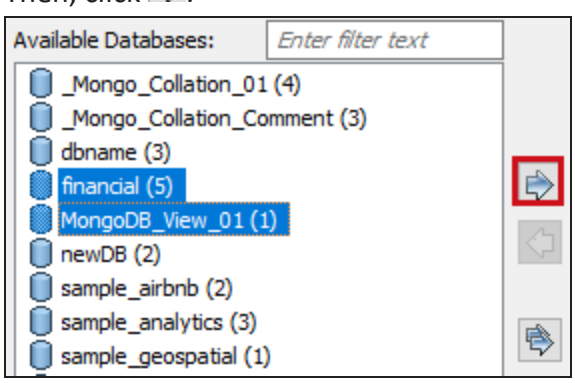


6. Click **Connect**.
   On successful connection, your connection information is displayed under Recent Connections.
7. Click **Next**.

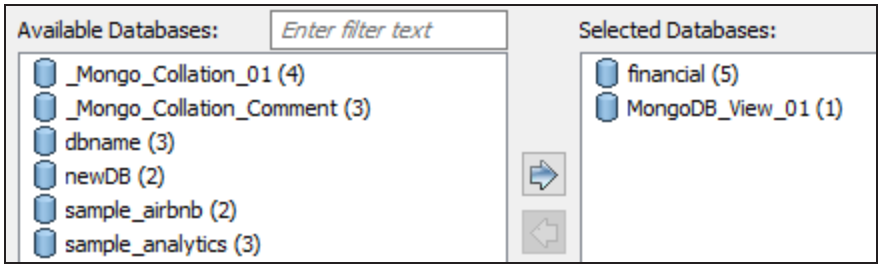
The Database section appears. It displays a list of available databases.



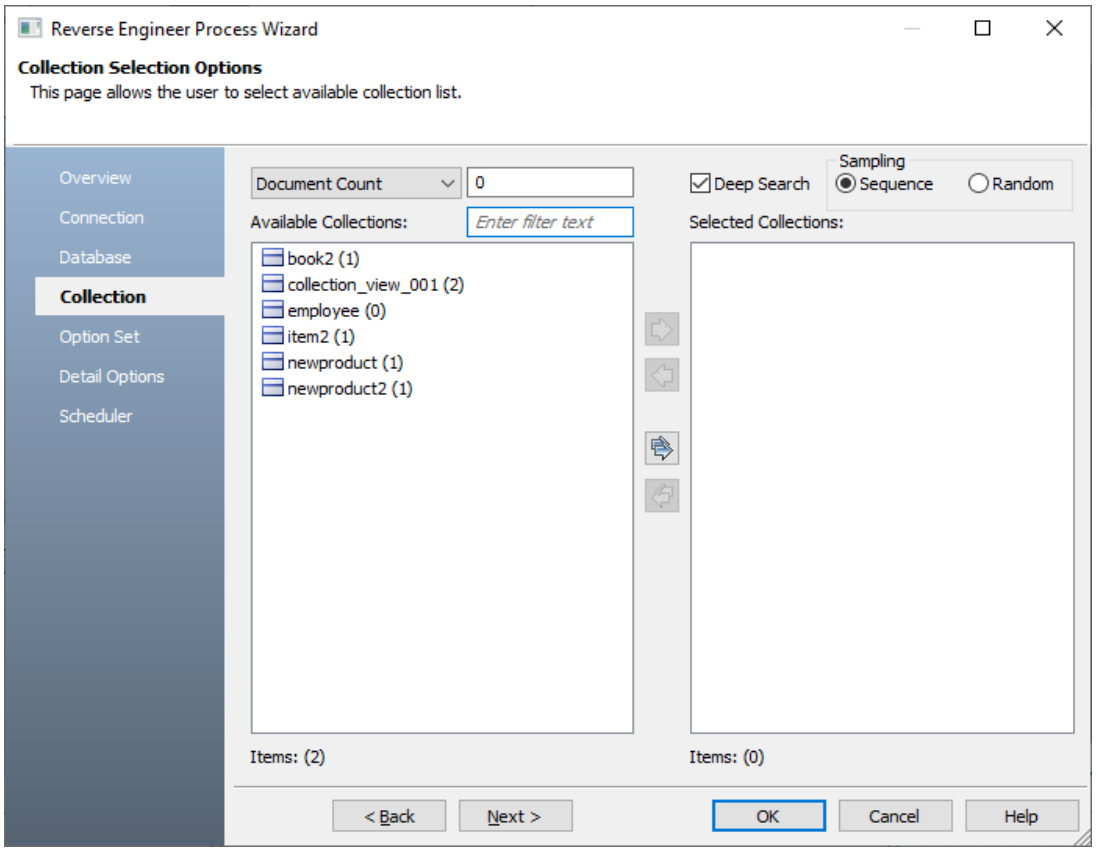
8. Under **Available Databases**, select the databases that you want to reverse engineer. Then, click .



This moves the selected databases under Selected Databases.



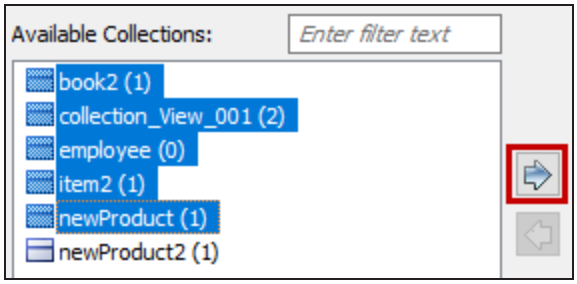
9. Click **Next**.
   The Collection section appears. It displays a list of available collections in the databases that you selected in step 8.



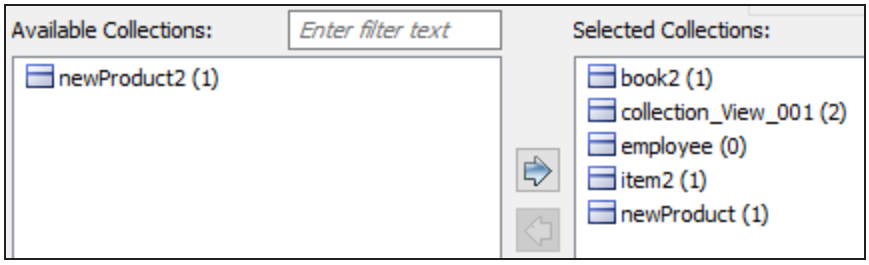
10. Use the following options:

- **Document Count/Document (%)**: Use this option to specify the number of documents or percentage of total records that the newly generated model schema would contain.
- **Deep Search**: Use this option to specify whether the deep search algorithm is used to retrieve the right samples for schema generation.
- **Sampling**: Use the Sequence or Random sampling methods to sample records in the selected collections. Sampling enables you to retrieve right estimates for accurate collection schema generation.

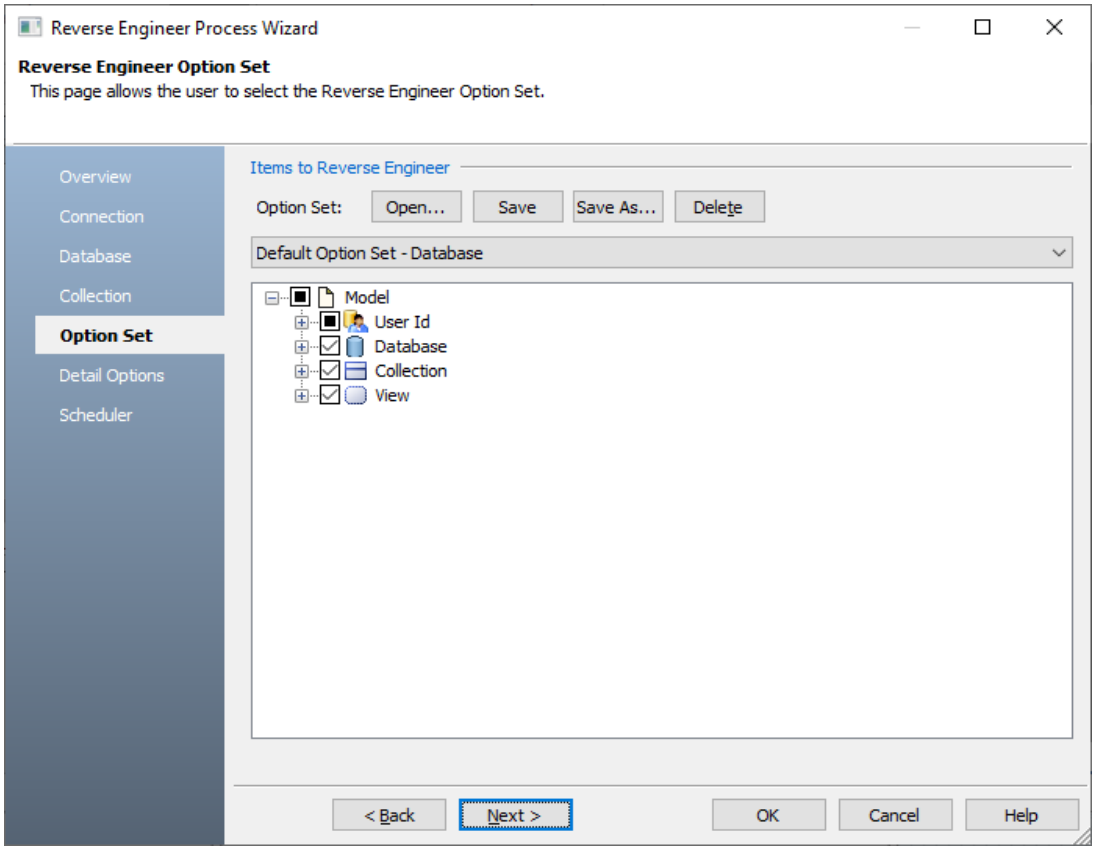
11. Under **Available Collections**, select the collections that you want to reverse engineer. Then, click .



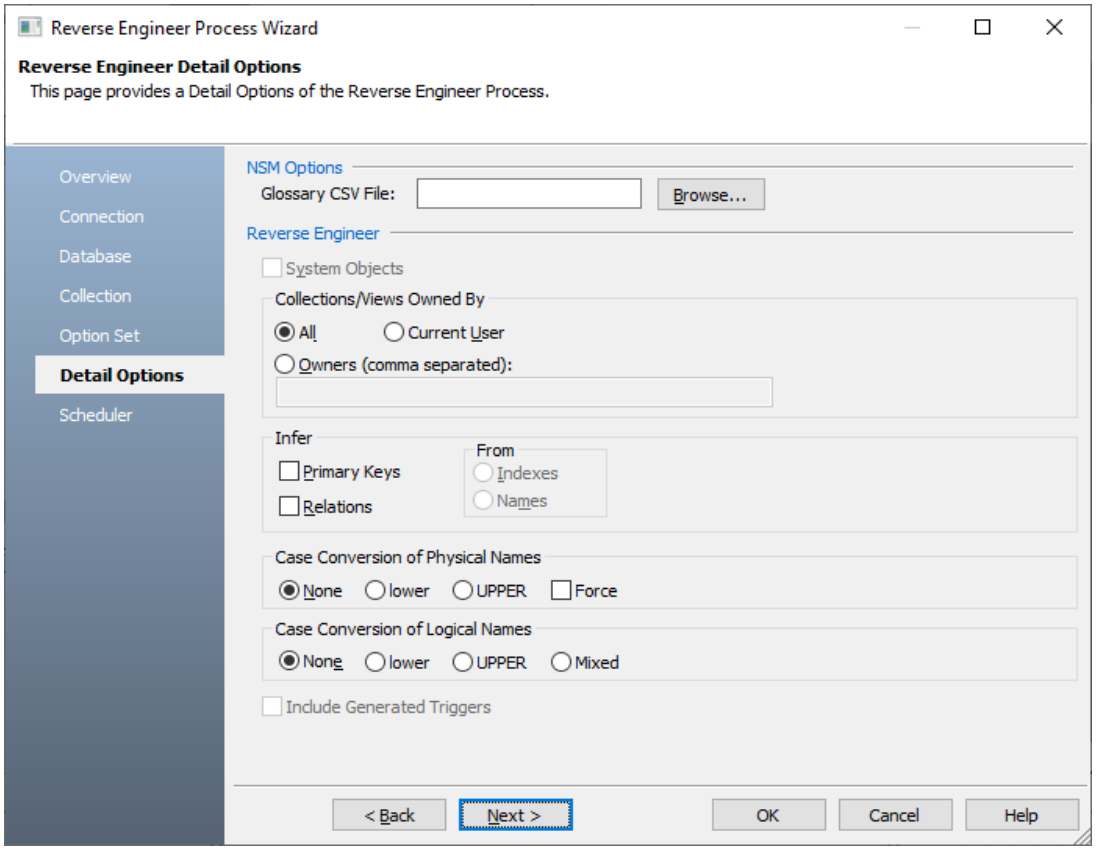
This moves the selected collections under Selected Collections.



12. Click **Next**.
The Option Set section appears. It displays the default option set. You can either use the default or a custom option set.

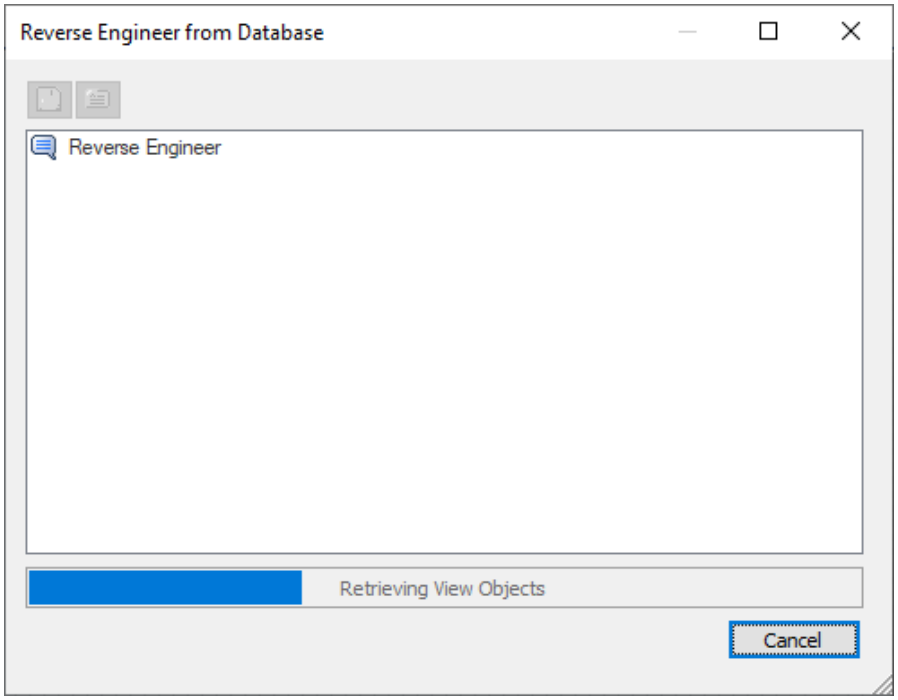


13. Click **Next**.
    The Detail Options section appears. Set up appropriate options based on your requirement.

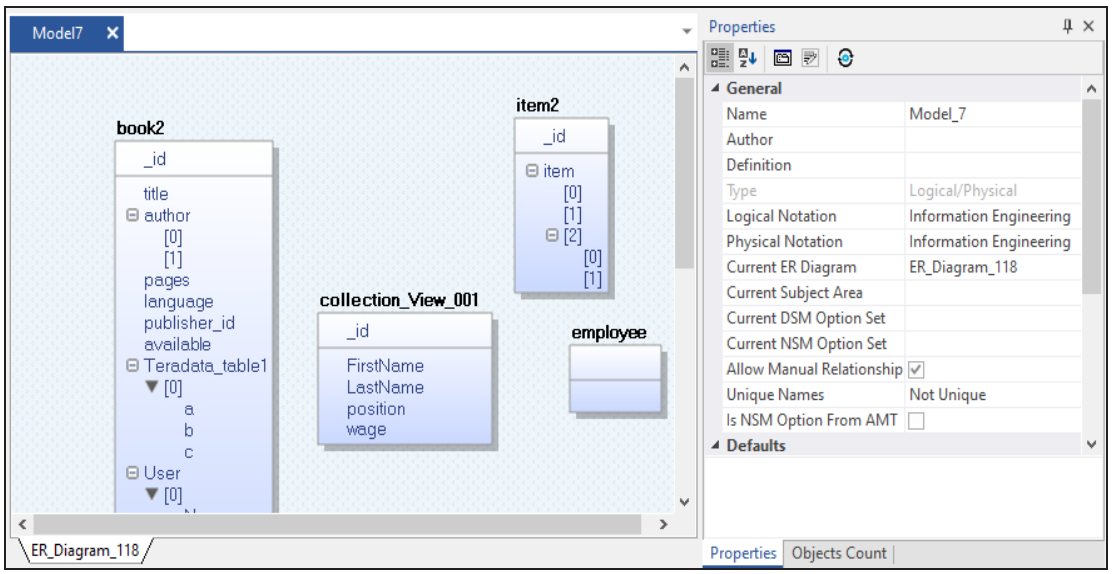

Reverse Engineer Process Wizard
Reverse Engineer Detail Options
This page provides a Detail Options of the Reverse Engineer Process.
Overview
Connection
Database
Collection
Option Set
Detail Options
Scheduler
NSM Options
Glossary CSV File:
Browse...
Reverse Engineer
System Objects
Collections/Views Owned By
All
Current User
Owners (comma separated):
Infer
Primary Keys
Relations
From
Indexes
Names
Case Conversion of Physical Names
None
lower
UPPER
Force
Case Conversion of Logical Names
None
lower
UPPER
Mixed
Include Generated Triggers
< Back
Next >
OK
Cancel
Help

14. Click **OK**.

The reverse engineering process starts.



Once the process is complete, based on your selections, a schema is generated and a model is created.



Forward Engineering Models

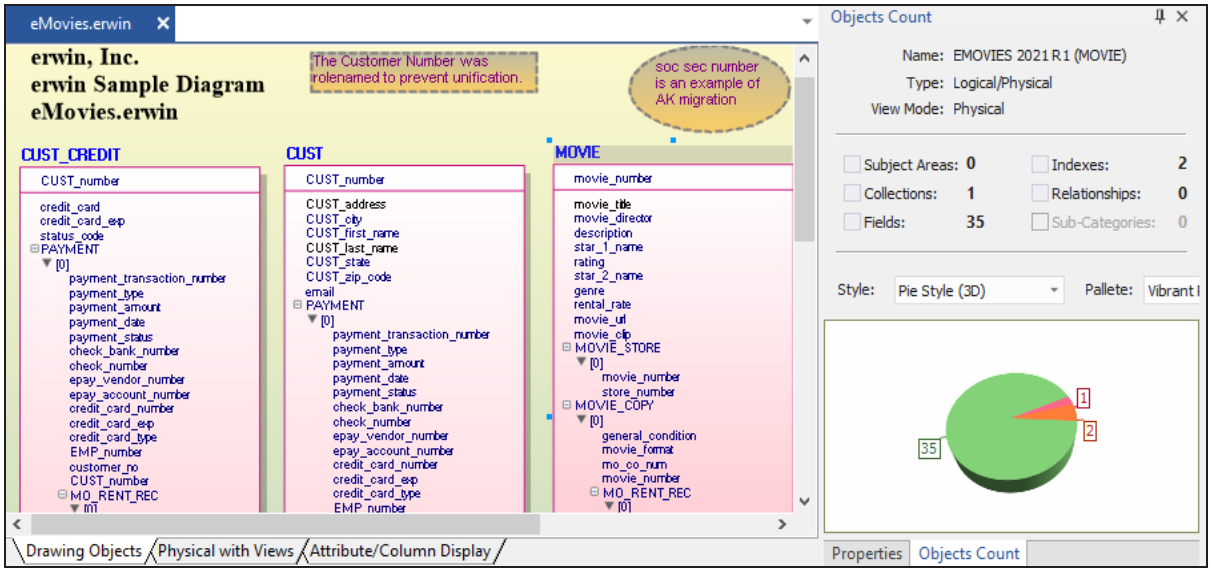
You can generate a physical database schema from a physical model using the Forward Engineering process.

This topic walks you through the steps to forward engineer a MongoDB model. Similarly, you can forward engineer a model to your Cassandra Keyspace and Couchbase Bucket.

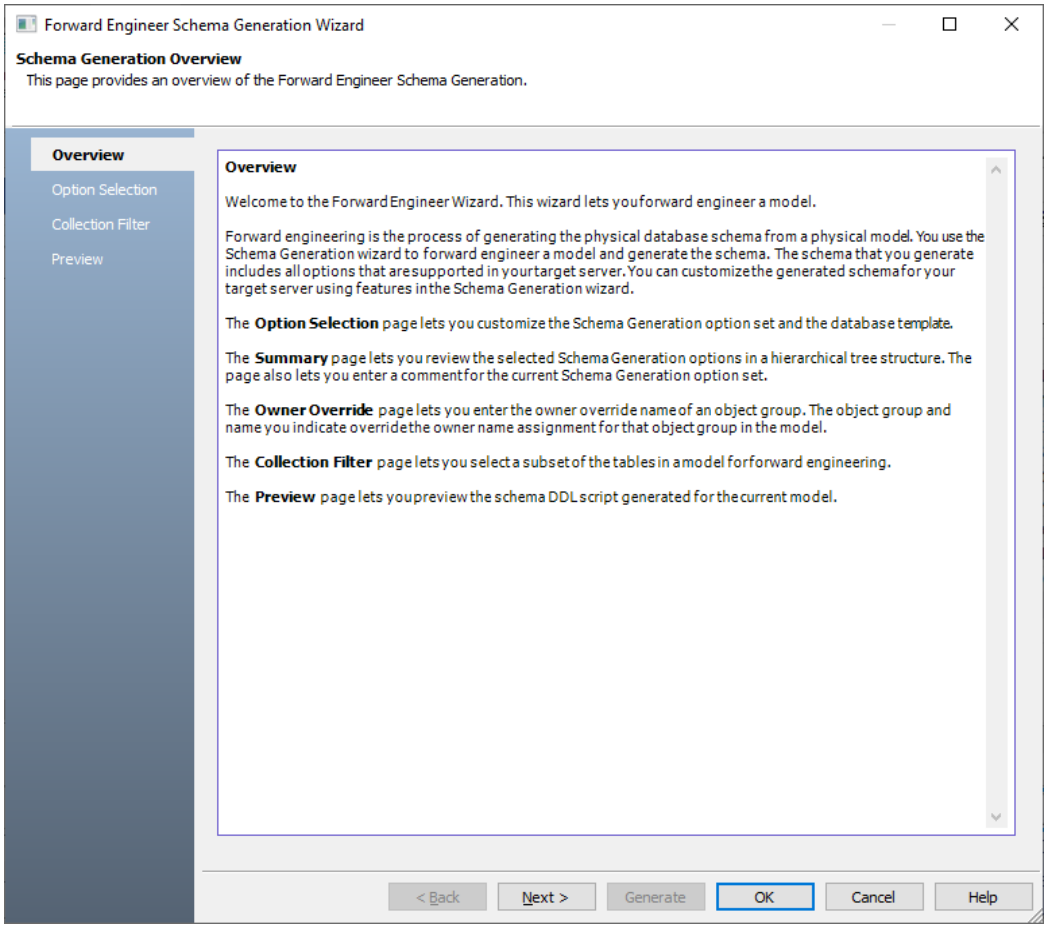
To forward engineer a model:

1. Open your MongoDB model in erwin Data Modeler (DM).
   Note: Ensure that you are in the Physical mode.

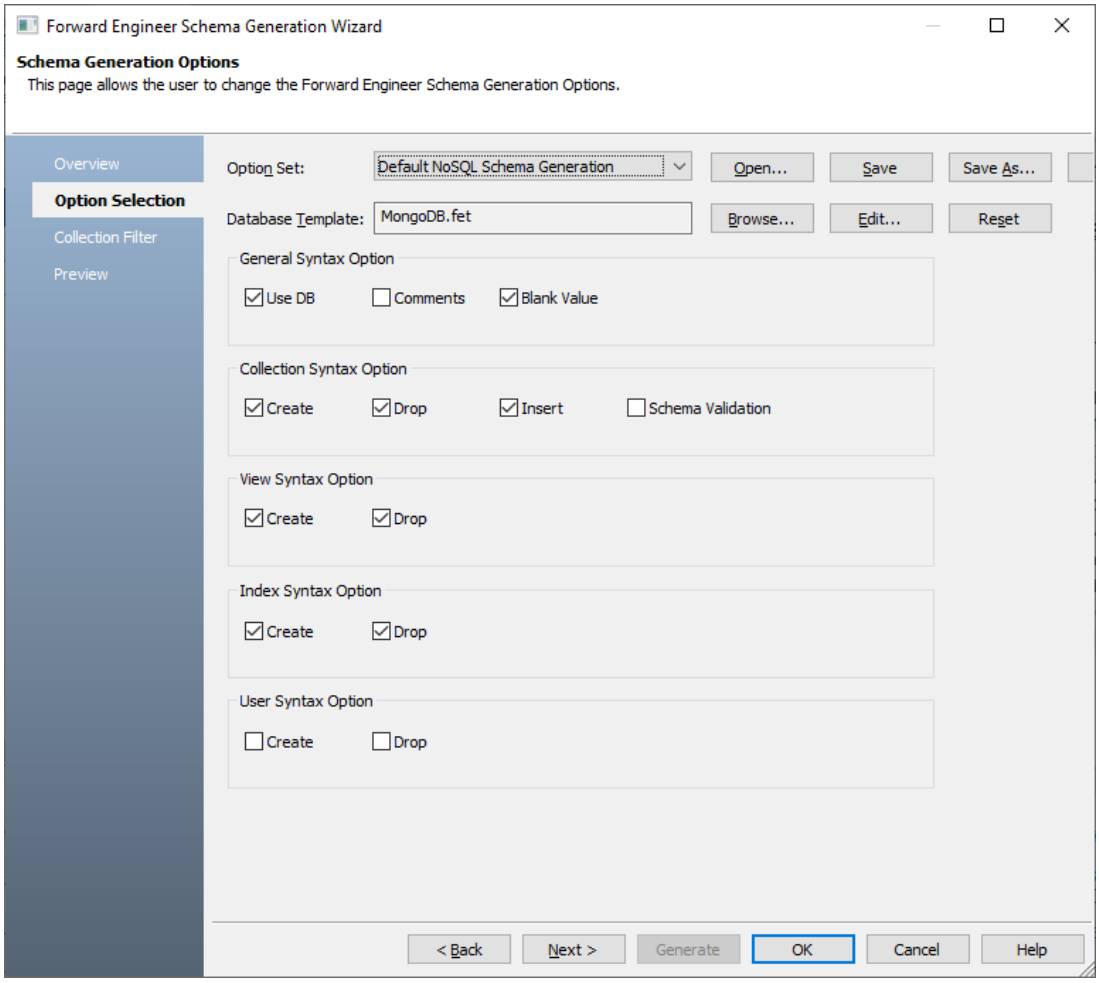
   For example, the following image uses a MongoDB model with two collections.



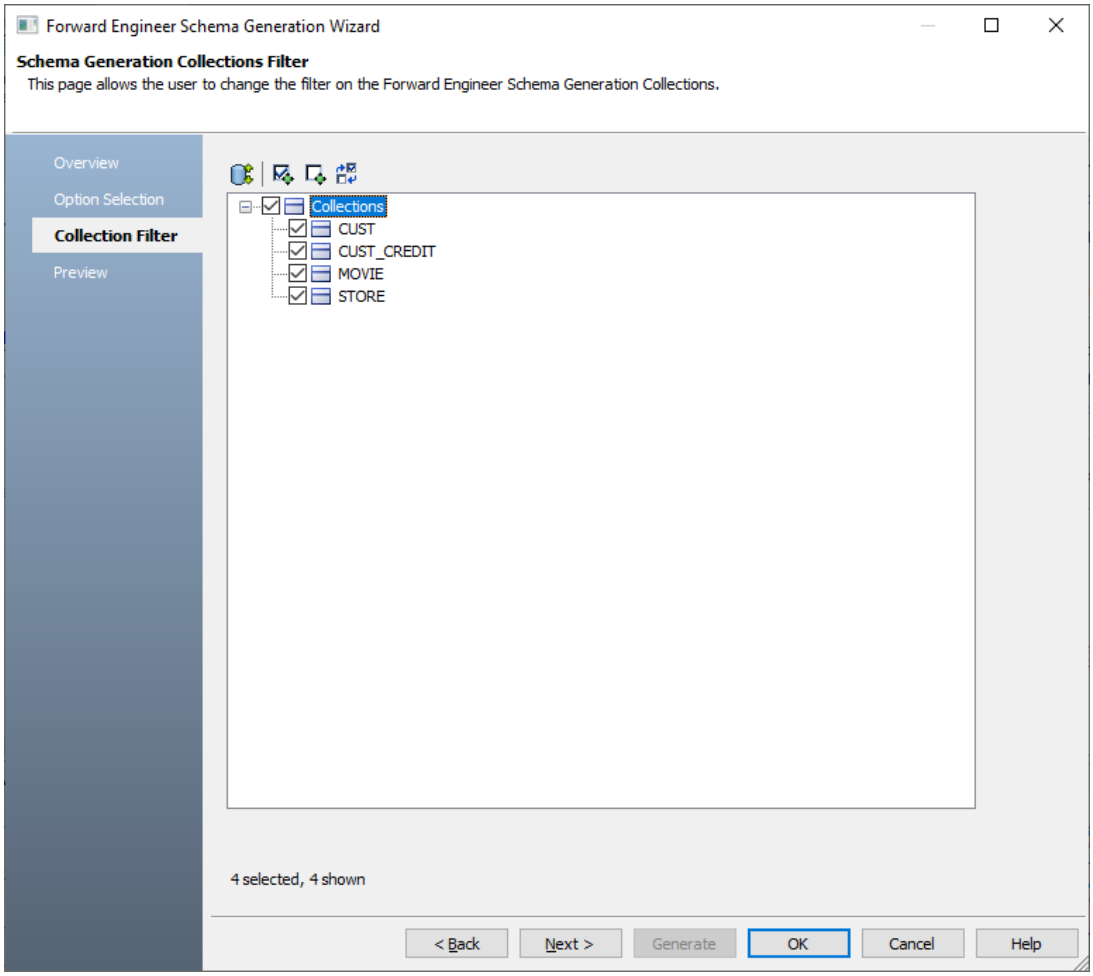
2. Click **Actions** > **Schema**.
   The Forward Engineer Schema Generation Wizard appears.



3. Click **Option Selection**.
   The Option Selection section displays the default option set. Clear the **Drop** check boxes and select other syntax check boxes as required.

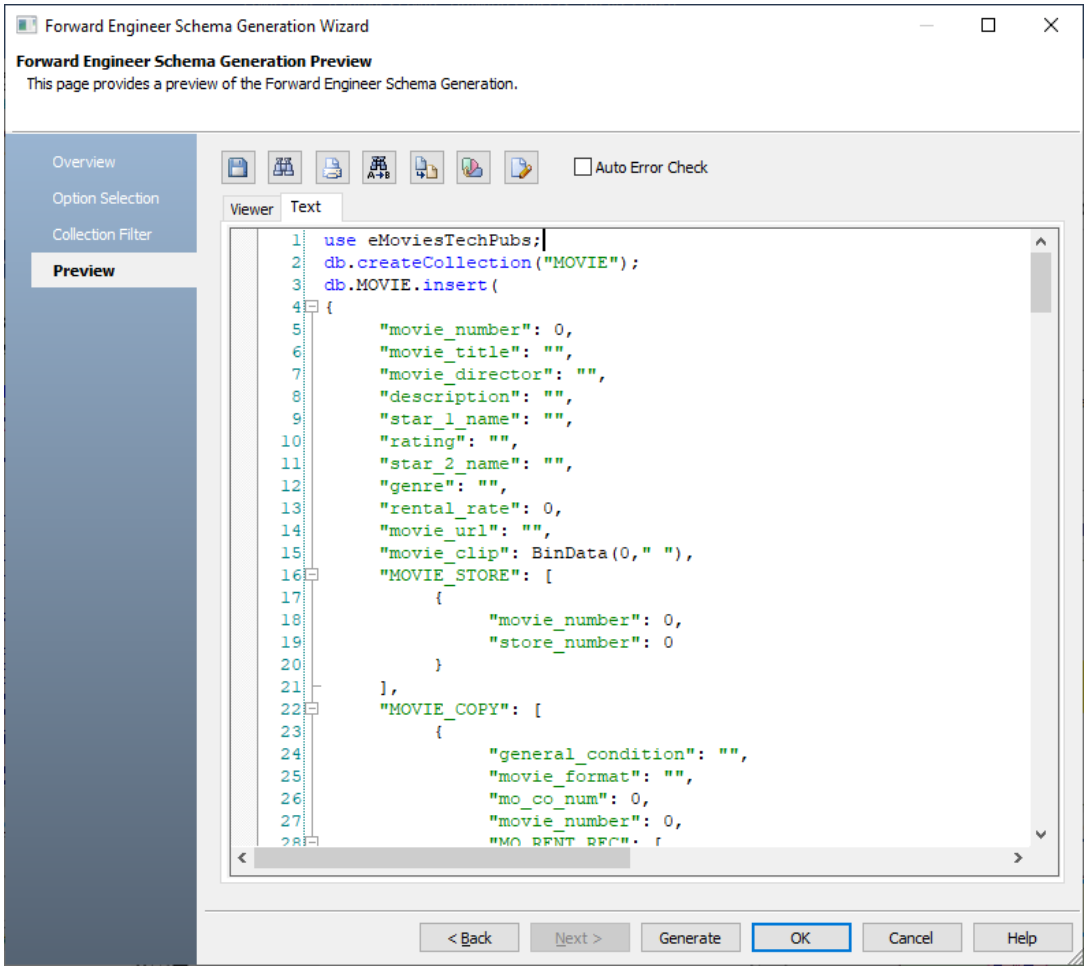


4. Click **Next**.
   The Collection Filter section appears. It displays a list of collections available in your model.



5. Select the collections that you want to forward engineer.

6. Click **Preview** to view the schema script.



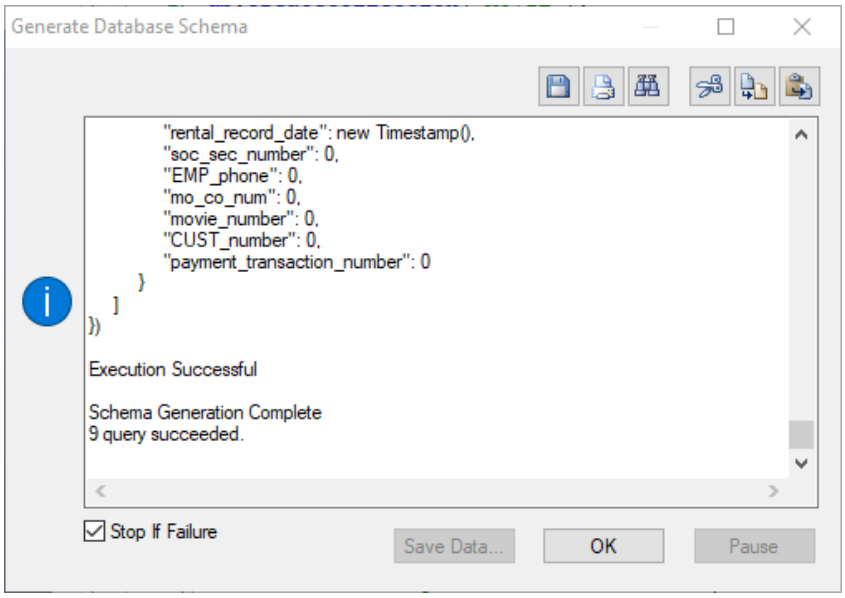
Use the following options:

- **Auto Error Check**: Select this option to enable auto error check by the forward engineering wizard.
- **Error Check** (): Use this option to run an error check. Based on the results, you can correct the generated script.
- **Text Options** (): Use this option to configure the preview text editor's look and feel, such as window, font, syntax color settings. For more information,

refer to the Forward Engineering Wizard - Preview Editor topic.

- **Save** (): Use this option to save the generated script in the JSON or BSON format.

7. Click **Generate**.
   The forward engineering process starts. The script generates your physical database schema. You can access your database and verify the newly generated schema.



Comparing Changes using Complete Compare

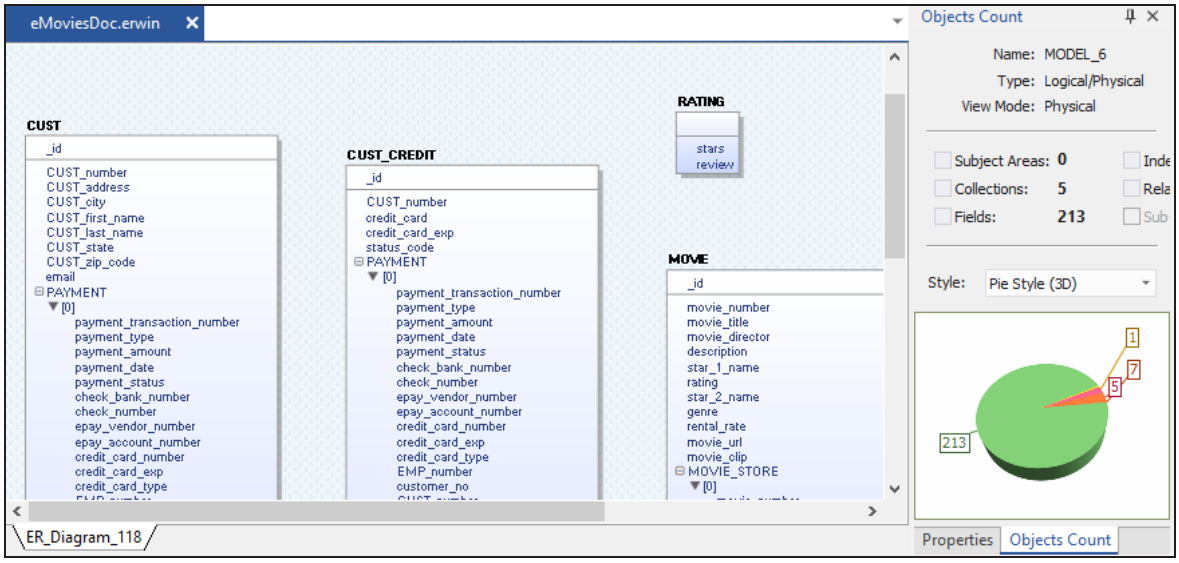
You can compare your model with database, script, or another local model to check for differences using the Complete Compare wizard. Based on the results, you can then resolve or merge differences. Thus, maintaining a consistent model and database.

This topic walks you through the steps to compare a MongoDB model with database. Similarly, you can compare your Cassandra and Couchbase models.

To compare models with database:

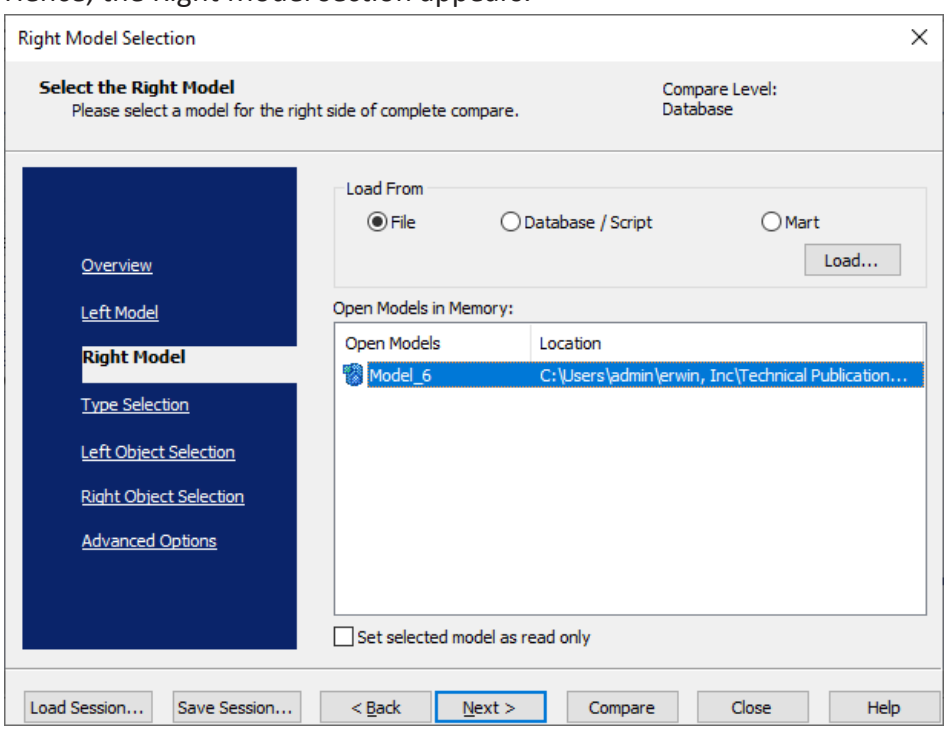
1. Open your MongoDB model in erwin Data Modeler (DM).
   Note: Ensure that you are in the Physical mode.

For example, the following image uses a MongoDB model with two collections.

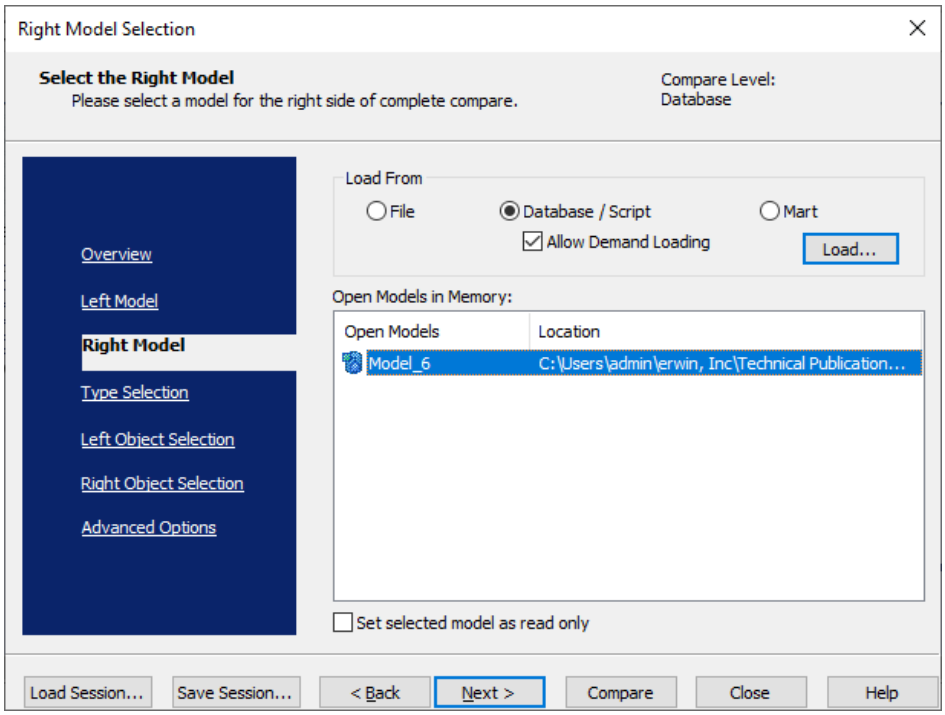


2. Click **Actions** > **Complete Compare**.

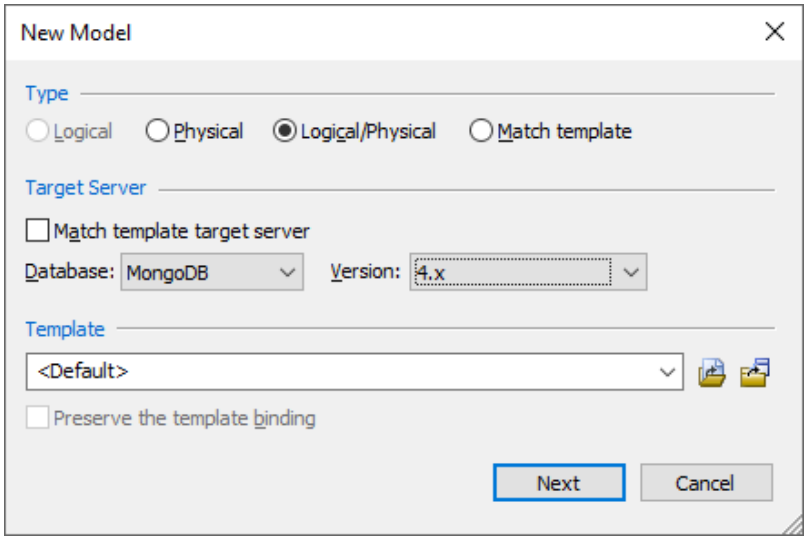
   By default, the Complete Compare wizard assigns the open model as the Left Model. Hence, the Right Model section appears.



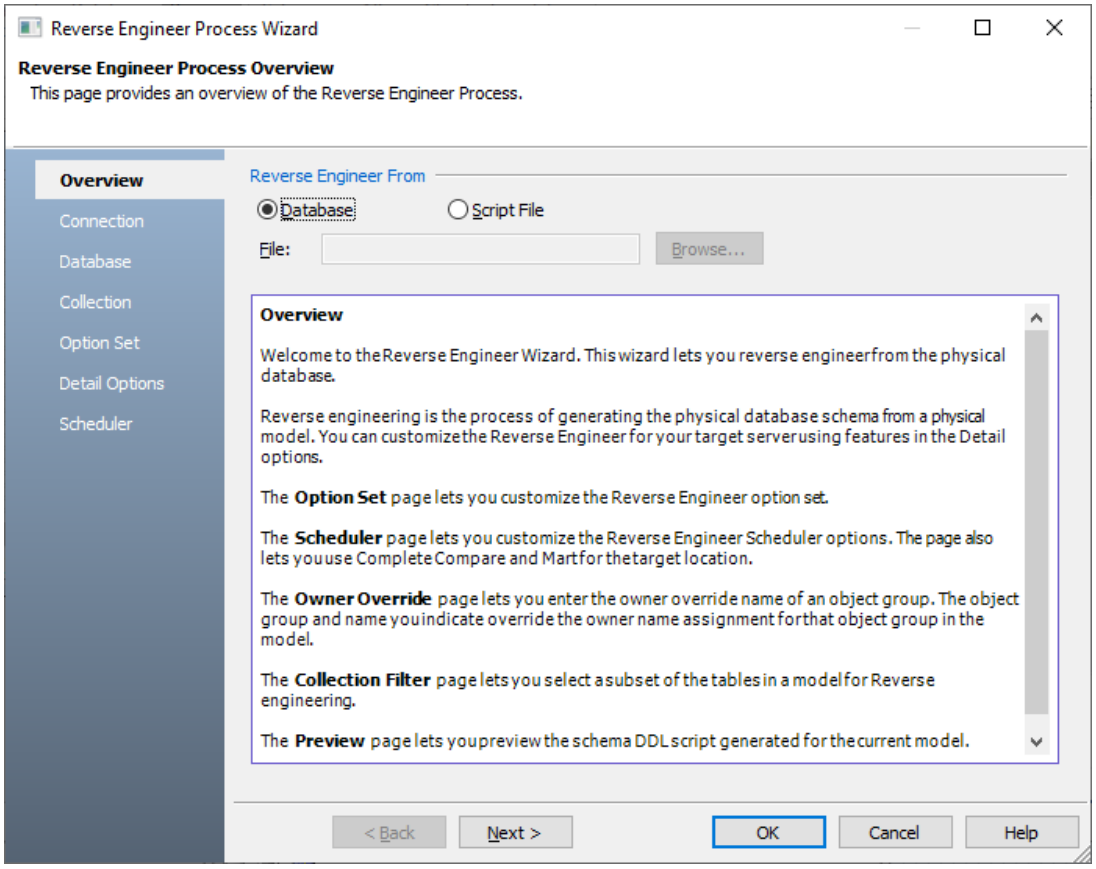
3. Click **Database/Script**.
   By default, the Allow Demand Loading option is selected.



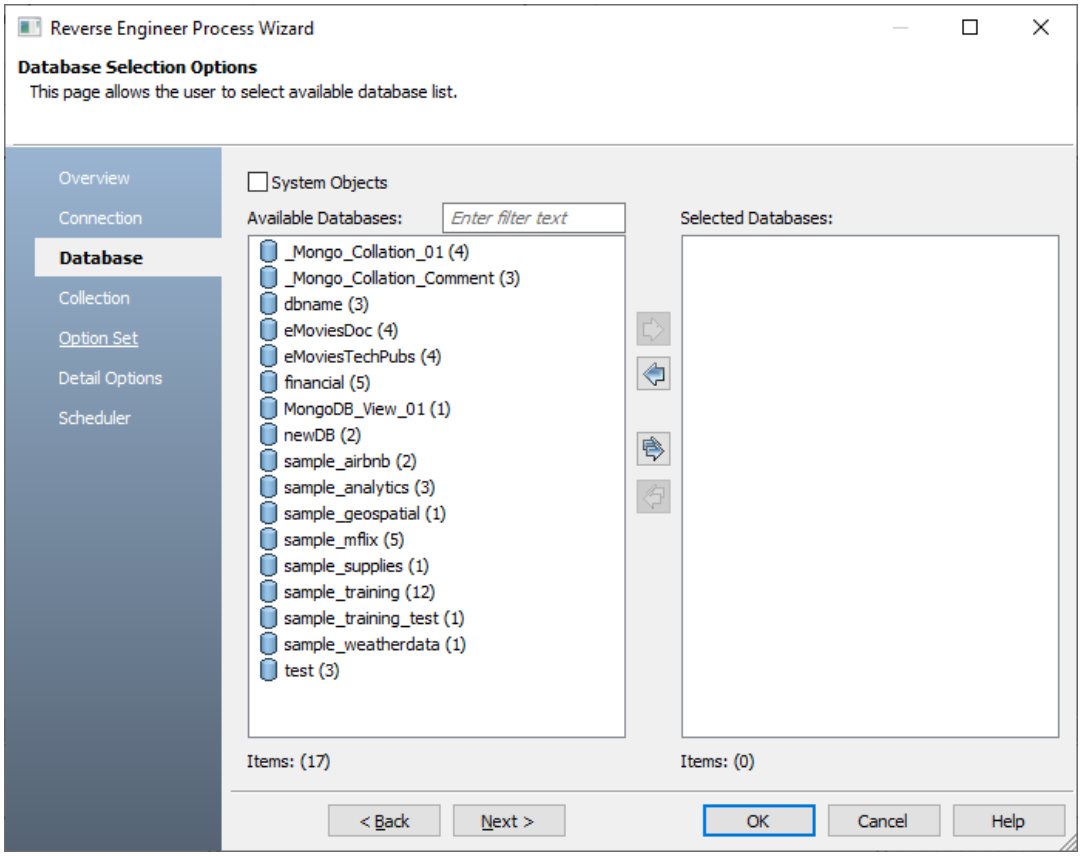
4. Click **Load**.
   The New Model dialog box appears. This starts the reverse engineering process to pull a model from the database to compare.



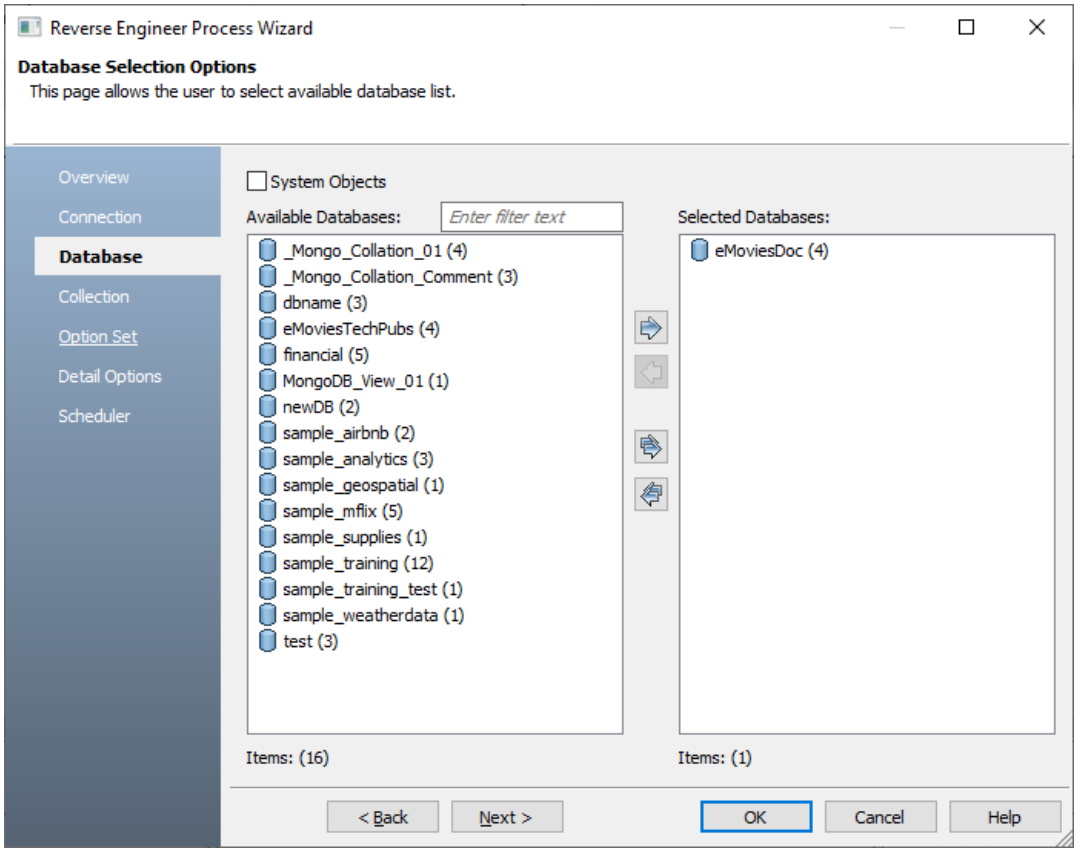
5. Ensure that the Database is set to the correct one. In this case, MongoDB. Then, click **Next**.
   The Reverse Engineer Process Wizard appears.



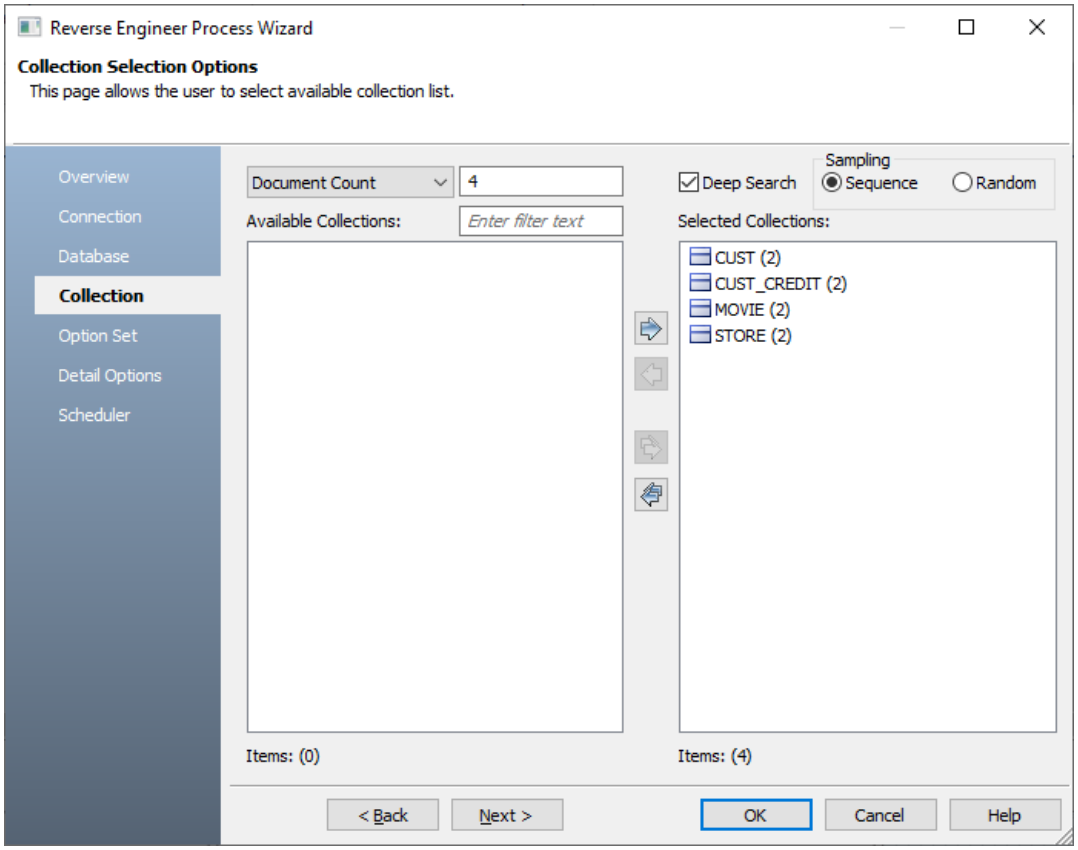
6. Click **Database**. Then, click **Next**.
   The Connection section appears. Use this section to connect to the database from which you want to reverse engineer the model.
7. After connection is established, click **Next**.
   The Database section appears. It displays a list of available databases.



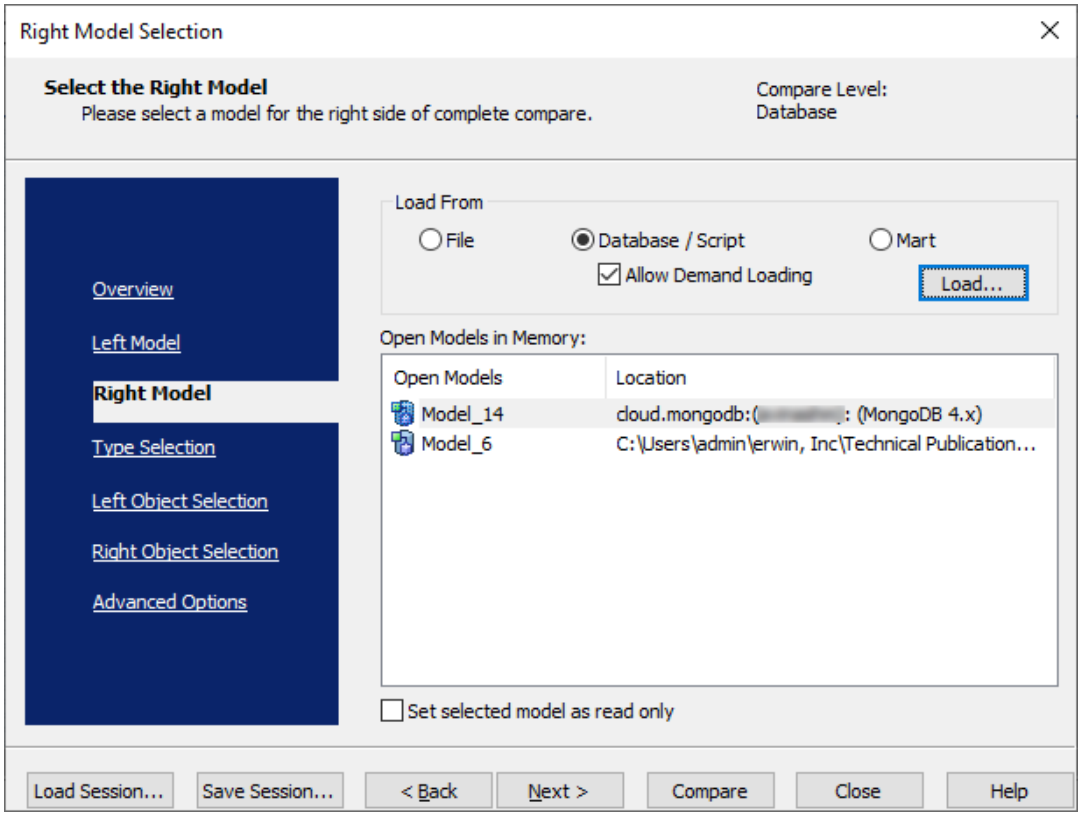
8. Under **Available Databases**, select the databases that you want to reverse engineer. Then, click .
   This moves the selected databases under Selected Databases.



9. Click **Next** and in the Collection section, click  .
   This selects all the available collections. Also, ensure that the Document Count/Document % is not set to zero (0).

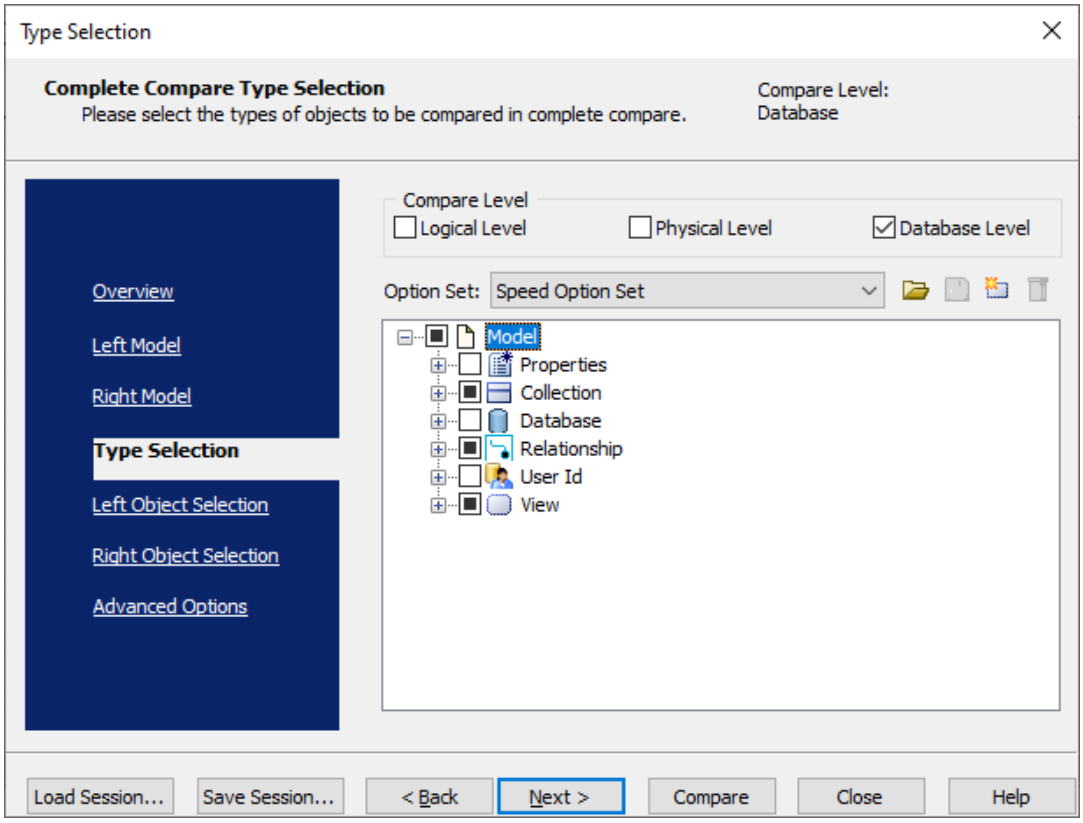


10. Click **Next** and in the Option Set section, keep the default configuration.
11. Click **Next** and in the Detail Options section, keep the default configuration.
12. Click **OK**.
    The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse engineered.



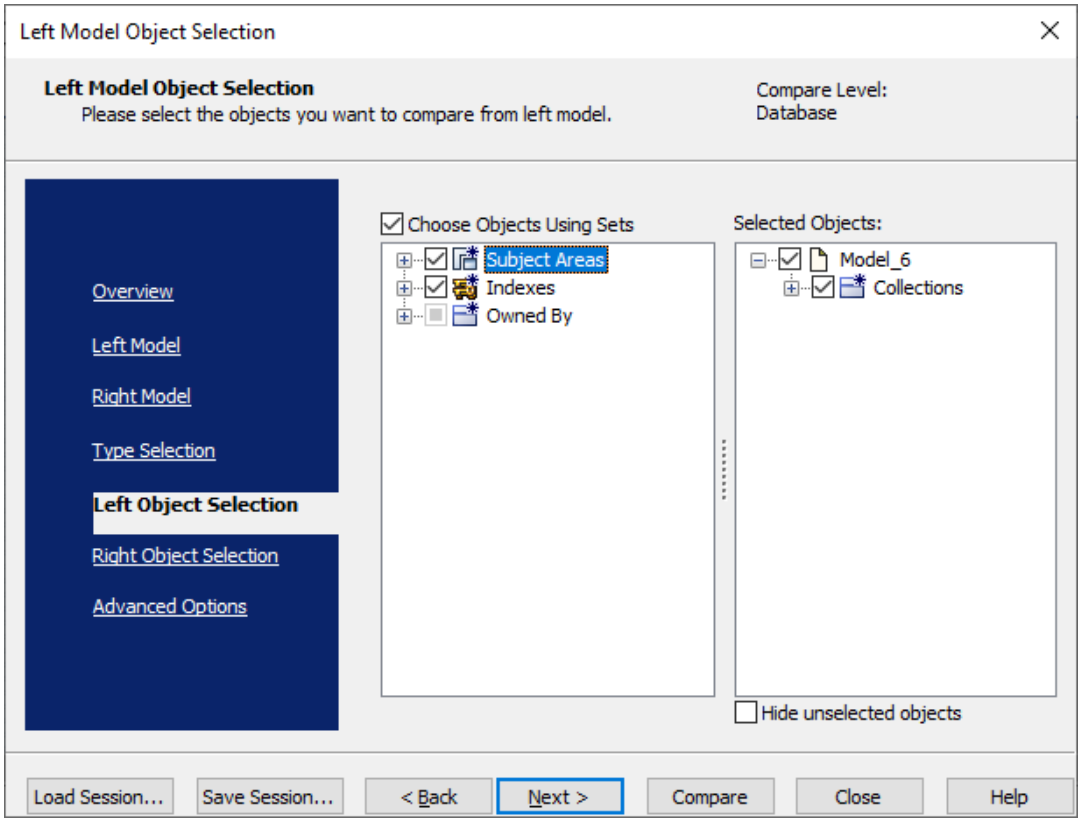
13. Click **Next** and in the Type Selection section, select the appropriate options.

For example, the following image shows the default options.



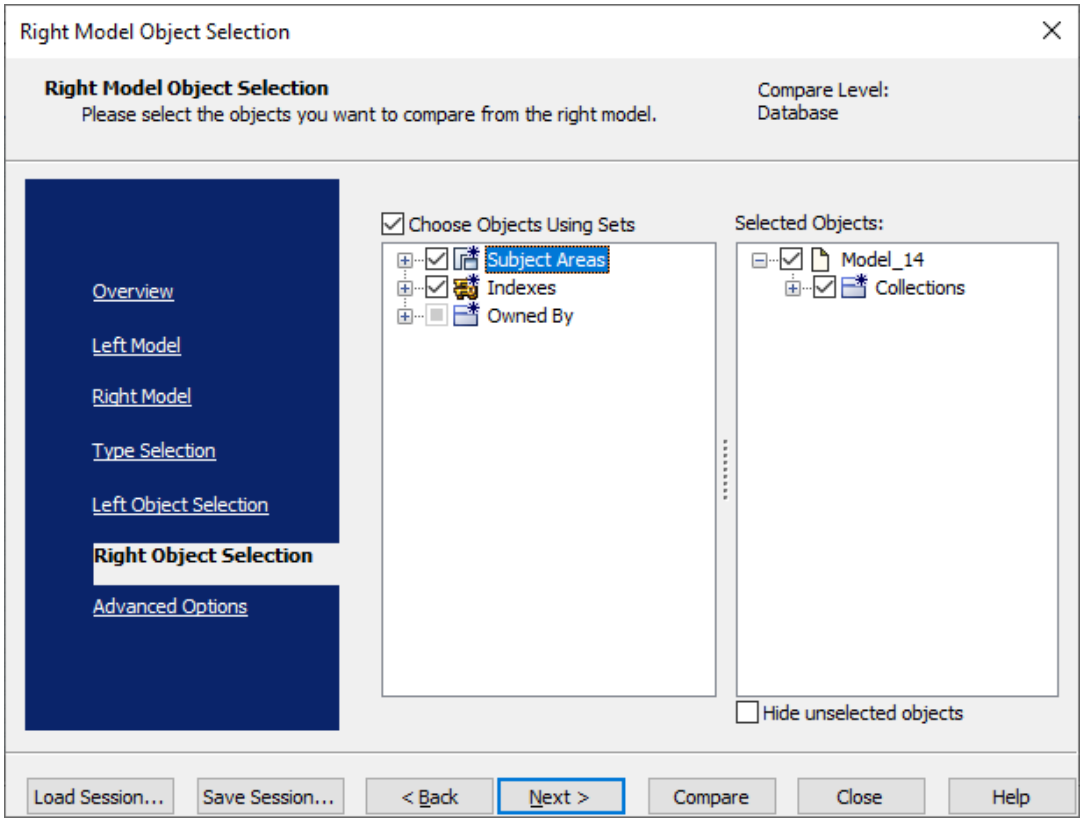
14. Click **Next** and in the Left Object Selection section, select the appropriate options.

For example, the following image shows the default options.

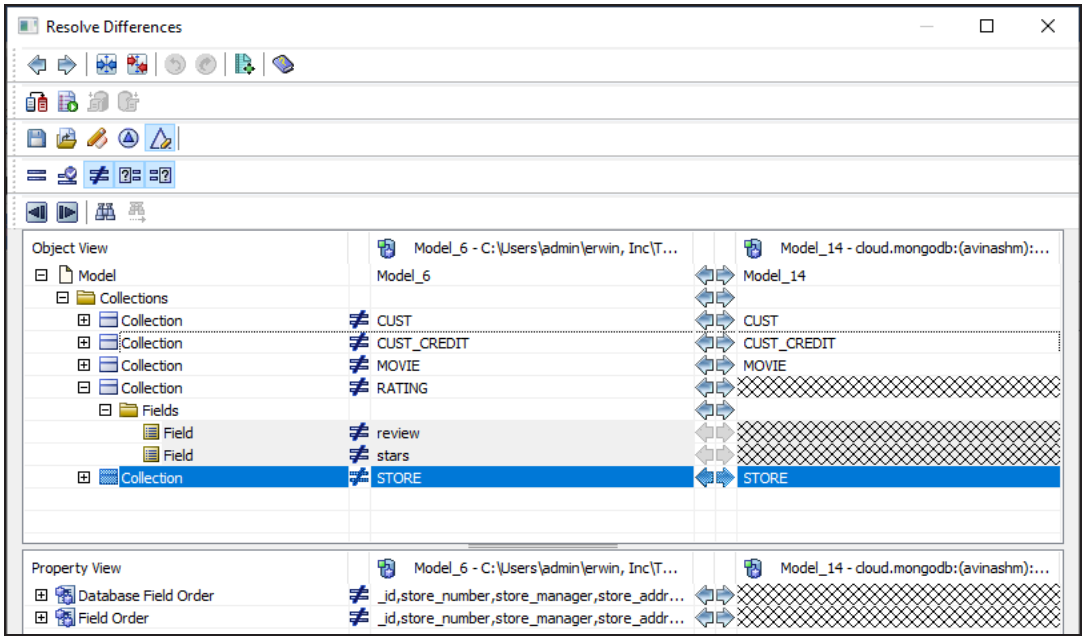


15. Click **Next** and in the Right Object Selection section, select the appropriate options.

For example, the following image shows the default options.



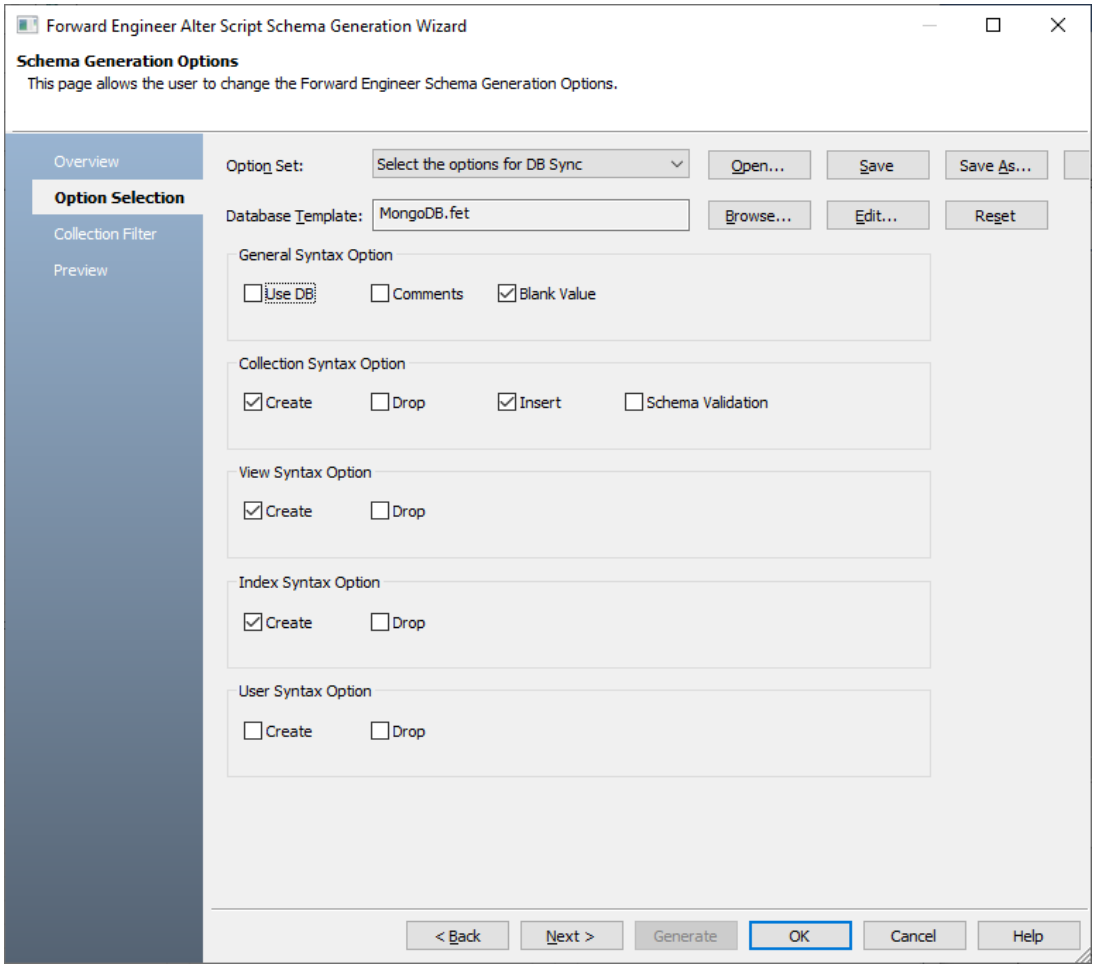
16. Click **Compare**.
    The comparison process runs, and the Resolve Differences dialog box appears. It displays the differences between your model and database.
    For example, the following image shows that the Rating collection is available in your model but not in the database.



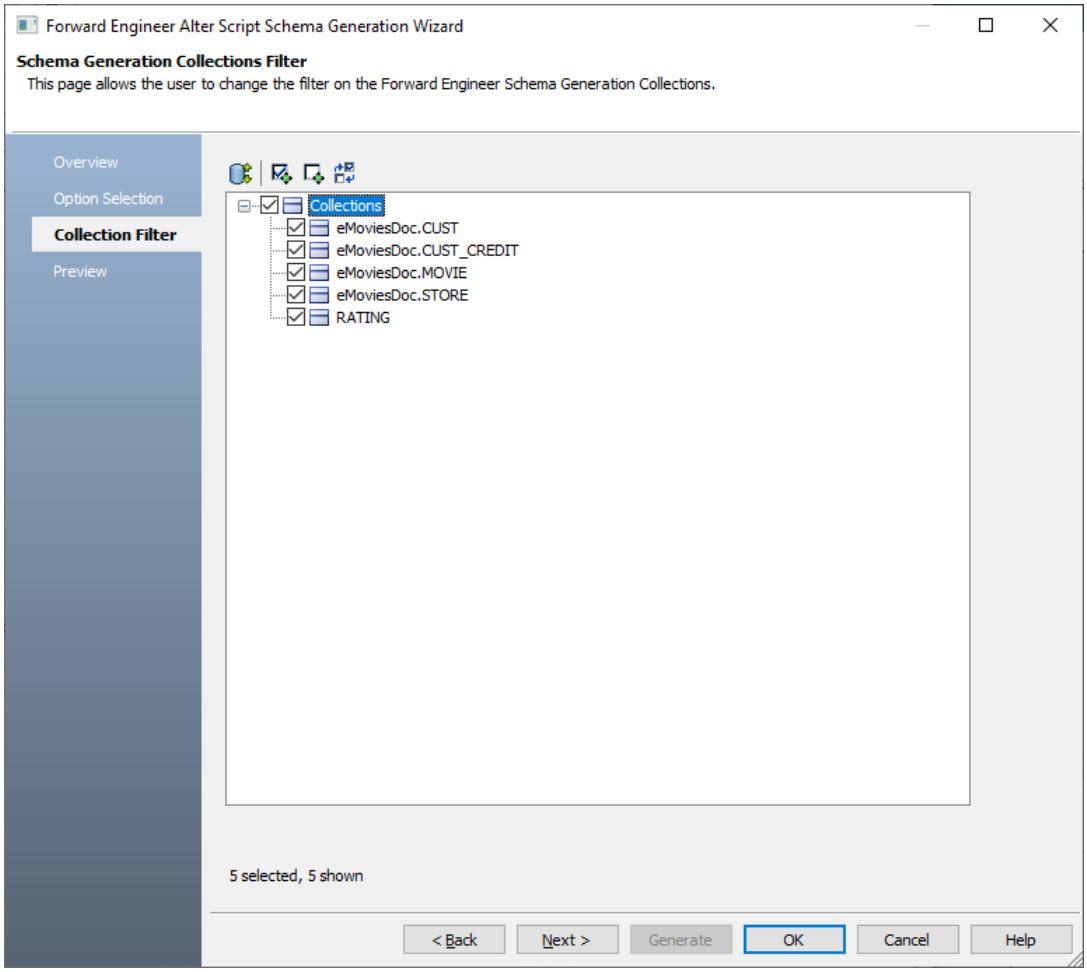
Select the Rating collection and click . This will move the Rating collection to the right model (from the database). Similarly, resolve other differences.

17. As differences were moved to the right model, click .
    This launches the Forward Engineering Alter Script Generation Wizard.

18. Click **Option Selection** and clear all the **Drop** check boxes.



19. Click **Collection Filter** and select or verify the collections to be included on the forward engineering script.



20. Click **Preview** to view and verify the alter script.

21. Click **Generate** and connect to your MongoDB database.
    The forward engineering process starts. The script generates your physical database schema. You can access your database and verify the newly generated schema.

22. Click **OK**. Then click **Finish**.
    This closes the Resolve Differences dialog box and displays the Complete Compare wizard.

23. Click **Close**.

JSON and AVRO Support

erwin Data Modeler (DM) now includes modeling support for JSON and AVRO file formats. The following table lists the supported objects and data types for each format:

File Format	Objects	Data Types
JSON	▪ JSON Objects ▪ Fields ▪ Relationships	▪ Object ▪ Array ▪ Integer ▪ Null ▪ String ▪ Number ▪ Boolean
AVRO	▪ Records ▪ Fields ▪ Relationships	▪ Array ▪ Boolean ▪ Union ▪ Map ▪ int ▪ Double ▪ Object ▪ String ▪ Byte ▪ enum ▪ Fixed ▪ Long

Similar to relational or NoSQL databases, JSON and AVRO as target databases support:

- Reverse engineering models from scripts
- Forward engineering models

Reverse Engineering Models - JSON and AVRO

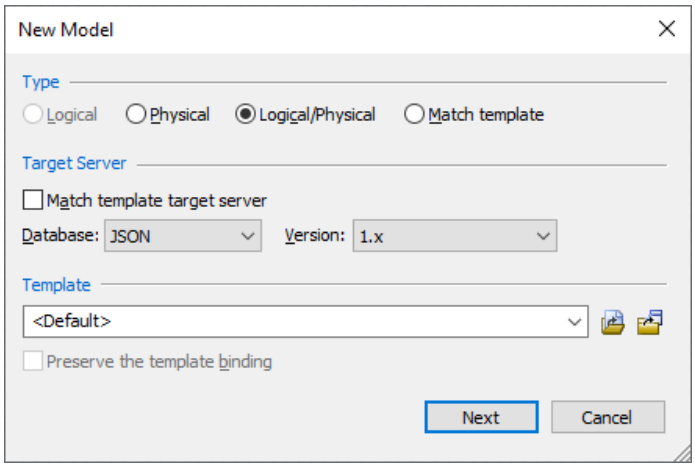
You can create a data model from JSON and AVRO scripts using the Reverse Engineering process.

Note: For reverse engineering German language JSON scripts, ensure the script Encoding is set to Convert to ANSI.

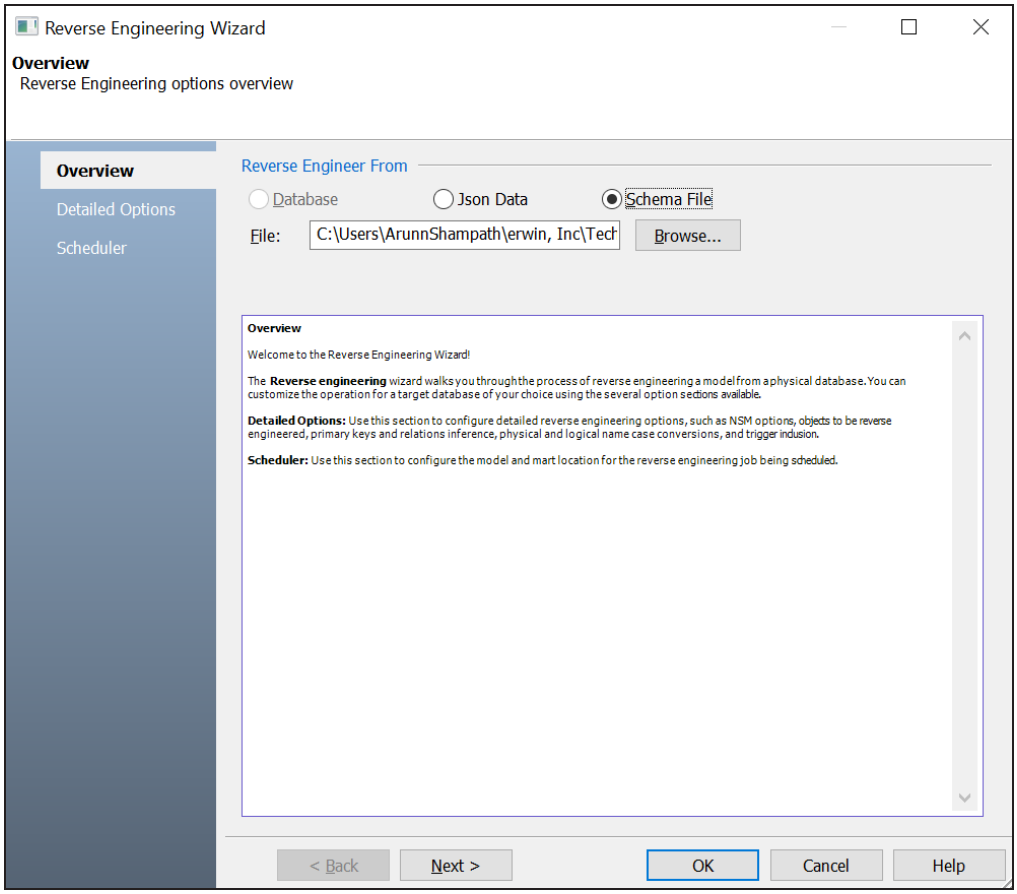
This topic walks you through the steps to reverse engineer a JSON model from a script file. Similarly, you can reverse engineer a model from your AVRO script file.

To reverse engineer a model:

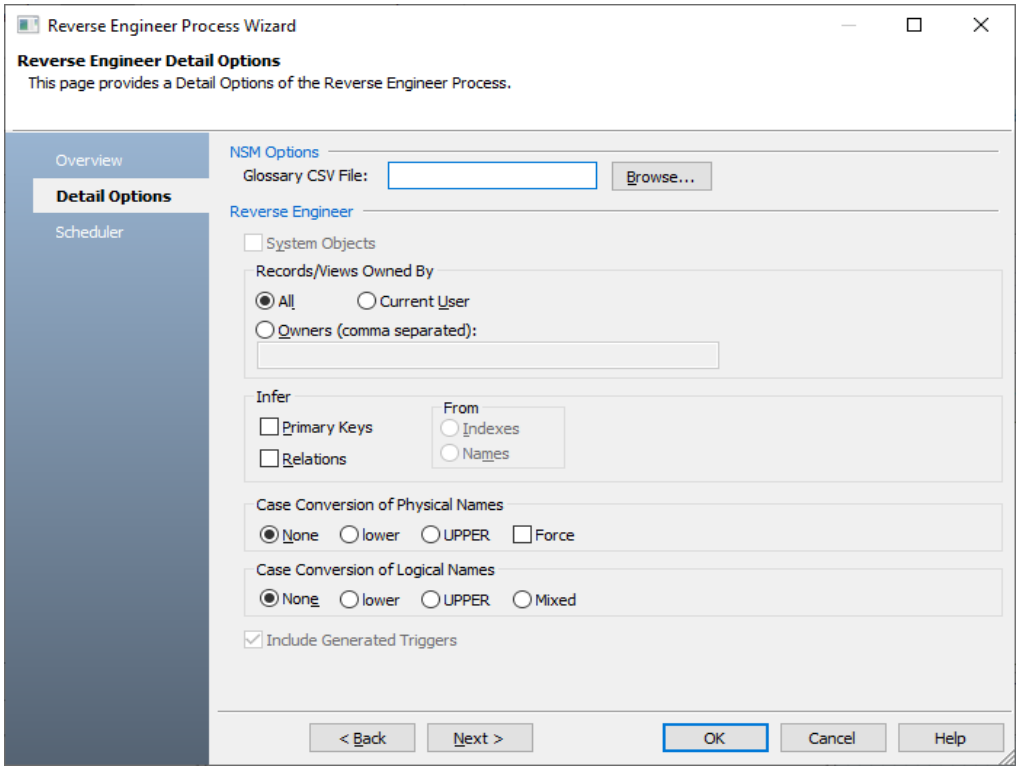
1. In erwin Data Modeler (DM), click **Actions** > **Reverse Engineer**.
   The New Model screen appears.



2. Click **Logical/Physical** and set **Database** to JSON.
3. Click **Next**.
   The Reverse Engineer Process Wizard appears.

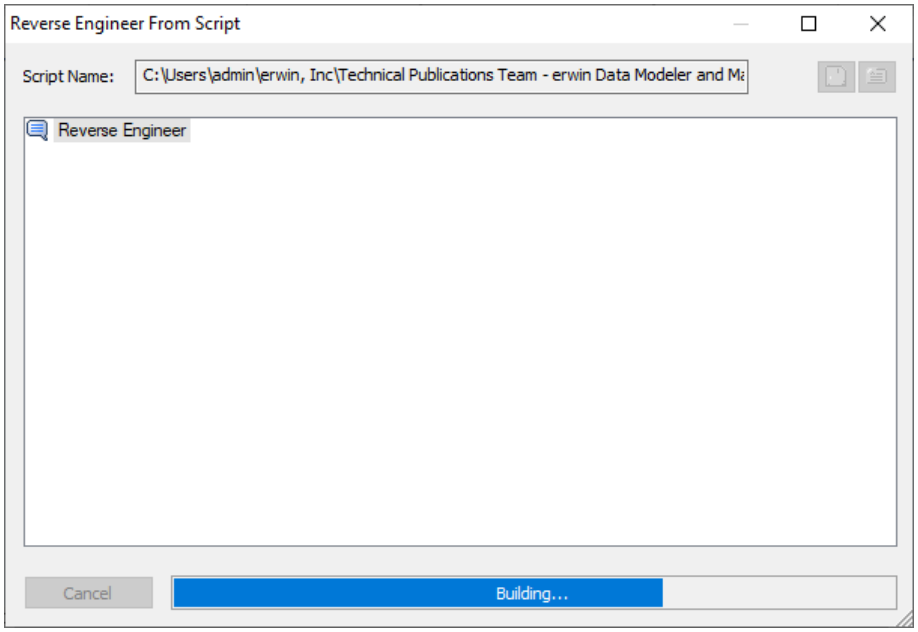


4. Select **Json Data** or **Schema File** format option. Then, click **Browse** and select one or multiple script files.
5. Click **Next**.
   The Detail Options section appears. Set up appropriate options based on your requirement.

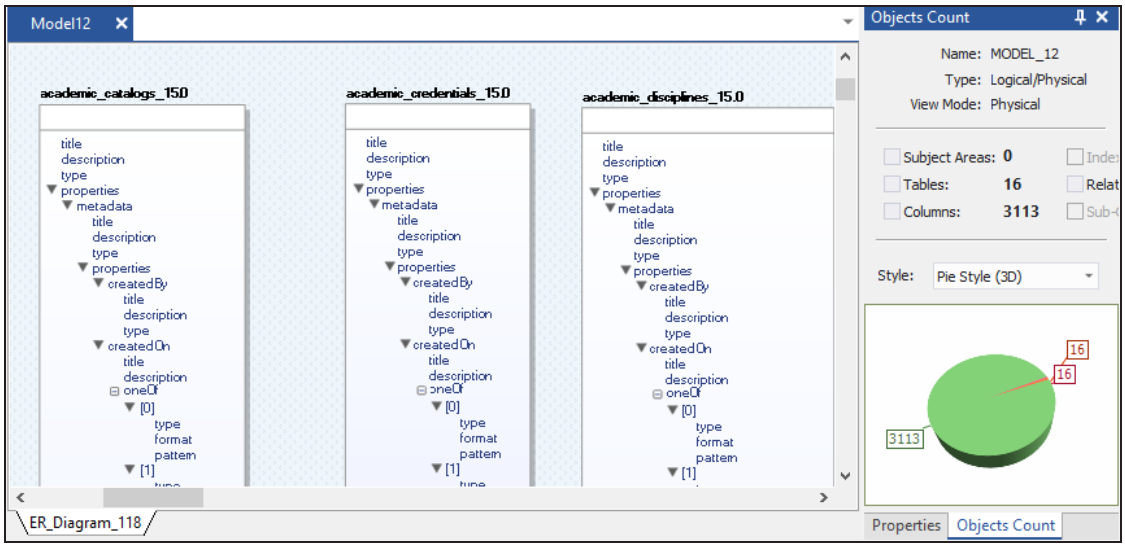

Reverse Engineer Process Wizard
Reverse Engineer Detail Options
This page provides a Detail Options of the Reverse Engineer Process.
Overview
Detail Options
Scheduler
NSM Options
Glossary CSV File:
Browse...
Reverse Engineer
System Objects
Records/Views Owned By
All
Current User
Owners (comma separated):
Infer
Primary Keys
Relations
From
Indexes
Names
Case Conversion of Physical Names
None
lower
UPPER
Force
Case Conversion of Logical Names
None
lower
UPPER
Mixed
Include Generated Triggers
< Back
Next >
OK
Cancel
Help

6. Click **OK**.

The reverse engineering process starts.



Once the process is complete, based on your selections, a schema is generated, and a model is created.



Forward Engineering Models - JSON and AVRO

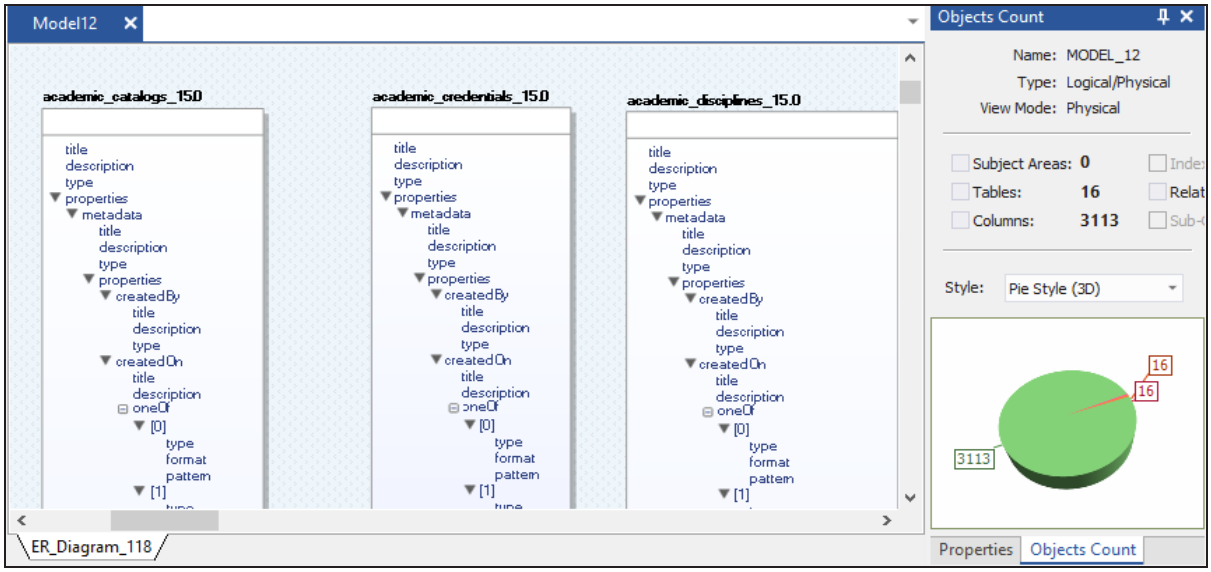
You can generate a physical schema from a physical model using the Forward Engineering process and then, save it in the JSON and AVRO file formats.

This topic walks you through the steps to forward engineer a JSON model. Similarly, you can forward engineer an AVRO model.

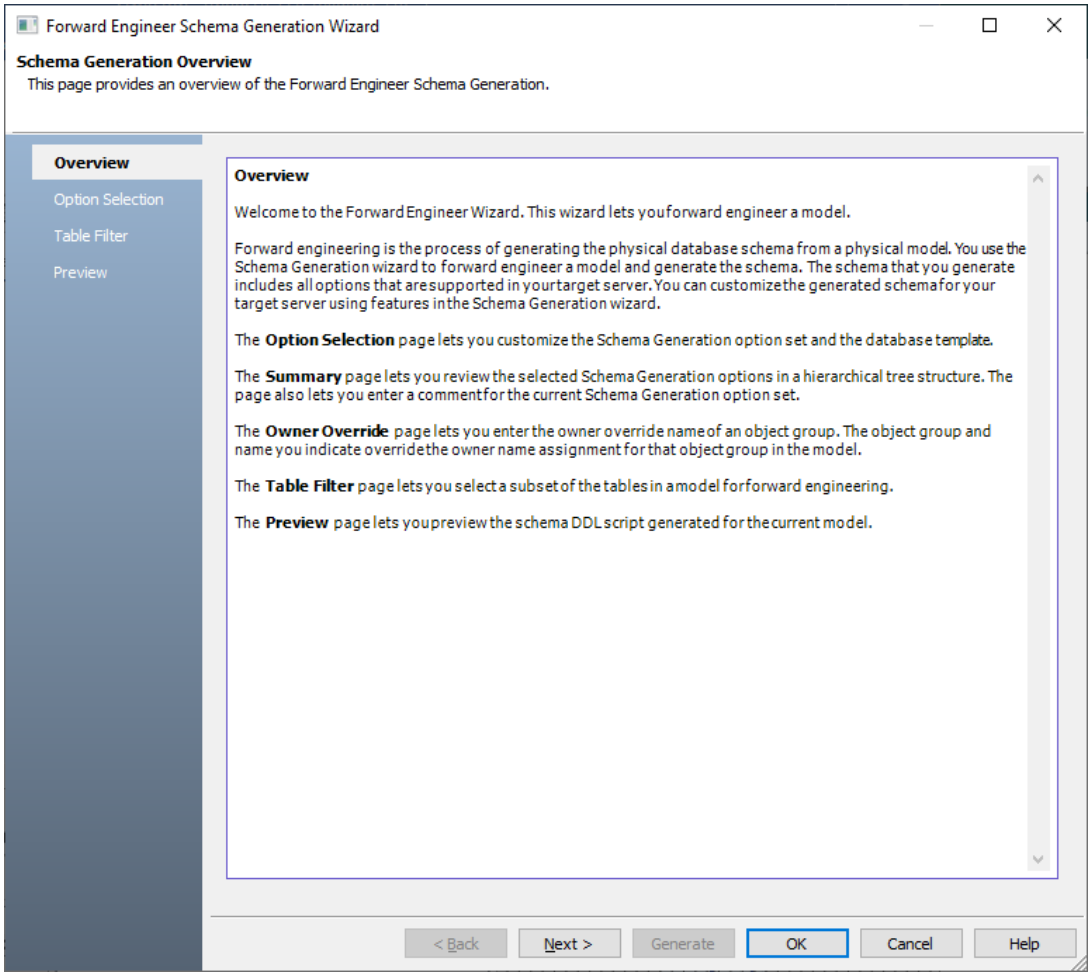
To forward engineer a model:

1. Open your JSON model in erwin Data Modeler (DM).
   Note: Ensure that you are in the Physical mode.

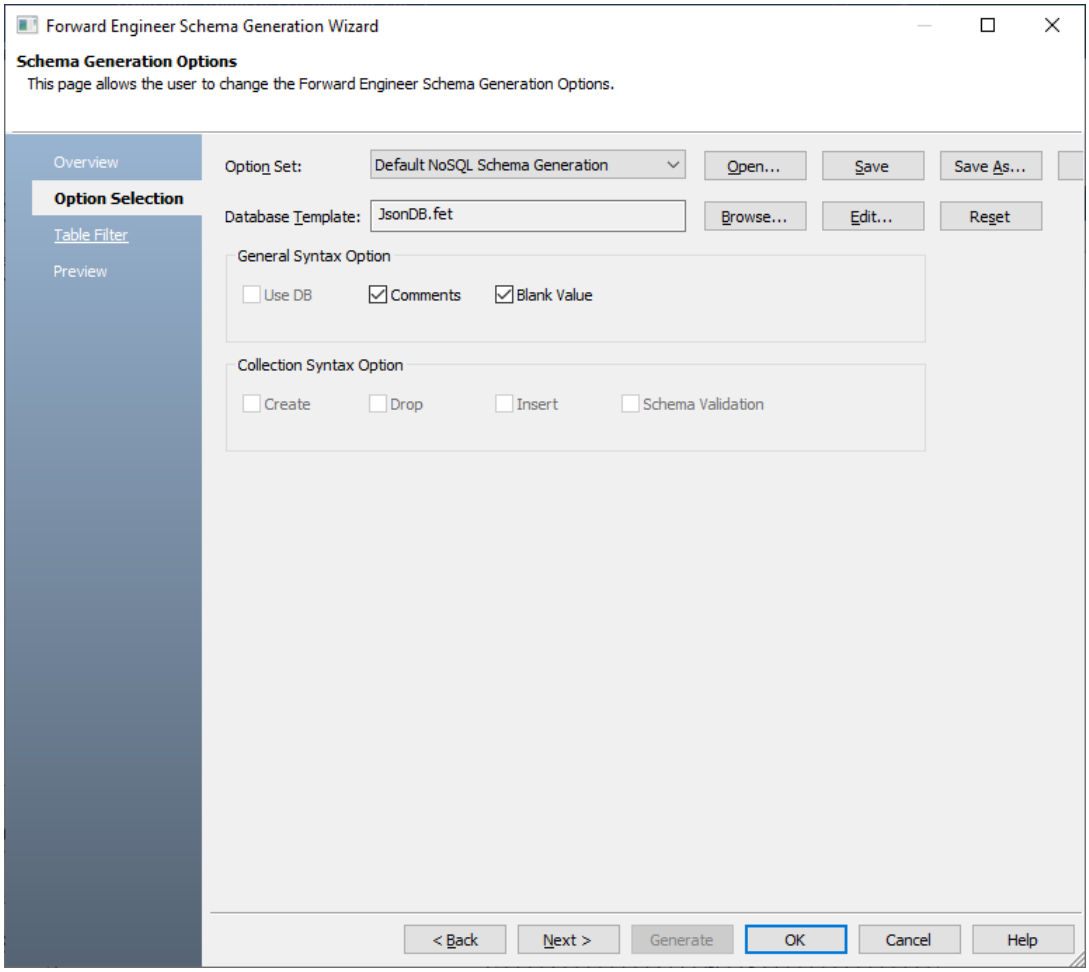
   For example, the following image uses a JSON model with 16 tables.



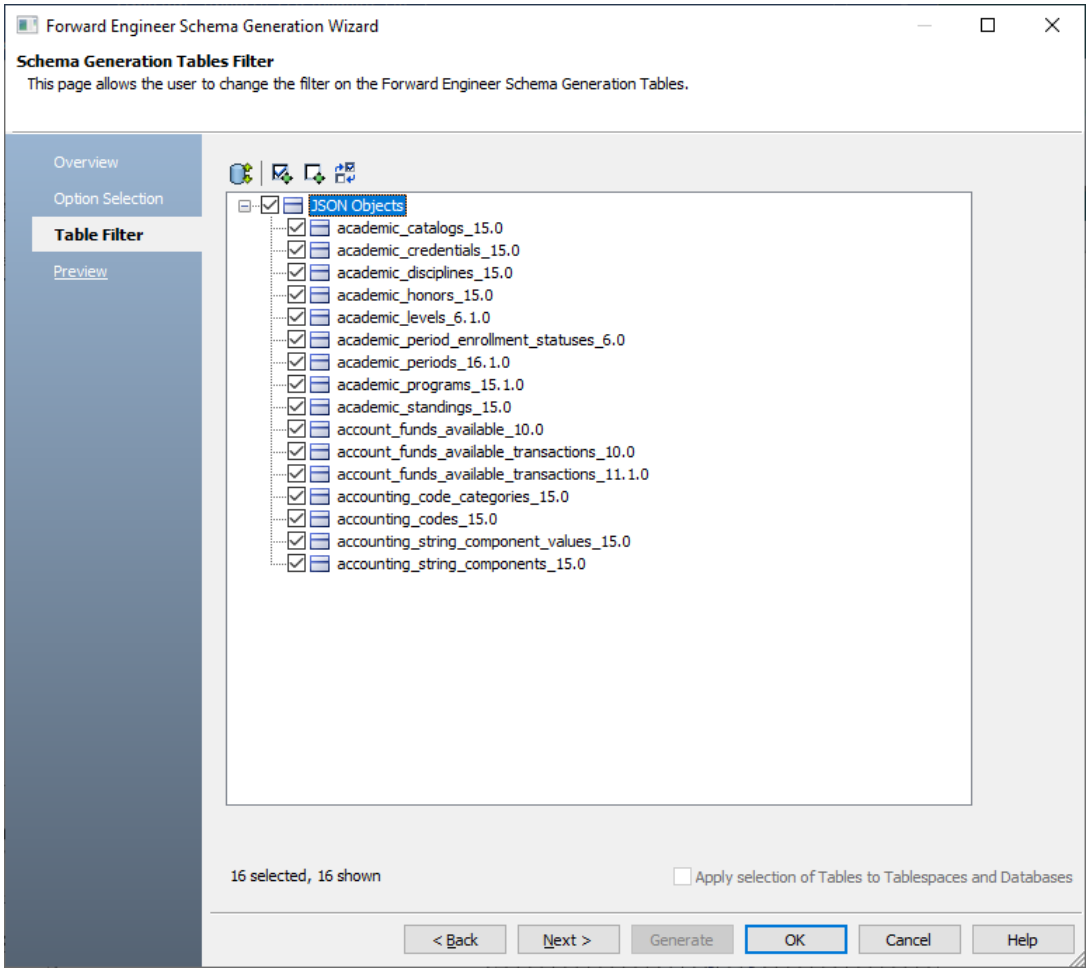
2. Click **Actions** > **Schema**.
   The Forward Engineer Schema Generation Wizard appears.



3. Click **Option Selection**.
   The Option Selection section displays the default option set. Select appropriate syntax options.

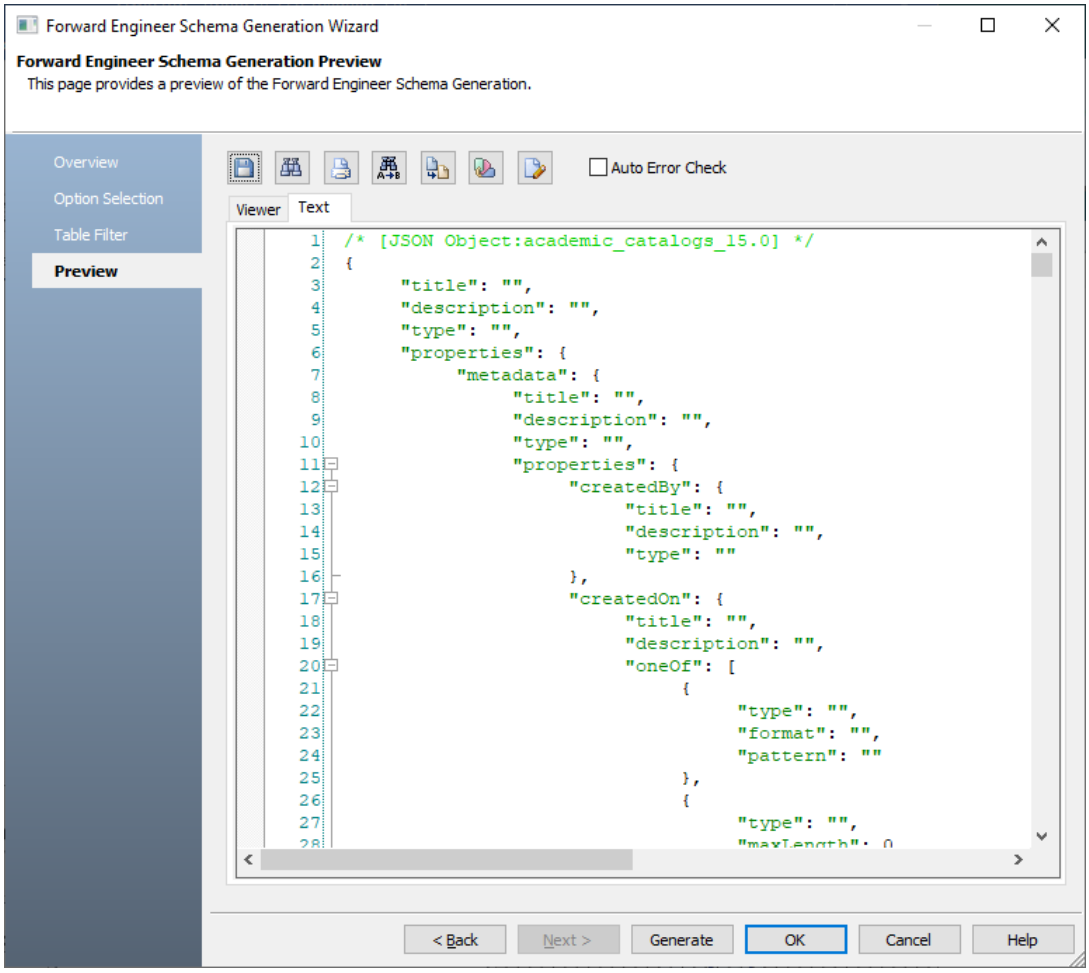


4. Click **Next**.
   The Table Filter section appears. It displays a list of tables (JSON objects) available in your model.



5. Select the tables (JSON objects) that you want to forward engineer.

6. Click **Preview** to view the schema script.



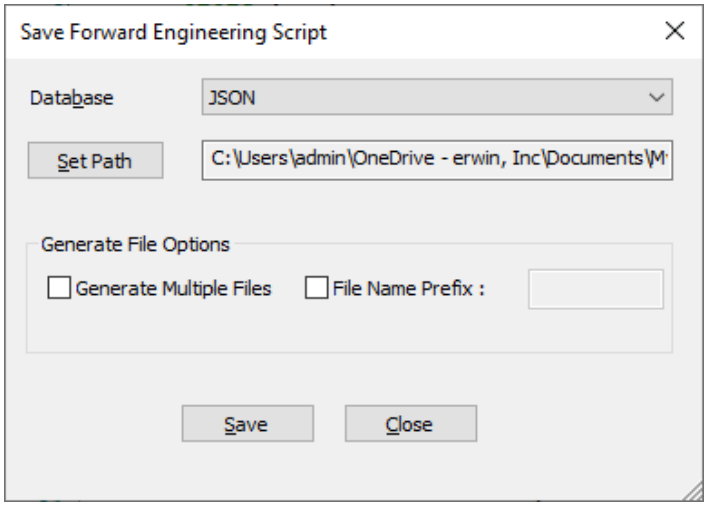
Use the following options:

- **Auto Error Check**: Select this option to enable auto error check by the forward engineering wizard.
- **Error Check** (): Use this option to run an error check. Based on the results, you can correct the generated script.
- **Text Options** (): Use this option to configure the preview text editor's look and feel, such as window, font, syntax color settings. For more information,

refer to the Forward Engineering Wizard - Preview Editor topic.

- **Save** (): Use this option to save the generated script.

7. Click **Generate**.
   The following screen appears.



8. Use the following options:
   - **Set Path**: Use this option to set the location to save the script file.
   - **Generate Multiple Files**: By default, a single script file is created. Select this option to save the script into multiple files by objects.
   - **File Name Prefix**: Select this option to add a script file name. Enter a file name. If this option is not selected, the script file is saved with a default name (Erwin_FE_Script.json).
9. Click **Save**.
   Your script file is saved at the configured location. You can open it in any text editor and verify.

Oracle Support Summary

erwin Data Modeler (DM) now supports Oracle 12c R2, 18c, 19c, and 21c as target databases. This implementation supports the following objects:

- Cluster
- Column
- Comment
- Context
- Database
- Database Link
- Directory
- Disk Group
- Function
- Index Editor for Clusters
- Index Editor for Materialized Views
- Index Editor for Tables
- Library
- Materialized Views
- Materialized View Log
- Package
- Package Body
- Pre and Post Scripts
- Rollback Segment
- Sequence
- Stored Procedure
- Synonym

- Table
- Tablespace
- Tablespace Group
- Trigger
- Views

The following table lists the supported data types:

Numeric	String Literals	Date and Time	Other
• BINARY_ DOUBLE • BINARY_ FLOAT • DEC • DEC() • DEC(,) • DECIMAL • DECIMAL() • DECIMAL(,) • DOUBLE PRECISION • REAL • FLOAT • FLOAT() • INT • INTEGER • NUMBER	• CHAR • CHAR() • CHARVARYING() • CHARACTER • CHARACTER() • CHARACTERVARYING () • CLOB • DBURITYPE • URITYPE • HTTPURITYPE • JSON • JSON() • LONG • NATIONAL CHAR • NATIONAL CHAR VARYING() • NATIONAL CHAR()	• DATE • INTERVAL DAY TO SECOND • INTERVAL YEAR TO MONTH • TIMESTAMP • TIMESTAMP WITH LOCAL TIMEZONE • TIMESTAMP WITH TIMEZONE • TIMESTAMP () • TIMESTAMP () WITH LOCAL TIMEZONE • TIMESTAMP () WITH	• JSON* • ANYDATA • ANYDATASET • ANYTYPE • BFILE • BLOB • LONGRAW • ORDAUDIO • ORDDICOM • ORDDOC • ORDIMAGE • ORDVIDEO • RAW() • SDO_GEOMETRY • SDO_GEORASTER • SI_AVERAGECOLOR • SI_COLOR

• NUMBER() • NUMBER(,) • NUMERIC • NUMERIC() • NUMERIC (,) • ROWID • SMALLINT	• NATIONAL CHARACTER • NATIONAL CHARACTER VARYING() • NATIONAL CHARACTER() • NCHAR • NCHAR VARYING() • NCHAR() • NCLOB • NVARCHAR2() • UROWID • UROWID() • VARCHAR() • VARCHAR2() • XDBURITYPE • XMLTYPE	TIMEZONE	• SI_ COLORHISTOGRAM • SI_FEATURELIST • SI_ POSITIONALCOLOR • SI_STILLIMAGE • SI_TEXTURE

*This datatype is supported only for Oracle 21c.

Microsoft SQL Server Support

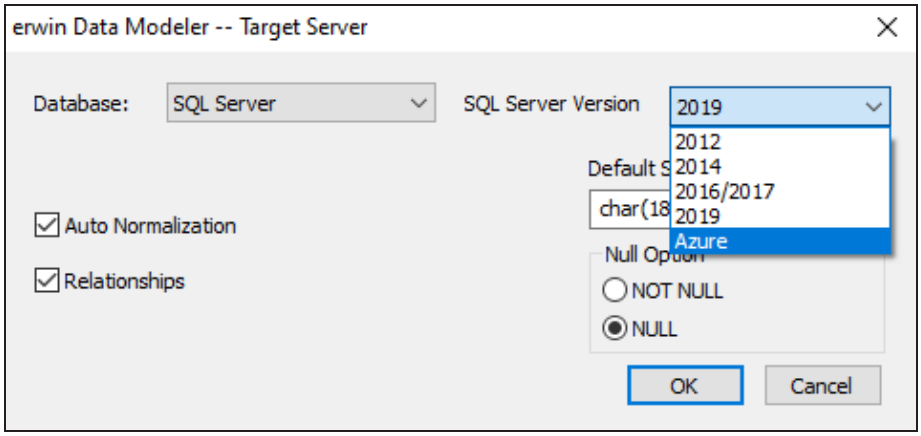
Support for Microsoft SQL Server 2019 as a target database has been enhanced to implement the following objects:

- External Library
- External Language
- External Data Source
- External File Format
- External Table
- Statistics

For detailed information on supported objects and data types, refer SQL Server support summary.

Microsoft Azure SQL Server Support

Microsoft Azure SQL support in erwin DM has been revamped. It is now supported on top of Microsoft SQL Server to leverage common functionality. On the New Model and Target Database dialog boxes, you can find Azure under SQL Server Version drop-down list.



The following table lists the supported objects:

Supported Objects	
• Always Encrypted Keys • Application Roles • Assemblies* • Asymmetric Keys • Certificates • Credentials* • Database Roles • Databases • Database Triggers • External Data	• Resource Pools* • Schemas • Sequences • Server Audits* • Server Audit Specification* • Statistics • Stored Procedures • Symmetric Keys • Spatial Indexes (Table) • Synonyms

Source • External File Format* • External Library* • External Table • Full-Text Catalogs • Full-Text Indexes (Table) • Full-Text Stoplists • Functions • Logins • Partition Functions • Partition Schemes	• Tables • ColumnStore Indexes (Table) • XML Indexes (Table) • Indexes (Table) • Table Triggers • Triggers • User Ids • Views • View Indexes • View Triggers • XML Schema Collections

* These objects are supported only for Azure SQL Managed Instance.

The following table lists the supported data types:

Exact Numerics	Approximate Numerics	Date and Time	Character Strings	Unicode Character Strings	Binary Strings	Geo Types	Others
• bigint • numeric • bit • small-	• float • real	• date • datetimeoffset • datetime2	• char • varchar • text-	• nchar • nvarchar • ntext	• binary • varbinary • image	• rowversion • hierarchyid • uniqueidentifier	• CHAR VARYING • CHARAC-

lint • decimal • smallmoney • int • tinyint • money		• smalldatetime • datetime • time	xt			• sql_variant • xml • geometry • geography	TER • CHARACTER VARYING • NATIONAL CHAR VARYING • NATIONAL CHAR • NATIONAL CHARACTER • NATIONAL TEXT

MySQL Support

erwin Data Modeler (DM) now supports MySQL 8.x as a target database. This implementation supports the following objects:

- Database
- Event
- Function
- Function_UDF
- Logfile Group
- Server
- Spatial Ref System
- Stored Procedure
- Table
  - Index
  - Table Column
- Tablespace
- Trigger
- User ID
- Validation Rule
- View
  - View Column

The following table lists the supported data types:

Numeric	String Literals	Date and Time	Other
• TINYINT • SMALLINT • MEDIUMINT • INT, INTEGER	• CHAR • VARCHAR • BINARY • CHAR BYTE	• DATE • TIME • DATETIME • TIMESTAMP • YEAR	• Geometry Type • POINT • LINESTRING • POLYGON • MULTIPOINT

• BIGINT • DECIMAL, DEC, NUMERIC, FIXED • FLOAT • DOUBLE, DOUBLE PRECISION, REAL • BIT	• VARBINARY • TINYBLOB • BLOB • BLOB and TEXT Data Types • MEDIUMBLOB • LONGBLOB • TINYTEXT • TEXT • MEDIUMTEXT • LONGTEXT • JSON Data Type • ENUM • Set Data Type		• MULTILINESTRING • MULTIPOLYGON • GEOMETRYCOLLECTION • GEOMETRY

Note:

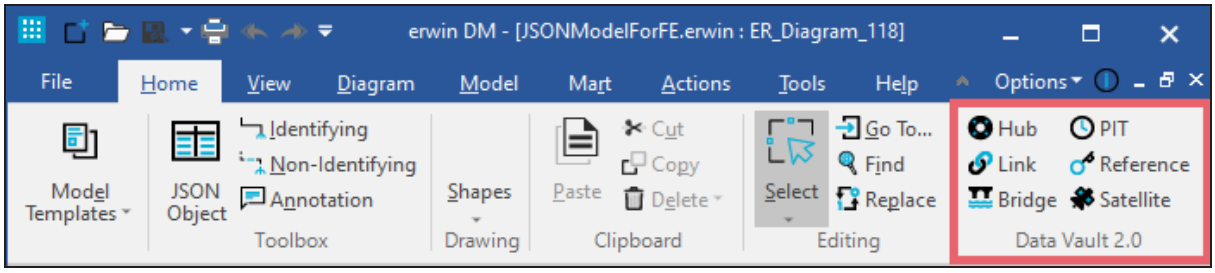
- Refer to MySQL database documentation for detailed information on specific MySQL objects and properties. erwin® Data Modeler documentation for the property editors provides brief descriptions of the controls on each dialog box and tab, which you can use as a point of reference while working with database design features.
- As a best practice, use the MySQL ANSI ODBC driver for Reverse Engineering from Database (REDB) while using erwin® Data Modeler

Data Vault 2.0 Support

erwin Data Modeler (DM) now Data Vault 2.0 as a modeling technique across all target databases. This implementation supports the following Data Vault 2.0 components by default through API:

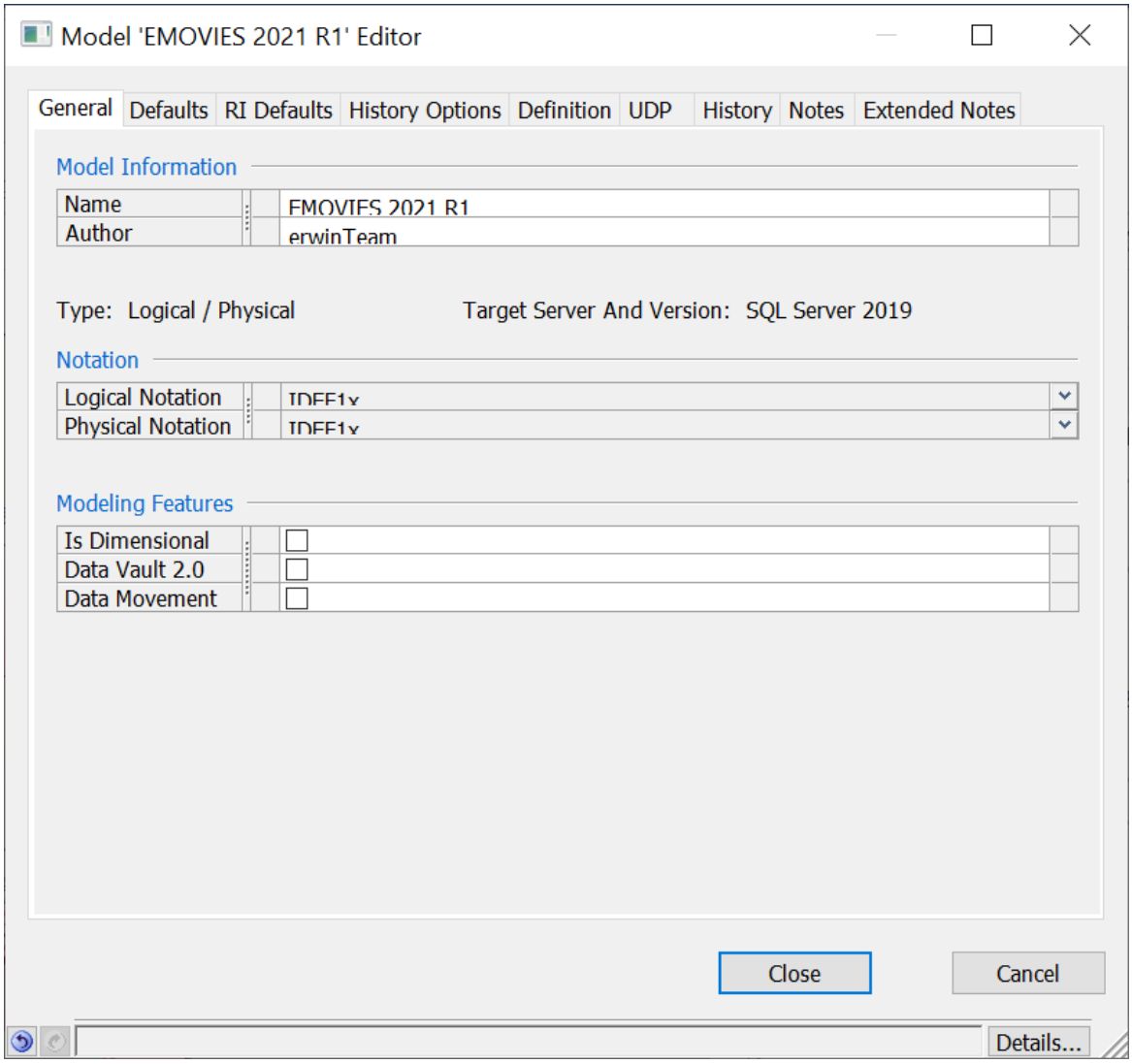
- Hub
- Link
- Satellite
- Reference
- PIT
- Bridge

These components are available on Home tab of the ribbon.



To enable Data Vault 2.0 on your model, follow these steps:

1. Right-click the model and click **Properties**.
2. On the **Model Editor** > **General** tab, select the **Data Vault 2.0** check box.



Once enabled, Data Vault 2.0 components are available via the Model Explorer. You can now convert your model to a Data Vault model.

You can also create custom components and apply them to tables. However, these custom components do not appear on the ribbon.

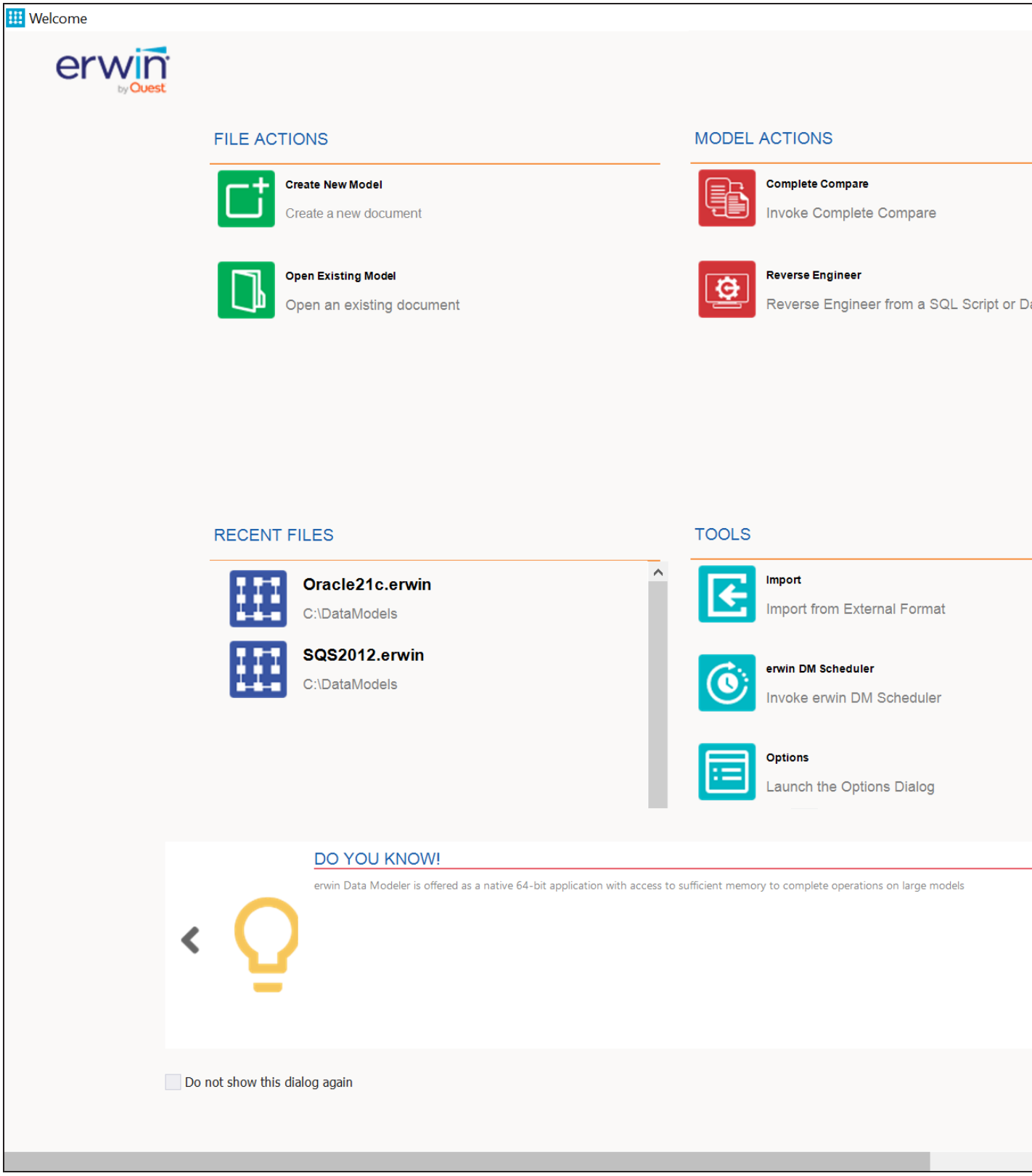
Productivity and UI Enhancements

Several additions and enhancements have been implemented to improve erwin Data Modeler's (DM) productivity and usage experience. These enhancements are:

- Welcome Page
- Objects Count Pane
- Properties Pane
- Object Browser
- Normalization and Denormalization
- Reverse Engineering and Forward Engineering Wizard Redesign
- Improved Speed Mode

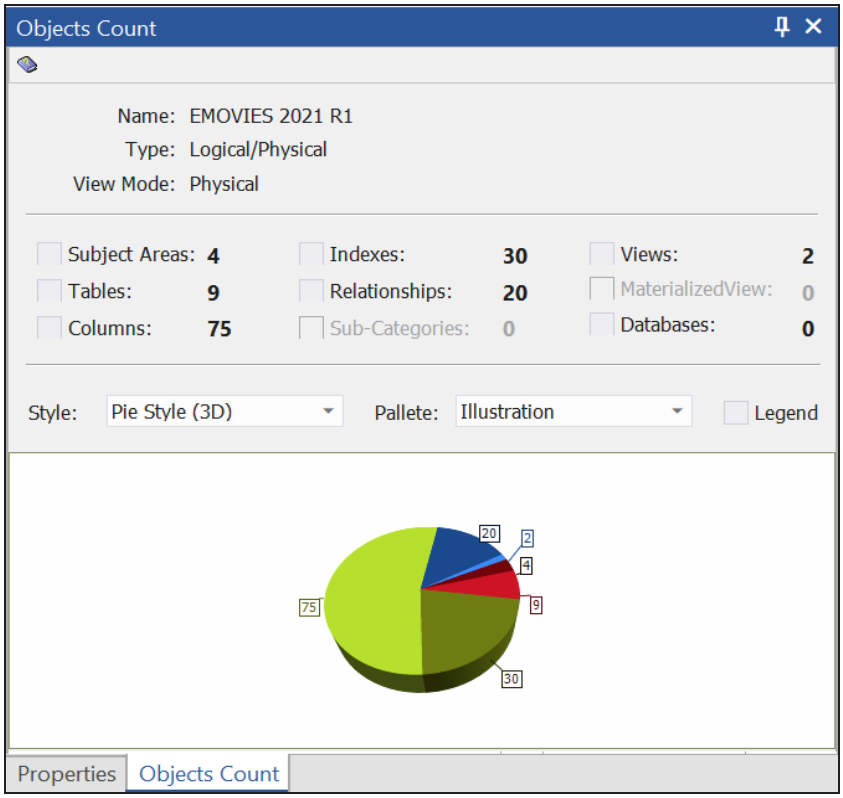
Welcome Page

The Welcome page is a starter page that helps new users to get started with erwin DM. It appears when you launch erwin DM and is also accessible via **Help** > **Welcome**. It contains shortcuts to key actions that are performed frequently, such as opening models or creating new ones, running reverse engineering or complete compare wizards, and connecting to the Mart. Apart from these, the Welcome page provides access to recently used files, erwin DM Tools, Technical Support, and Help links.


Welcome
erwin by Quest
FILE ACTIONS
Create New Model
Create a new document
Open Existing Model
Open an existing document
MODEL ACTIONS
Complete Compare
Invoke Complete Compare
Reverse Engineer
Reverse Engineer from a SQL Script or Da
RECENT FILES
Oracle21c.erwin
C:\DataModels
SQS2012.erwin
C:\DataModels
TOOLS
Import
Import from External Format
erwin DM Scheduler
Invoke erwin DM Scheduler
Options
Launch the Options Dialog
DO YOU KNOW!
erwin Data Modeler is offered as a native 64-bit application with access to sufficient memory to complete operations on large models
Do not show this dialog again

Objects Count Pane

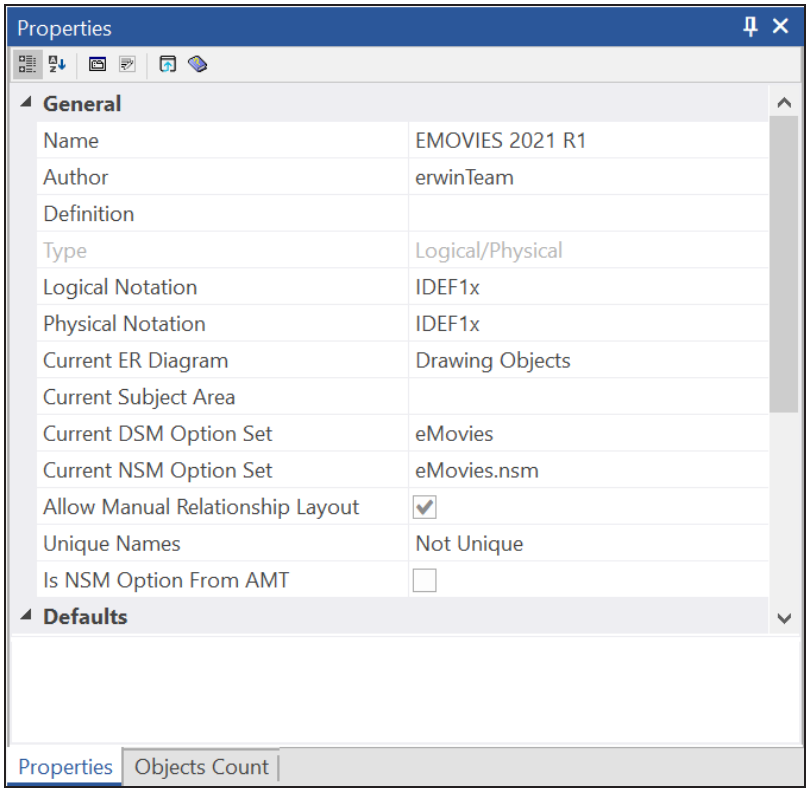
As data models become complex and large, it becomes necessary to get a snapshot of the objects within models. To facilitate this, erwin DM now includes an Objects Count pane. This pane displays information about a selected model and a count of all the objects present in it. Also, it displays a snapshot of this information in the pictorial format, which you can customize using the Style and Pallete options. By default, this pane opens on the right-side of the application. You can also access it via **View** > **Panes** > **Objects Count Pane**.



Properties Pane

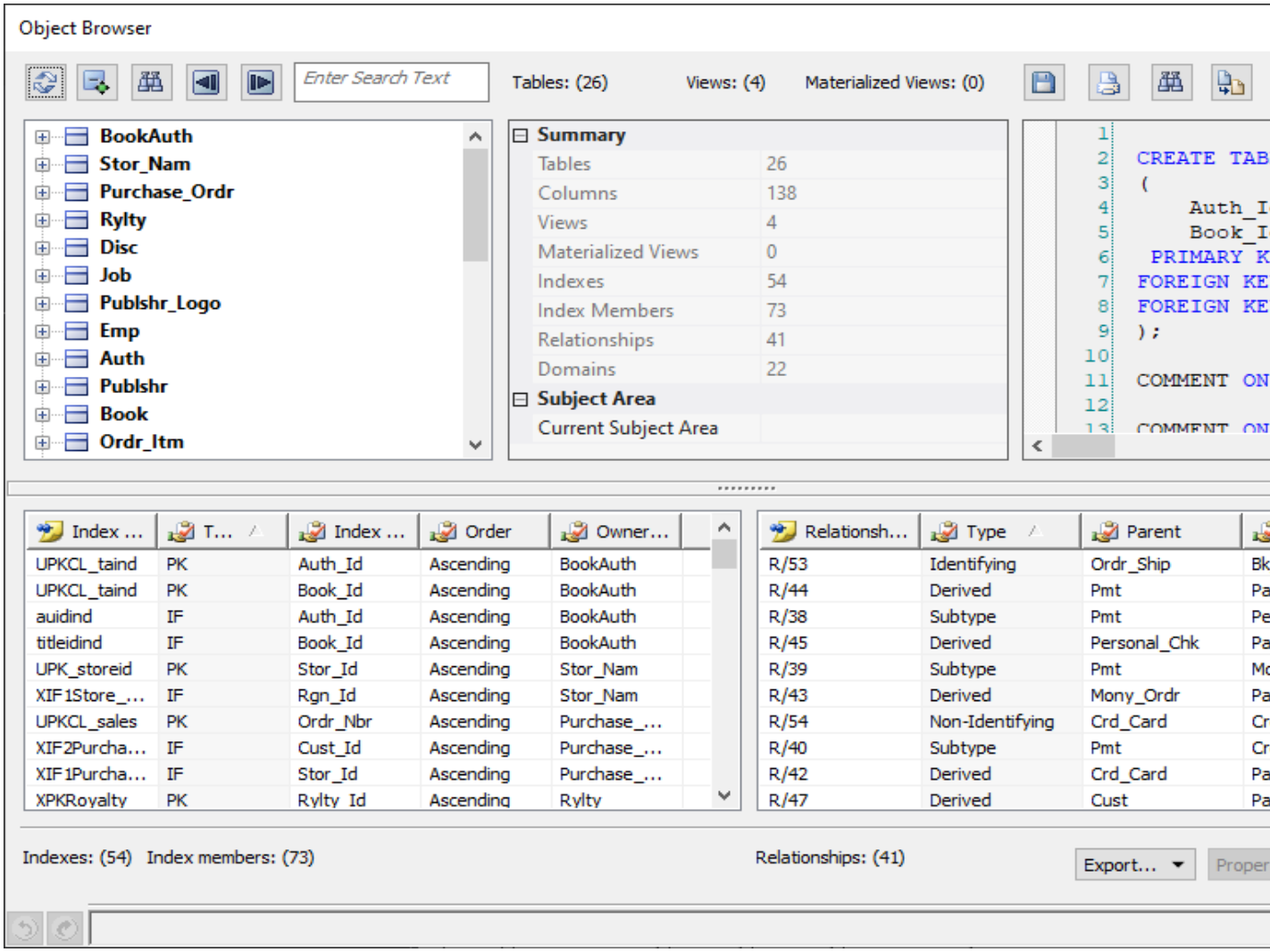
While working on data models, accessing the property editors to view or edit the model and its object's properties can get tedious and slow you down. To address this, erwin DM now includes a Properties pane. This pane enables you to view and edit the selected object's properties along with the model diagram, side-by-side. By default, this pane opens on the

right-side of the application. You can also access it via **View** > **Panes** > **Properties Pane** on the ribbon.



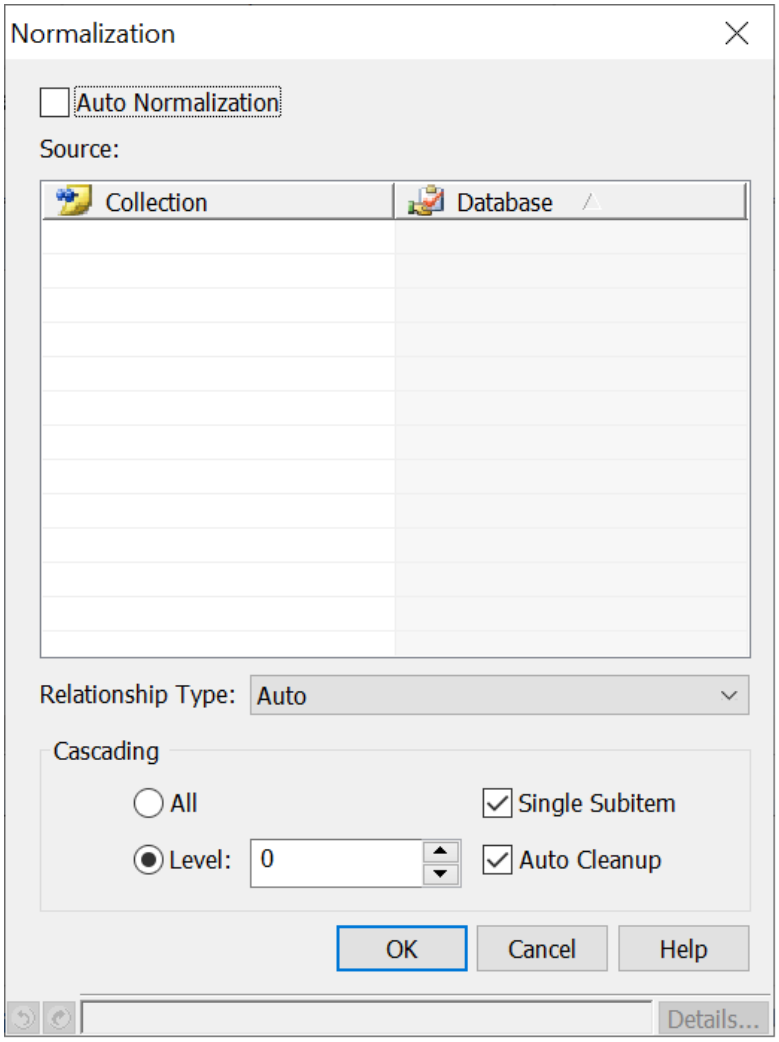
Object Browser

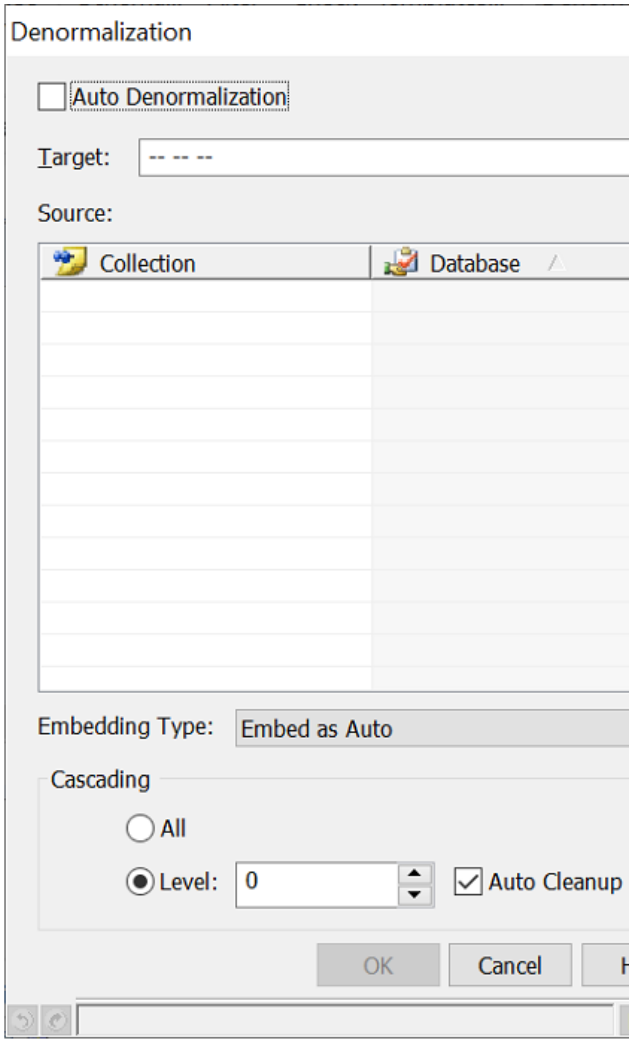
The Object Browser is a one-stop location where you can view tables, views, materialized views, indexes, relationships, and the complete model's or specific table's DDL. You can export this information as a report in CSV, HTML, or PDF formats. To access the Object Browser, on the **Properties** pane, click or on the ribbon, click **Tools** > **Object Browser**. For more information, refer to the Object Browser topic.



Normalization and Denormalization

The Normalization and Denormalization features enable you to define relationships in a NoSQL model. Normalization splits the fields in a collection into multiple collections based on the selected relationship type. Whereas Denormalization embeds multiple collections into a single collection based on the selected embedding type. To access these features, on the ribbon, click **Actions**. Then, click **Normalization** or **Denormalization**. For more information, refer to the Defining Relationships Using Embedding Method topic.





Reverse Engineering and Forward Engineering Wizard Redesign

The Reverse Engineering and Forward Engineering wizards have been redesigned for better arrangement of properties and ease of use. For more information, refer to the Reverse Engineering and Forward Engineering topics.

Reverse Engineer Process Wizard

Reverse Engineer Process Overview

This page provides an overview of the Reverse Engineer Process.

- **Overview**
- Connection
- Database
- Collection
- Option Set
- Detail Options
- Scheduler

Reverse Engineer From

(•) Database ( ) Script File

File: [ ] Browse...

Overview

Welcome to the Reverse Engineer Wizard. This wizard lets you reverse engineer from the physical database.

Reverse engineering is the process of generating the physical database schema from a physical model. You can customize the Reverse Engineer for your target server using features in the Detail options.

The **Option Set** page lets you customize the Reverse Engineer option set.

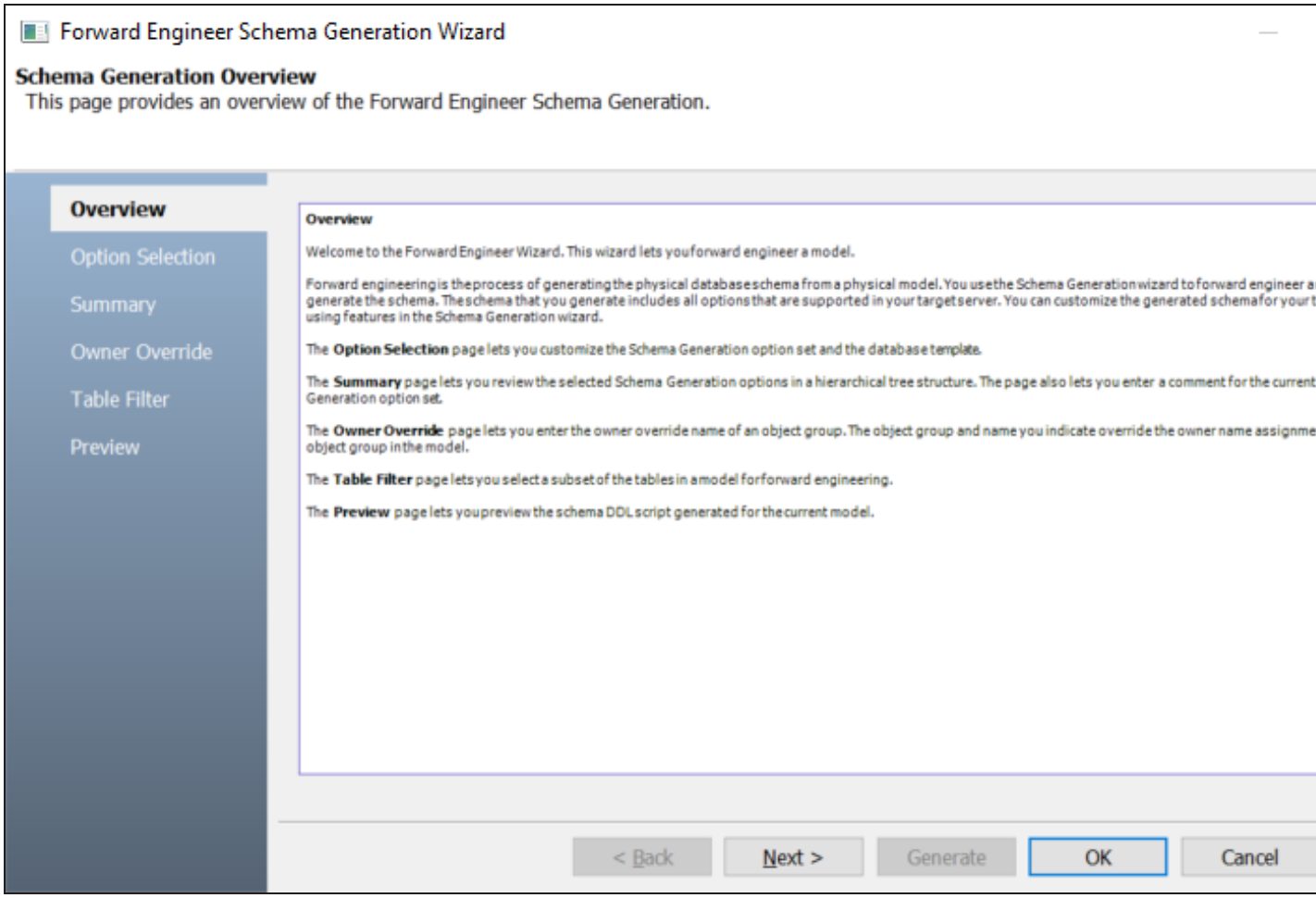
The **Scheduler** page lets you customize the Reverse Engineer Scheduler options. The page also lets you use Complete Compare and Mart for the target location.

The **Owner Override** page lets you enter the owner override name of an object group. The object group and name you indicate override the owner name assignment for that object group in the model.

The **Collection Filter** page lets you select a subset of the tables in a model for Reverse engineering.

The **Preview** page lets you preview the schema DDL script generated for the current model.

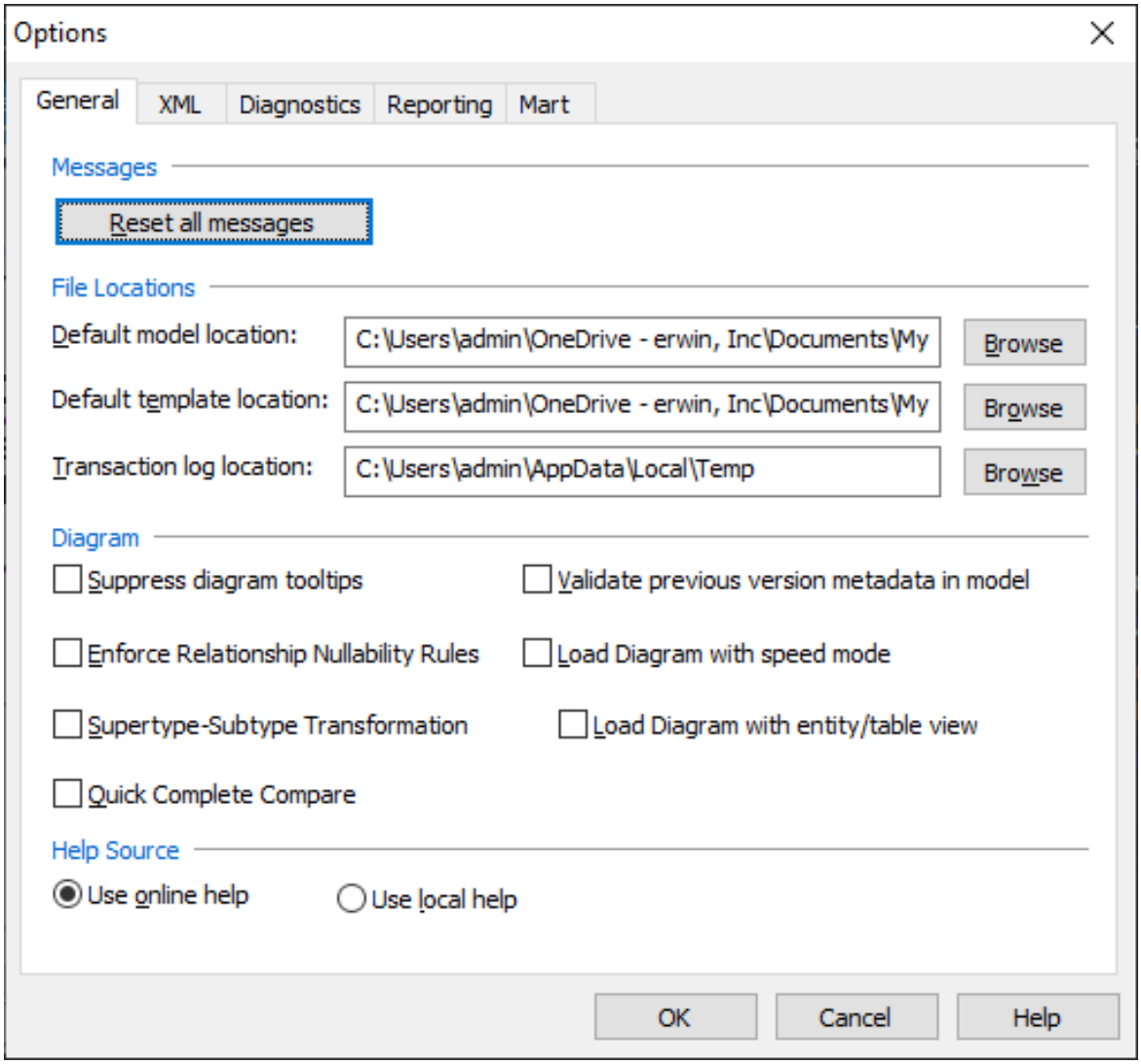
< Back | Next > | OK | Cancel | Help



Improved Speed Mode

The Load Diagram with speed mode option now provides another option, Load Diagram with entity/table view. This option improves the model load performance in case of large models significantly. It does so by rendering the model in the simplest way possible, with only the entities or tables, and their relationships. Also, by default, it disables the PK-FK high-light feature (**Display Diagram Highlight** option on Model Editor).

This option is available on **Tools** > **Options** dialog box.


Options
General
XML
Diagnostics
Reporting
Mart
Messages
Reset all messages
File Locations
Default model location:
C:\Users\admin\OneDrive - erwin, Inc\Documents\My
Browse
Default template location:
C:\Users\admin\OneDrive - erwin, Inc\Documents\My
Browse
Transaction log location:
C:\Users\admin\AppData\Local\Temp
Browse
Diagram
Suppress diagram tooltips
Validate previous version metadata in model
Enforce Relationship Nullability Rules
Load Diagram with speed mode
Supertype-Subtype Transformation
Load Diagram with entity/table view
Quick Complete Compare
Help Source
Use online help
Use local help
OK
Cancel
Help

JDBC Support

erwin DM now includes JDBC support for the following databases:

- Oracle
- SQL Server
- Azure SQL
- Cassandra
- Couchbase
- MongoDB
- MySQL
- MariaDB

Along with this, the JDBC support for Snowflake has been updated to support new features.

The JDBC database connection parameters for the above databases are as follows:

Oracle

Instance

Specifies the JDBC instance to which you want to connect.

For a cloud-based connected, the instance name is as follows:

TNS_ADMIN=<Path of unzipped cloud wallet file>

For example, *TNS_ADMIN=C:\\Users\\MyUser\\Wallet_DBTEST*

Note: Ensure that you have downloaded, saved, and unzipped the cloud wallet file.

Connection String

Specifies the connection string based on your JDBC instance in the following format:

jdbc:oracle:thin:@//<servername>:1521/

For example, *JDBC:ORACLE:thin:@//localhost:1521/*

For a cloud instance, the connection string is as follows:

jdbc:oracle:thin:@<dbname_priority>?

For example, *jdbc:oracle:thin:@dbtest_medium?*

SQL Server

Connection Type

Specifies the type of connection you want to use. Select *Use Native Connection* to connect using the API provided by the SQL Server Native client software. Select *Use ODBC Data Source* to connect using the ODBC data source that you have defined. Select *Use JDBC Connection* to connect using JDBC.

Instance

Specifies the JDBC instance to which you want to connect.

Database

Specifies the name of the database that you want to connect to.

Connection String

Specifies the connection string based on your JDBC instance and SQL Server database name in the following format:

jdbc:sqlserver://<servername>:1433=<SqlDBname>

For example, *jdbc:sqlserver://localhost:1433*

SQL Azure

Connection Type

Specifies the type of connection you want to use. Select *Use Native Connection* to connect using the API provided by the SQL Server Native client software. Select *Use ODBC Data* to connect using the ODBC data source that you have defined. Select *Use JDBC Connection* to connect using JDBC.

Instance

Specifies the JDBC instance to which you want to connect.

Database

Specifies the name of the database that you want to connect to.

Connection String

Specifies the connection string based on your JDBC instance and SQL Server database name in the following format:

jdbc:sqlserver://<servername>:<port>
For example, *jdbc:sqlserver://localhost:1433*

Connect to Managed Instance

Specifies whether the connection should be to an Azure SQL Managed Instance.

Cassandra

Connection Method

Specifies the type of connection you want to use. Select *Direct* to connect to connect to your cluster directly. Select *Connection String* to connect to your cluster using a connection string.

Hostname/IP

Specifies the hostname or IP address of the server where your cluster is hosted.

Port

Specifies the port configured for your cluster.

Connection String

Specifies the path to the secure connect ZIP file in the following format:

C:\<file name>.zip

For example, *C:\TempCass\secure-connect-testdb.zip*

Couchbase

Connection Method

Specifies the type of connection you want to use. Select *Direct* to connect to connect to your bucket directly. Select *Connection String* to connect to your bucket using a connection string.

Hostname/IP

Specifies the hostname or IP address of the server where your bucket is hosted.

Port

Specifies the port configured for your bucket.

Bucket

Specifies the name of the bucket to which you want to connect.

SSL Certificate Path

Specifies the path to the SSL certificate, if you have one. You can leave this field blank.

Connection String

Specifies the connection string in the following format:

couchbases://<database server>/<bucket>?ssl=no_verify

For example, *couchbases://server1.dp.cloud.couchbase.com/testbucket?ssl=no_verify*

MongoDB

Connection Method

Specifies the type of connection you want to use. Select *Direct* to connect to connect to your database directly. Select *Connection String* to connect to your database using a connection string.

Hostname/IP

Specifies the hostname or IP address of the server where your database is hosted.

Port

Specifies the port configured for your database.

Database

Specifies the name of the database to which you want to connect.

Connection String

Specifies the connection string in the following format:

mongodb://[username:password@]host1[:port1][,...hostN[:portN]][/[defaultauthdb][?options]]

For example, *mongodb+srv://myusername:**** @cluster0.v7gra.mongodb.net/test?retryWrites=true&w=majority*

MySQL

Hostname/IP

Specifies the hostname or IP address of the server where your database is hosted.

Port

Specifies the port configured for your database.

Database

Specifies the name of the database to which you want to connect.

Note: For the JDBC connection to work seamlessly, ensure that you download the required JDBC driver and rename it to mysql-connector-java-8.0.22.jar.

MariaDB

Connection String

Specifies the connection string in the following format:

jdbc:mariadb

Hostname/IP

Specifies the hostname or IP address of the server where your database is hosted.

Port

Specifies the port configured for your database.

Database

Specifies the name of the database to which you want to connect.

Note: For the JDBC connection to work seamlessly, ensure that you download the required JDBC driver and rename it to mariadb-java-client-2.6.1.jar.