erwin Data Modeler

Feature Tour

Release 12.0

Legal Notices

This Documentation, which includes embedded help systems and electronically distributed materials (hereinafter referred to as the “Documentation”), is for your informational purposes only and is subject to change or withdrawal by Quest Software, Inc and/or its affiliates at any time. This Documentation is proprietary information of Quest Software, Inc and/or its affiliates and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of Quest Software, Inc and/or its affiliates

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all Quest Software, Inc and/or its affiliates copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to Quest Software, Inc and/or its affiliates that all copies and partial copies of the Documentation have been returned to Quest Software, Inc and/or its affiliates or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, QUEST SOFTWARE, INC. PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL QUEST SOFTWARE, INC. BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF QUEST SOFTWARE, INC. IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is Quest Software, Inc and/or its affiliates.

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) 12.0 release.

The features and enhancements introduced in this release are:

- ArangoDB
- Amazon Keyspaces
- Google BigQuery
- Databricks
- DynamoDB
- Neo4j
- Parquet
- Couchbase 7.0
- Central Scheduler
- Git Support
- MongoDB: Schema Validation
- Snowflake Enhancements
- Cassandra: Deriving Models and Advanced Denormalization
- Oracle: View and Materialized View Enhancement
- Azure Synapse: Table Constraint Enhancement
- Diagramming: Hide and Unhide Diagram Nodes
- Data Vault Enhancements
- Productivity and UI Enhancements

- DM Connect for DI
- erwin Mart Server Enhancements

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

ArangoDB Support

erwin Data Modeler (DM) now supports ArangoDB 3.8 and above as a target database. This implementation supports the following objects:

- Collection
  - Field
  - Index
- Database
- Graph
  - Graph Edge
- Index
- Relationship
- Task
- User ID
- Views

The following is the list of supported data types:

- Array
- Boolean
- Double
- Integer
- Null
- Object
- String

ArangoDB implementation supports all erwin DM features and functions. The following sections walk you through these features:

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

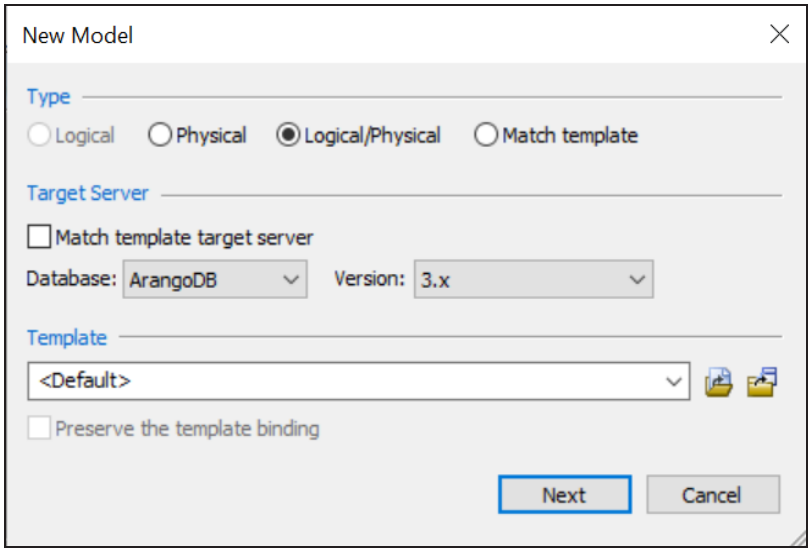
Reverse Engineering Models

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

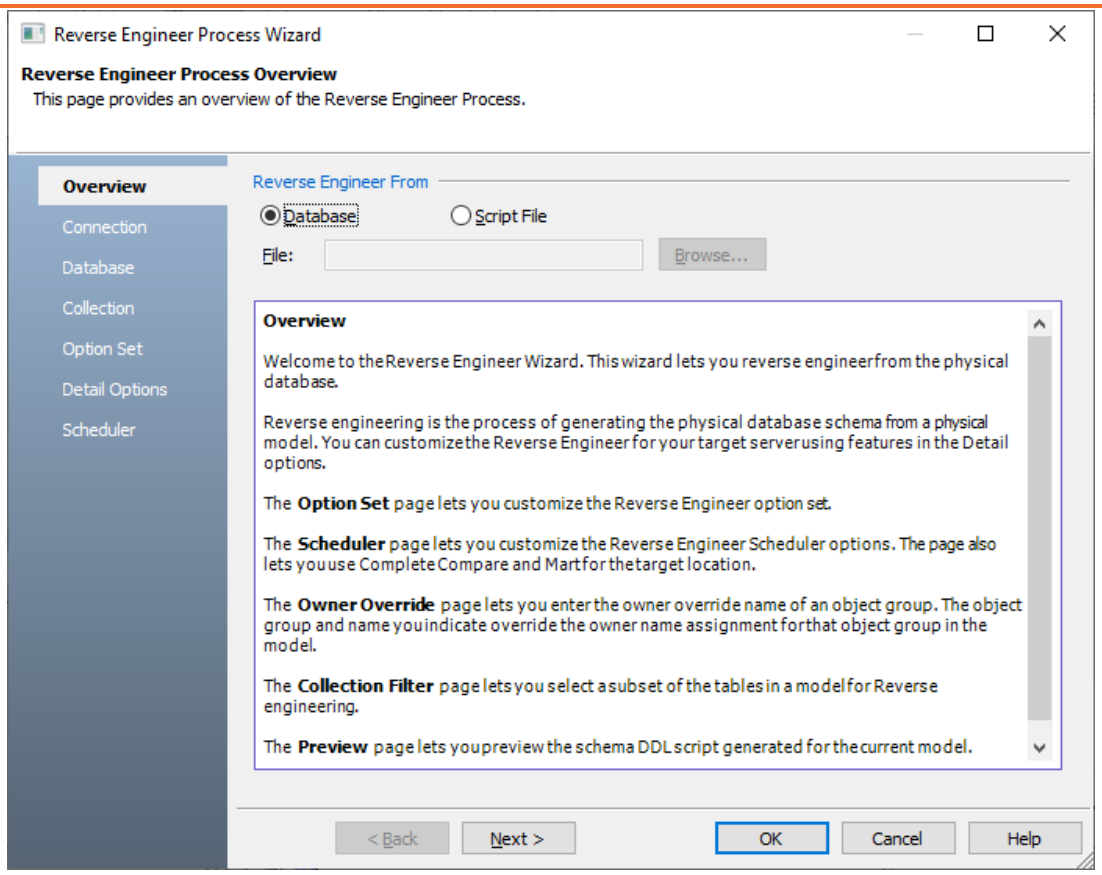
This topic walks you through the steps to reverse engineer an ArangoDB model. For detailed description of reverse engineering options, refer to the Reverse Engineering Options topic.

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 ArangoDB.



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.

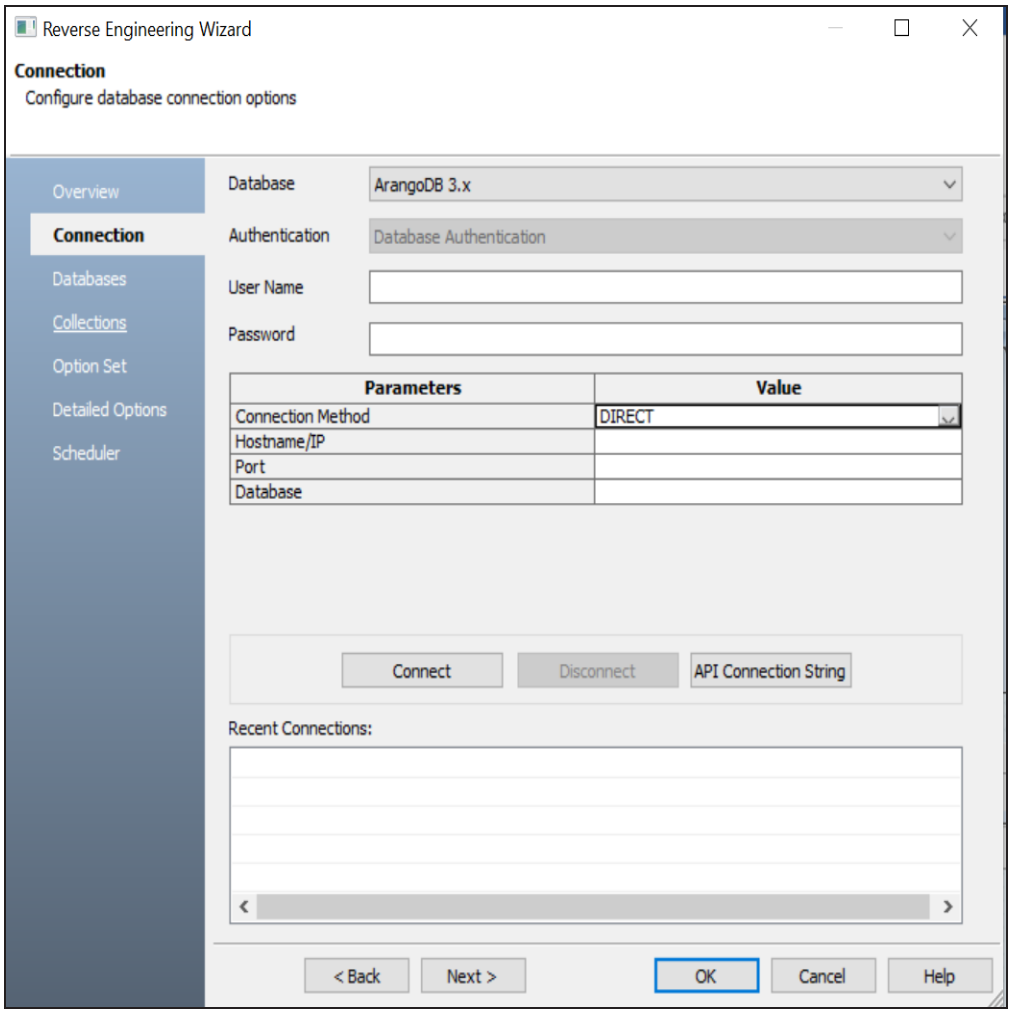
     If you click **Database**, continue to step 5.

   - **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 JSON file.

     If you click **Script File**, see step 13 below and ensure that Document Count or Document % is not set to zero (0).

5. Click **Next**.

The Connection tab appears.

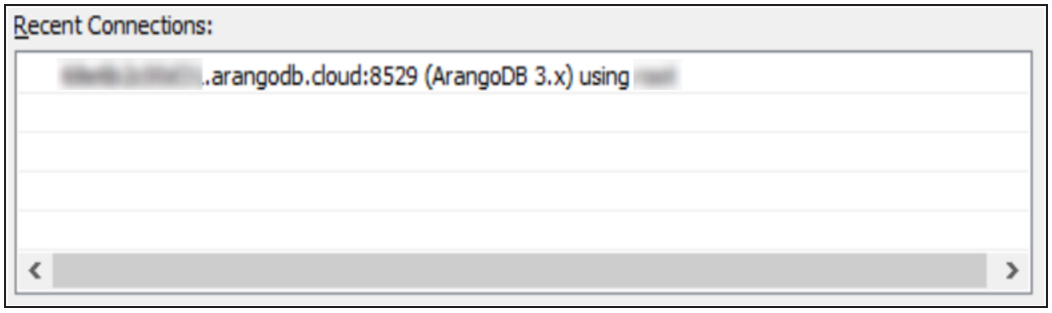


Use this tab to connect to the database from which you want to reverse engineer the model. You can connect to the database directly. The following table explains the connection parameters:

Parameter	Description	Additional Information
Connection Method	Specifies the type of connection you want to use. Select **Direct** to connect to your database directly.	
Hostname/IP	Specifies the hostname or IP address of the server	

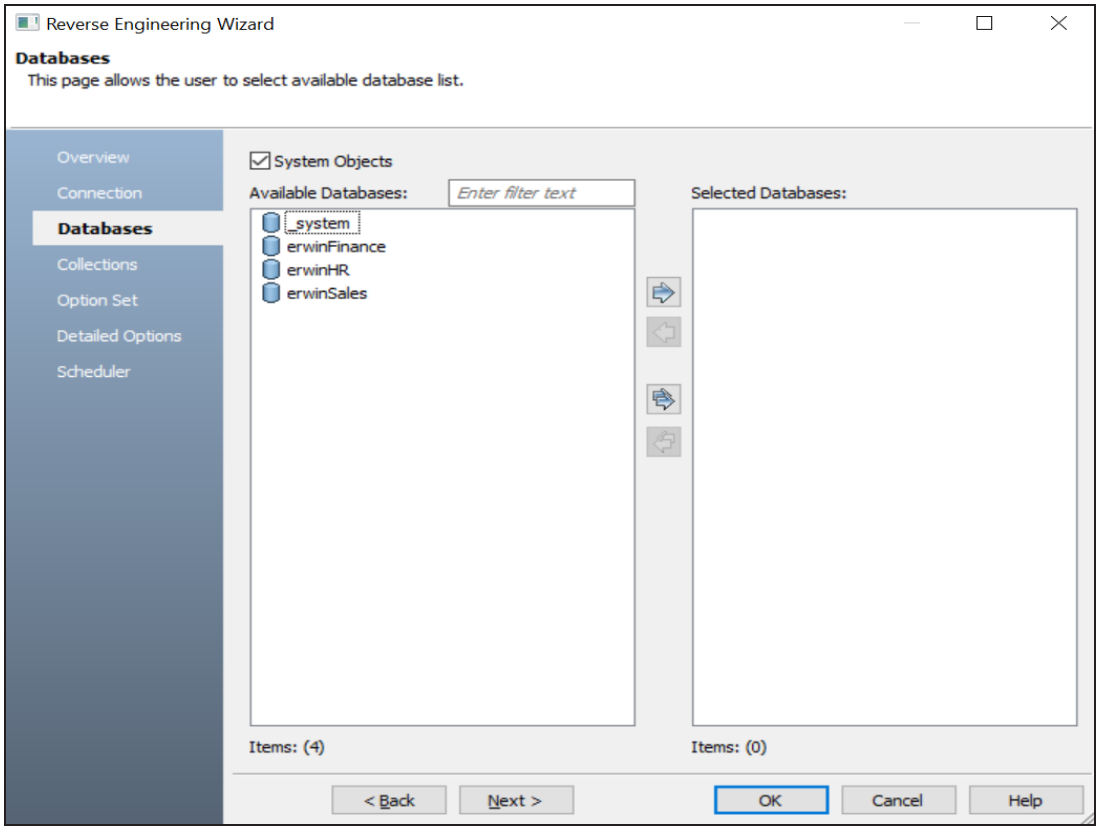
	where your database is hosted	
Port	Specifies the port configured for your database	Default port number is 8529.
Database	Specifies the name of the database to which you want to connect	

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

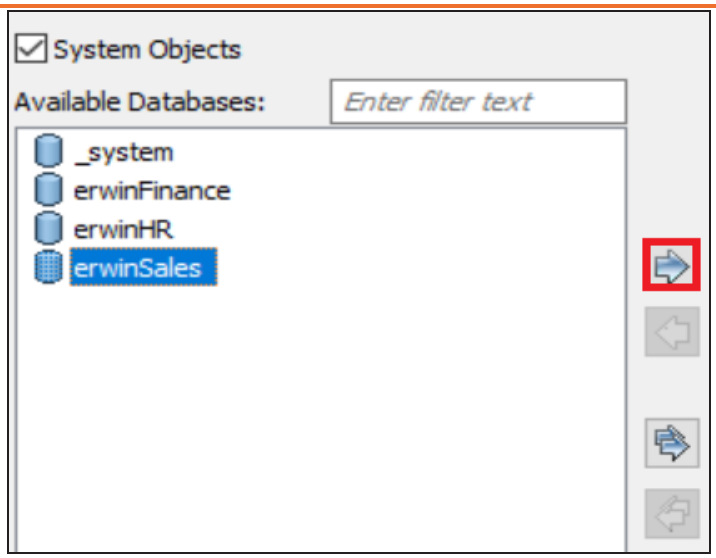


7. Click **Next**.

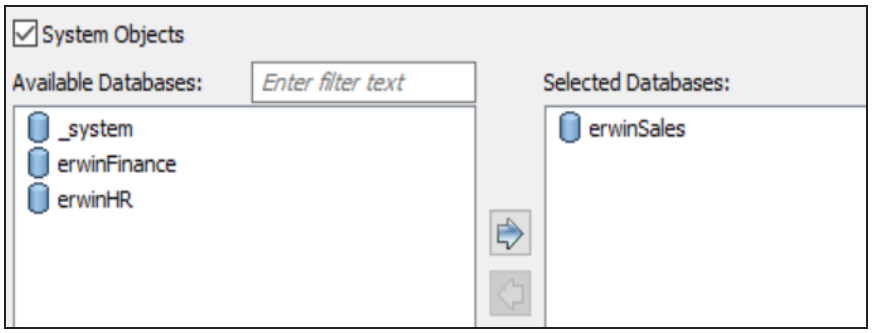
The Databases tab appears. It displays a list of available databases.



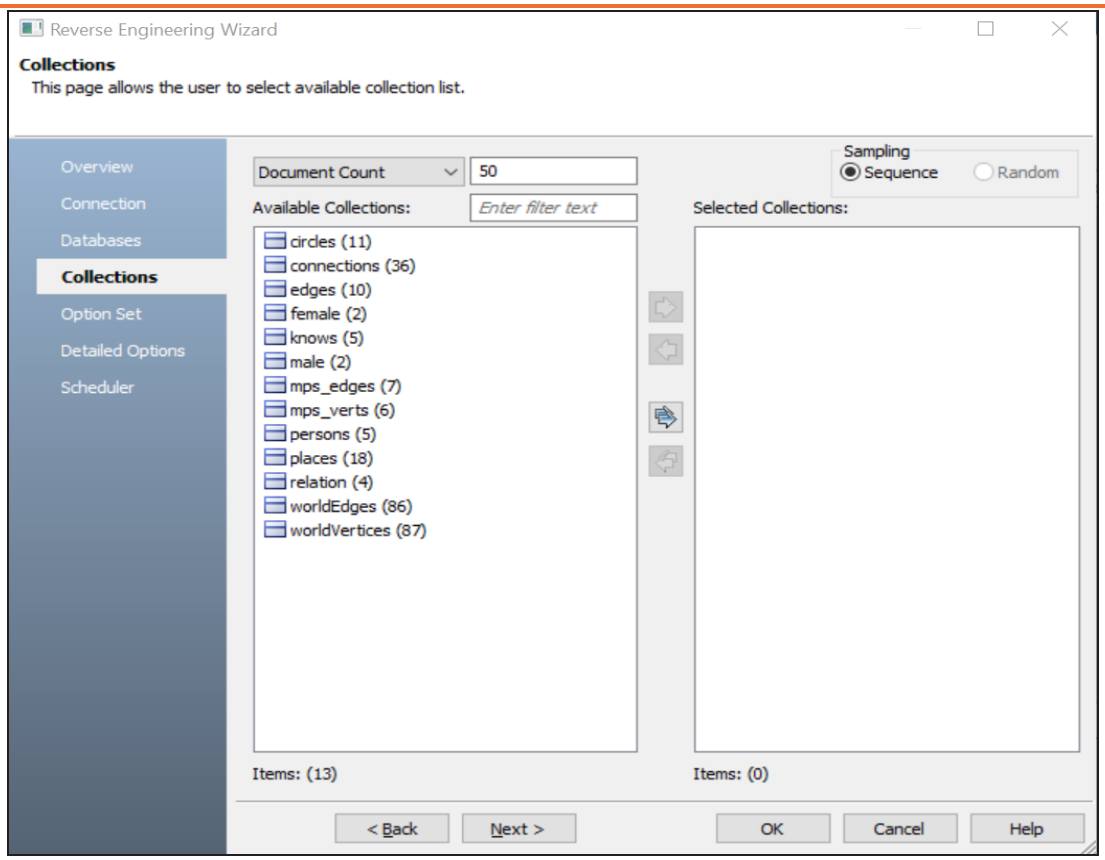
8. Under **Available Databases**, select the databases that you want to reverse engineer. Then, click  .



This moves the selected databases under Selected Databases.



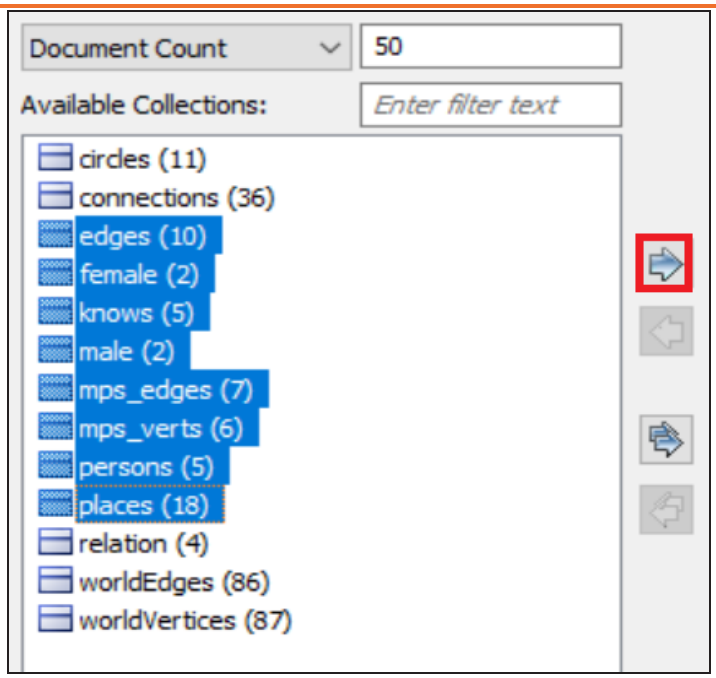
9. Click **Next**.
   The Collections tab appears. It displays a list of available collections in the databases that you selected in step 8.



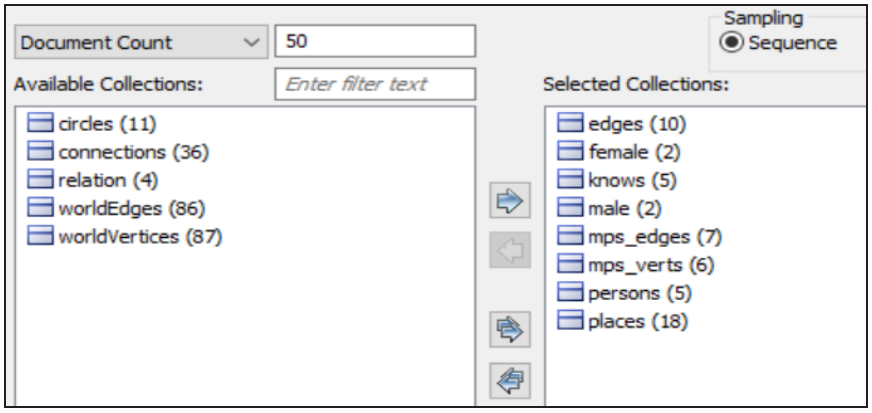
10. Use the following options:

    - **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.
    - **Sampling**: Use the Sequence sampling method to sample records in the selected collections. Sampling enables you to retrieve right estimates for accurate collection schema generation.

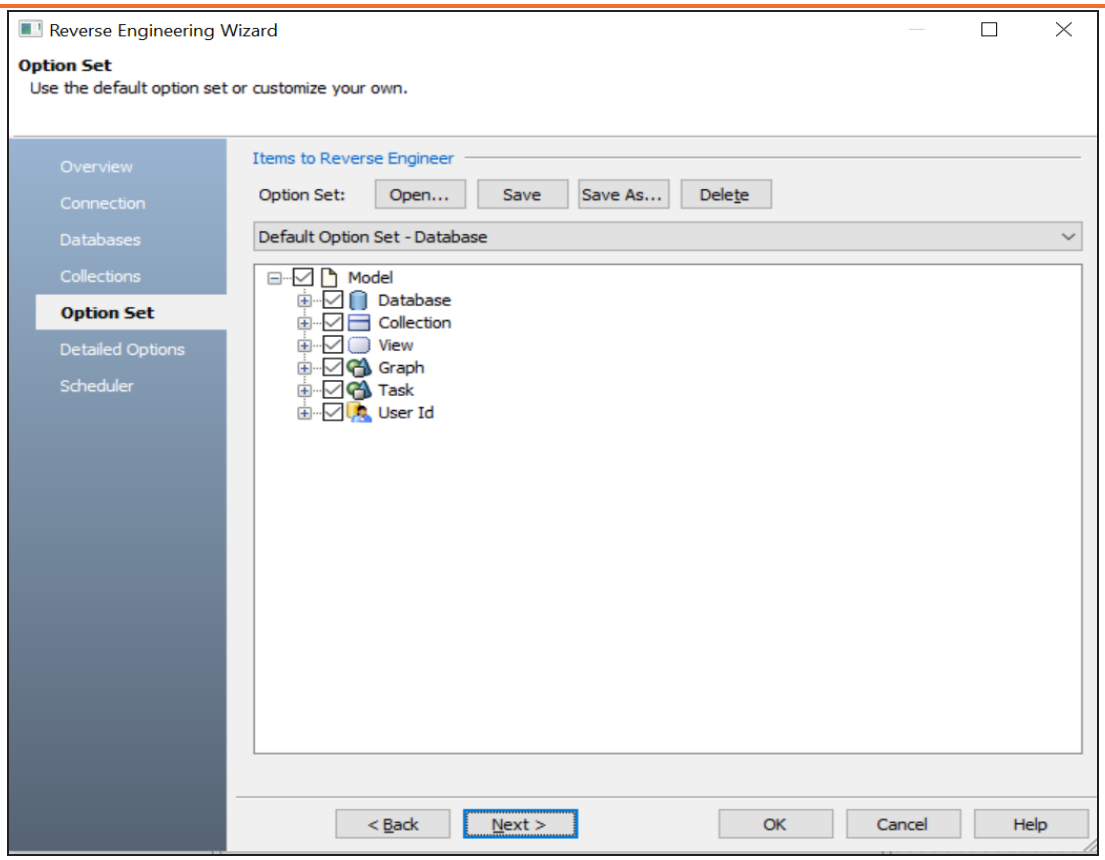
11. Under **Available Collections**, select the collections that you want to reverse engineer. Then, click  .



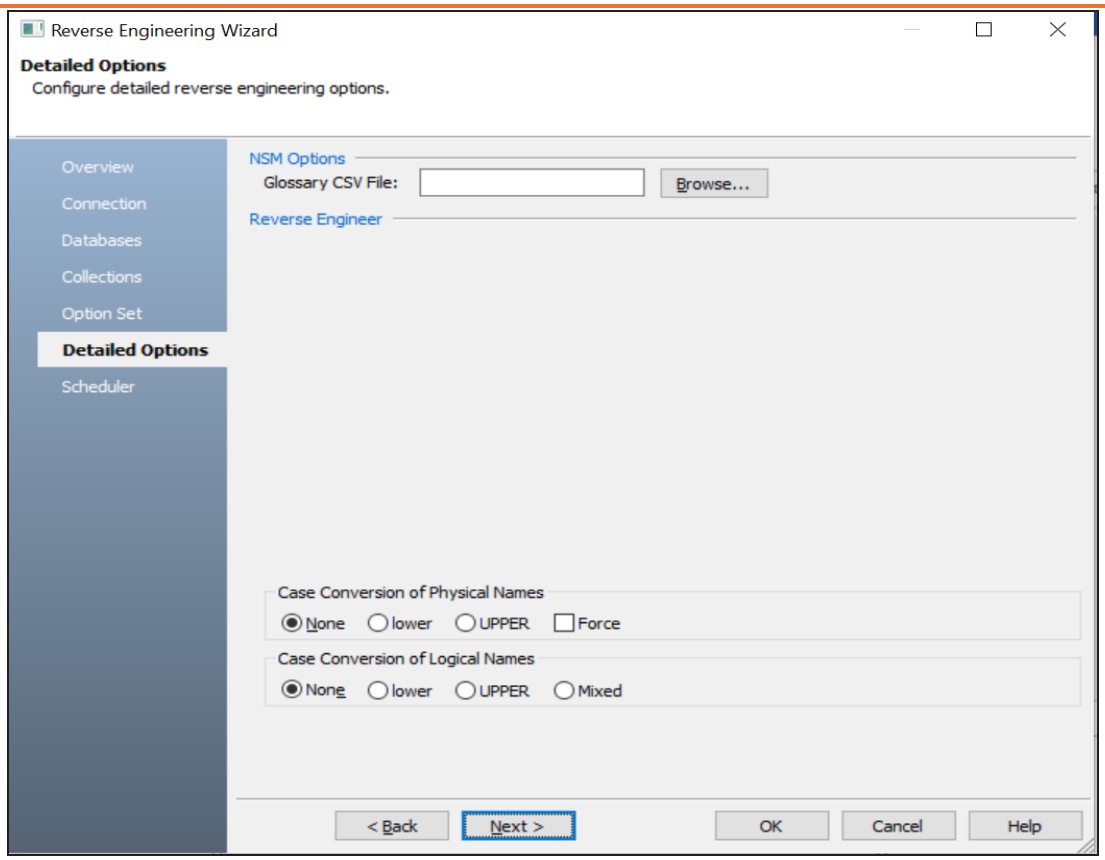
This moves the selected collections under Selected Collections.



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

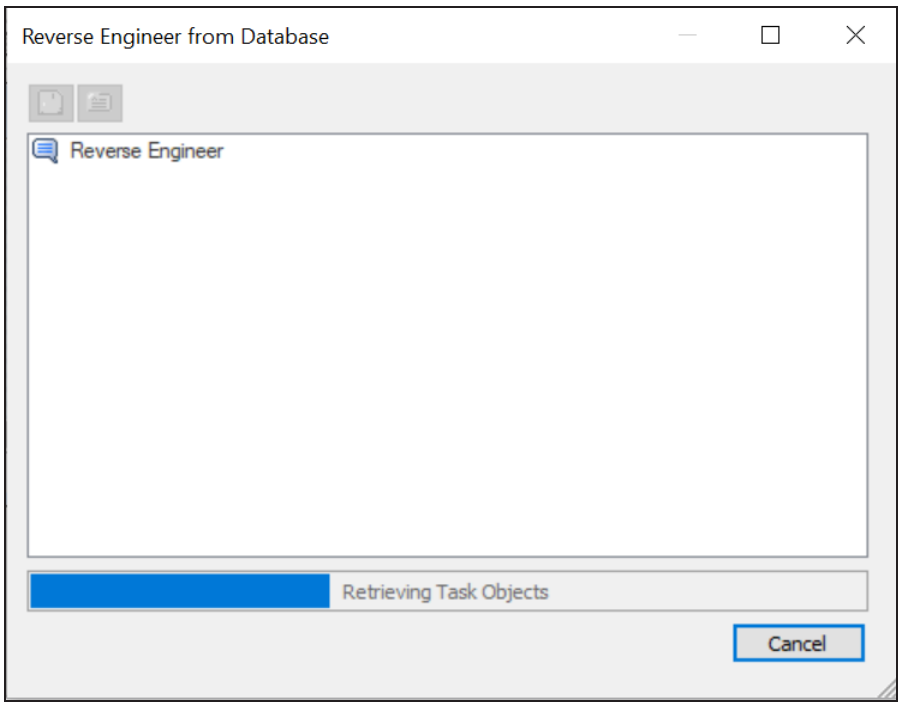


13. Click **Next**.
    The Detailed Options tab appears. Set up appropriate options based on your requirement.

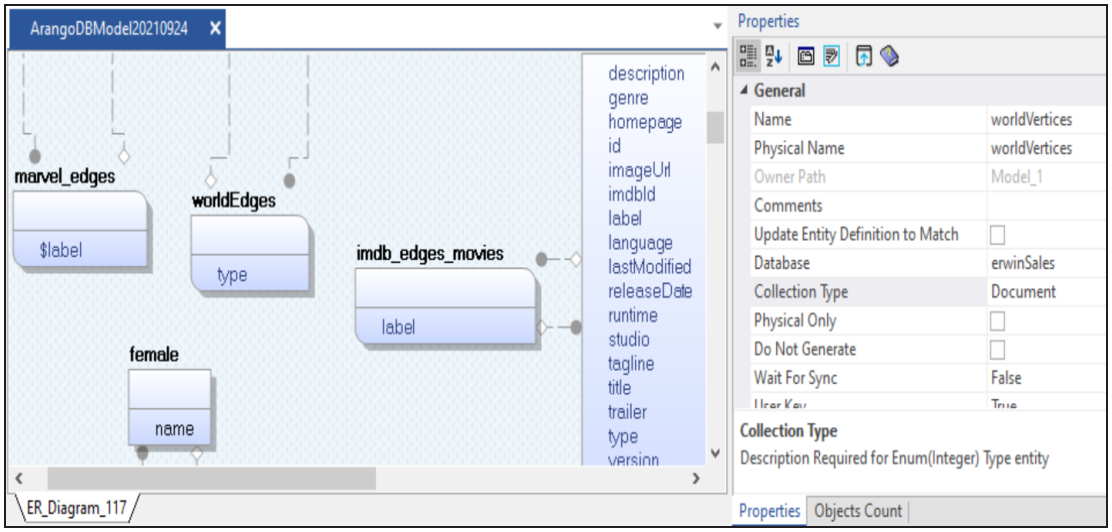

Reverse Engineering Wizard
Detailed Options
Configure detailed reverse engineering options.
Overview
Connection
Databases
Collections
Option Set
Detailed Options
Scheduler
NSM Options
Glossary CSV File:
Browse...
Reverse Engineer
Case Conversion of Physical Names
None
lower
UPPER
Force
Case Conversion of Logical Names
None
lower
UPPER
Mixed
< Back
Next >
OK
Cancel
Help

14. Click **OK**.

The reverse engineering process starts.



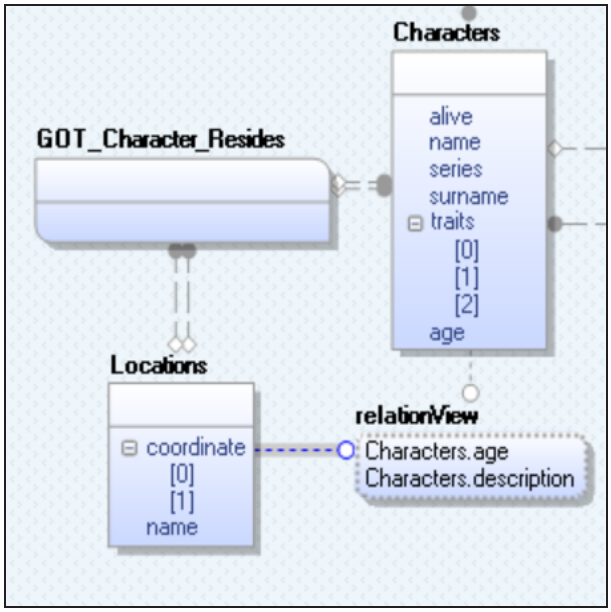
Once the process is complete, based on your selections, a schema is generated and a model is created.



ArangoDb has two types of collections:

- **Documents**: Documents contain data or schema. They are represented by rectangles in the ER diagram. For example, in the above model, **female** is a document.
- **Edges**: Edges contain relationship between document data points. They are represented by curved rectangles in the ER diagram. For example, in the above model, imdb_edge_movies is an edge.

The ER diagram displays the relationship between two documents via edges. For example, in the following model Characters is rated to Locations via the GOT_Character_Resides edge.



Reverse Engineering Options for ArangoDB

Following are the reverse engineering options for ArangoDB.

Overview

Parameter	Description	Additional Information
Reverse Engineer From	Specifies whether you want to reverse engineer from a script or database	**Database**: Indicates that the model is reverse engineered from database **Script File**: Indicates that the model is reverse engineered from a script
File	Specifies the script file location	This option is available when Script File is selected.

Connection

Parameter	Description	Additional Information
Connection Method	Specifies the type of connection you want to use. Select **Direct** to connect to your database directly.	
Hostname/IP	Specifies the hostname or IP address of the server where your database is hosted	
Port	Specifies the port configured for your database	Default port number is 8529.
Database	Specifies the name of the database to which you want to connect	

Databases

Parameter	Description	Additional Information
Available Data-	Specifies a list of available databases	

bases		
Selected Databases	Specifies a list of selected databases for reverse engineering	
System Objects	Specifies whether system databases are included under the Available Databases	

Collections

Parameter	Description	Additional Information
Document Count/Document (%)	Specifies the number of documents or percentage of total records that the newly generated model schema would contain	
Sampling	Specifies that the sampling method is Sequence. Sampling enables you to retrieve right estimates for accurate collection schema generation.	
Available Collections	Specifies a list of available collections	
Selected Collections	Specifies a list of selected collections for reverse engineering	

Option Sets

Parameter	Description	Additional Information
Option Set	Specifies the option set template for reverse engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save the configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at some external location. **Delete**: Use this option to delete an option set.

<Option Set Name>	Specifies the objects to be reverse engineered according to the selected option set. You can edit this list.	

Detailed Options

Parameter	Description	Additional Information
NSM Options	Specifies the naming standard glossary file in the .CSV format	
Case Conversion of Physical Names	Specifies how the case conversion of physical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Force**: Indicates whether the physical name property of all the logical/physical models is overridden. If this option is enabled, the logical/physical link is broken between the logical and physical name. If this option is not enabled, all logical and physical names are set to the same value after the process completes.
Case Conversion of Logical Names	Specifies how the case conversion of logical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Mixed**: Indicates that the mixed-case logical names are preserved

Scheduler

Parameter	Description	Additional Information
Model	Specifies the location and name of the reverse engin-	For example: C:\Scheduler\<Model Name>.erwin

	eered model	When you schedule a job on a remote server, ensure the model path is same for remote and local server.
Mart Folder	Specifies the location or library in your mart where the reverse engineered model is saved	To use this option, ensure that you are connected to a mart. For more information, refer to the Connecting to Mart topic.
Complete Compare	Specifies whether the Complete Compare (CC) process should run while reverse engineering	
Output File	Specifies the location of the CC output file generated	
File	Specifies that the target model location is on the local system	
Mart	Specifies that the target model location is in the mart	
Using Latest Version	Specifies whether the target model is the latest version of the model in the mart	This option is available only when Mart is selected.
Save To Mart	Specifies whether the reverse engineered model is saved to the mart	This option is available only when Using Latest Version is selected.
Target Model	Specifies the location of the target model for CC	
Option Set	Specifies the option set that is used for CC	**Advanced Default Option Set**: Indicates that all erwin DM metadata is included. CC works slowest with this option. **Speed Option Set**: Indicates that only the essential metadata is included. CC works the fastest with this option set. **Standard Default Option Set**: Indicates that

		standard metadata is included. CC works fast with this option set compared to the Advanced option set.

Forward Engineering Models

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

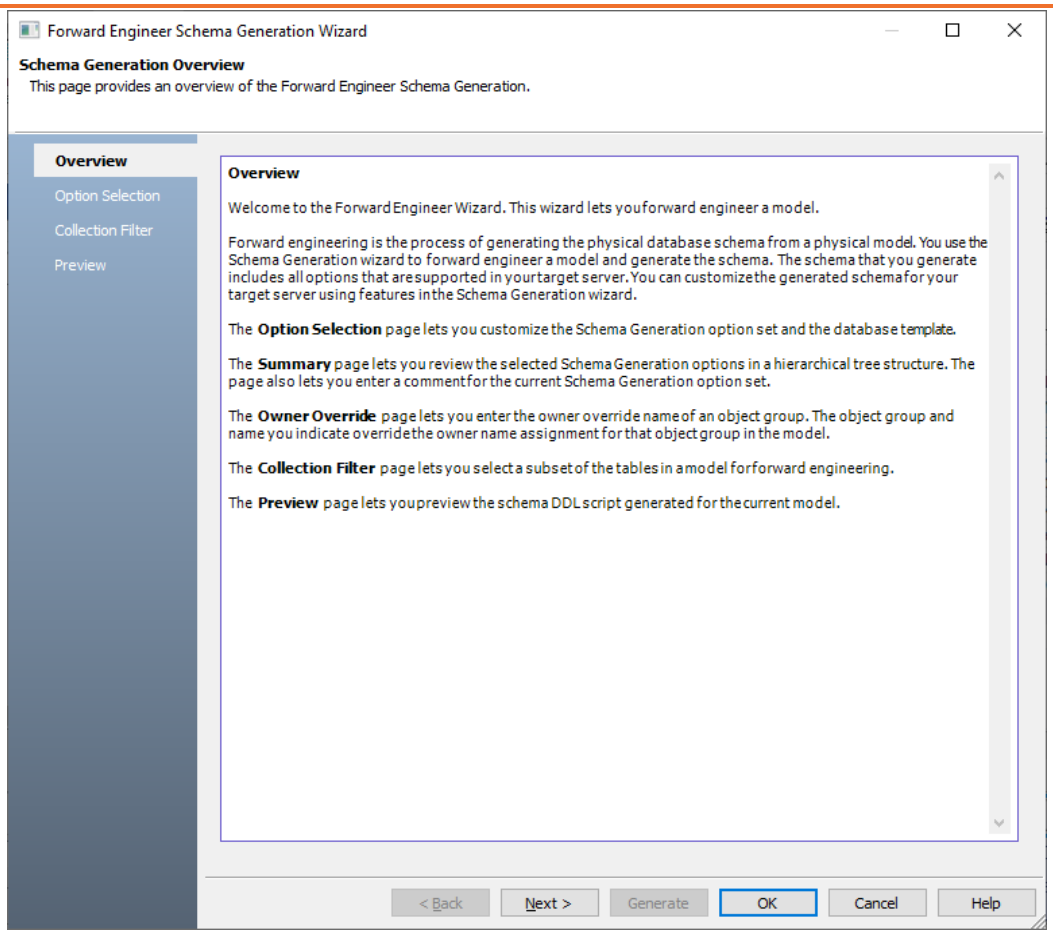
This topic walks you through the steps to forward engineer an ArangoDB model. For detailed description of forward engineering options, refer to the Forward Engineering Options topic.

To forward engineer a model:

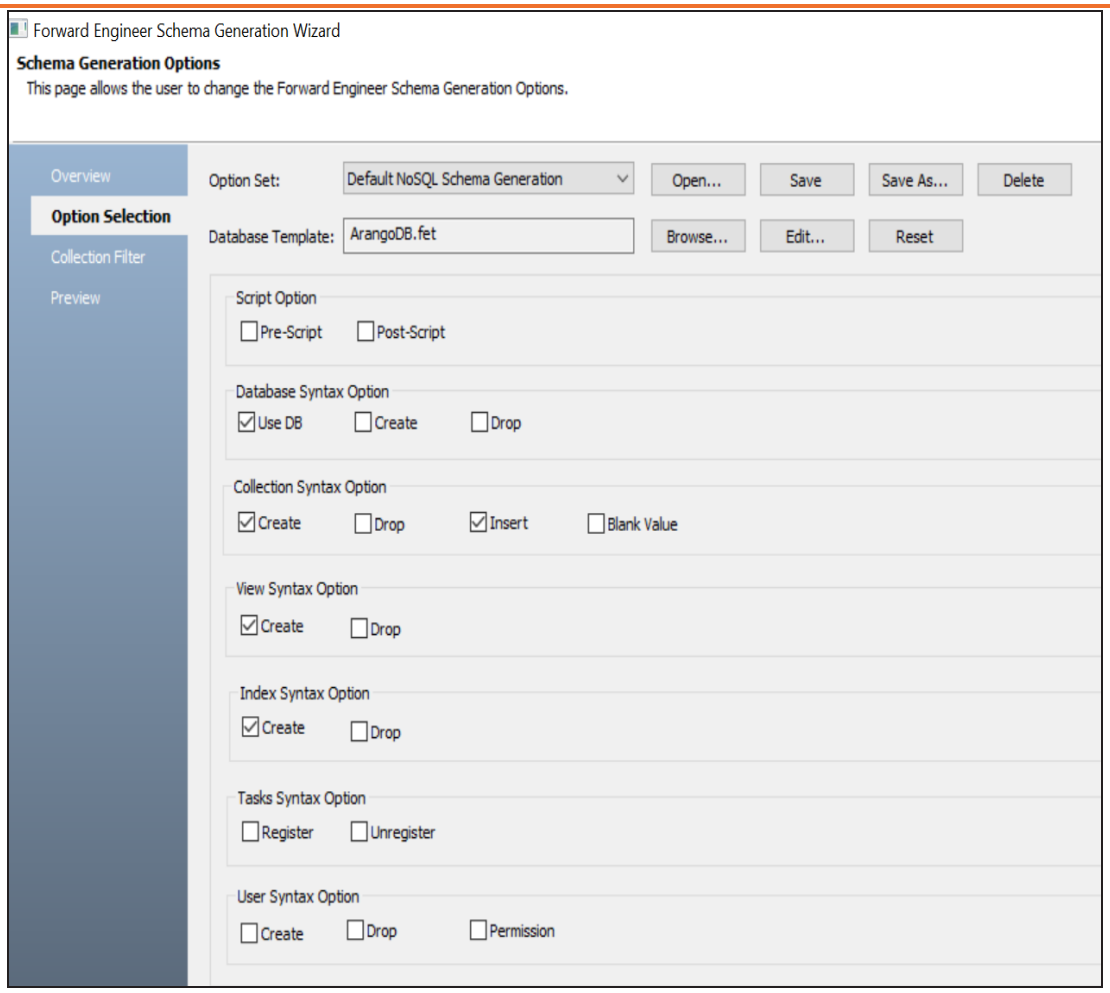
1. Open your ArangoDB model.

   
   Ensure that you are in the Physical mode.

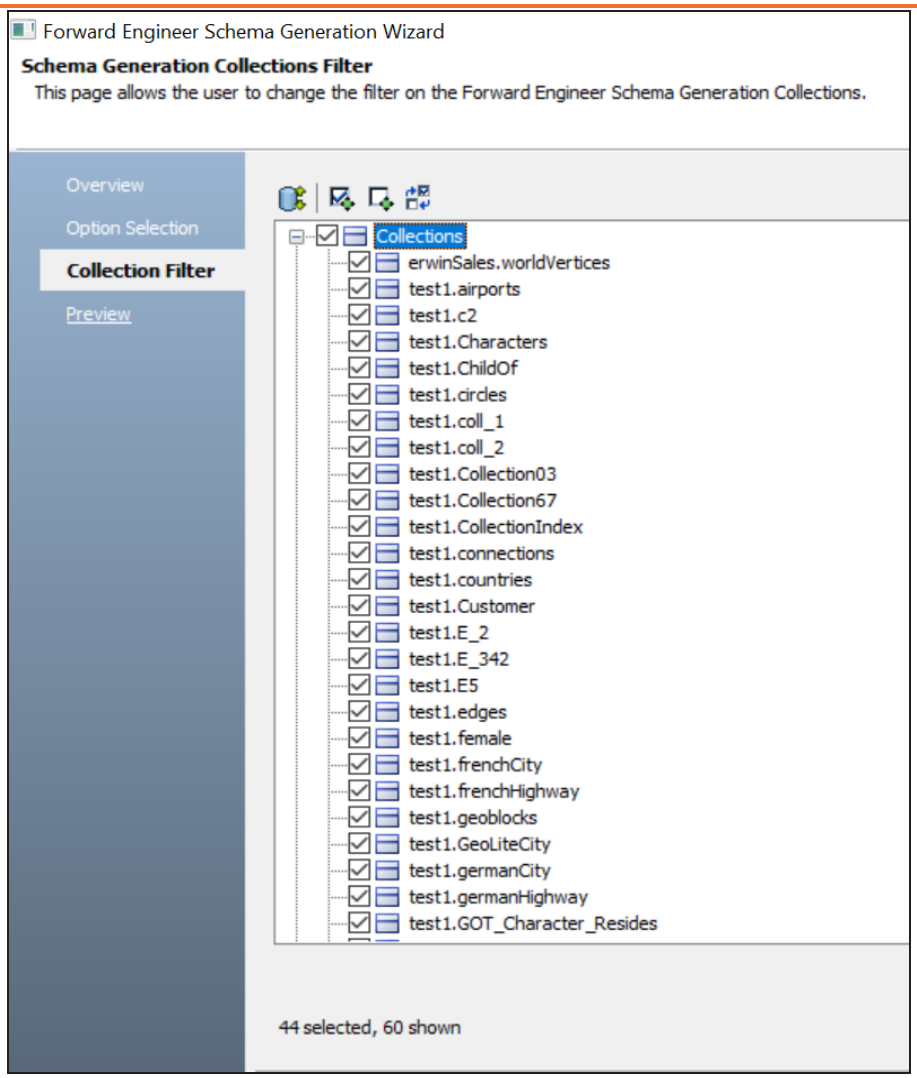
2. Click **Actions** > **Schema**.
   The Forward Engineer Schema Generation Wizard appears.



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

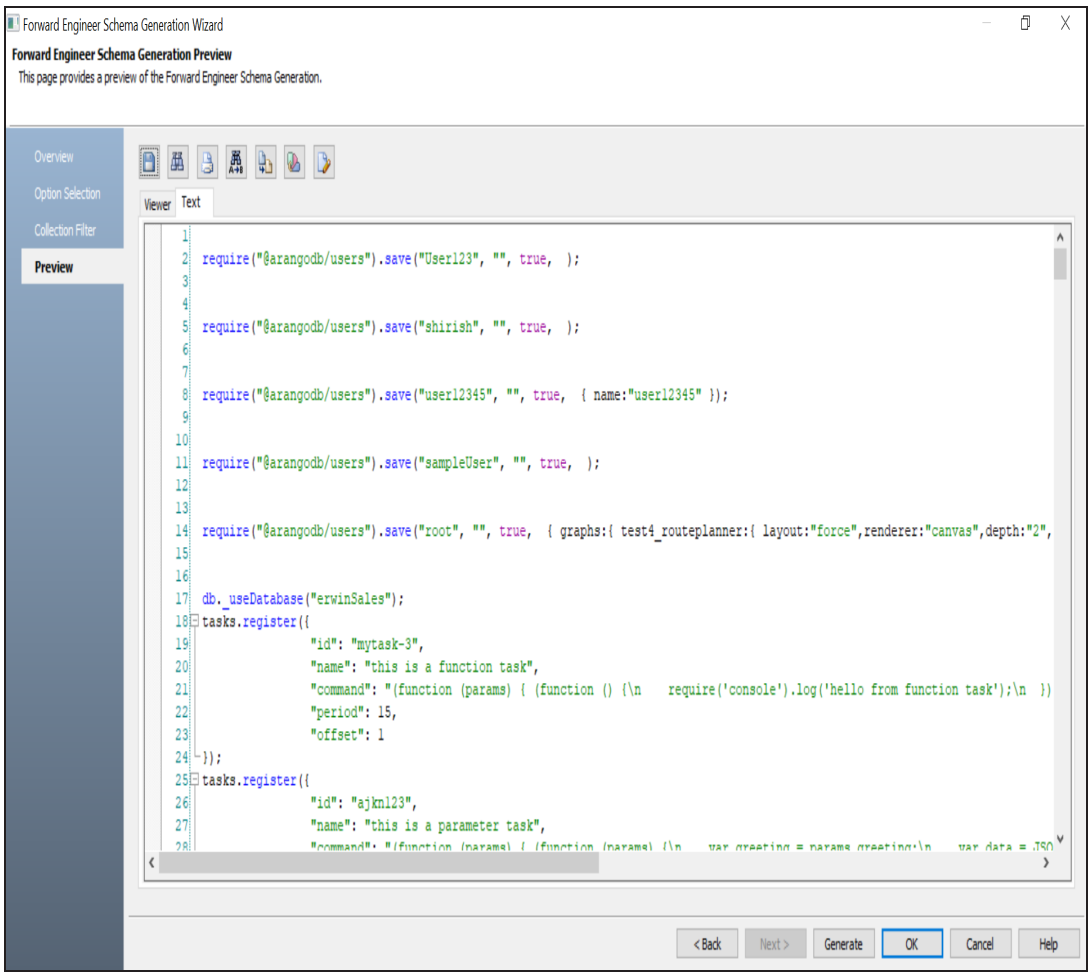


4. Click **Next**.
   The Collection Filter tab appears. It displays a list of collections available in your model.



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

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



Use the following options:

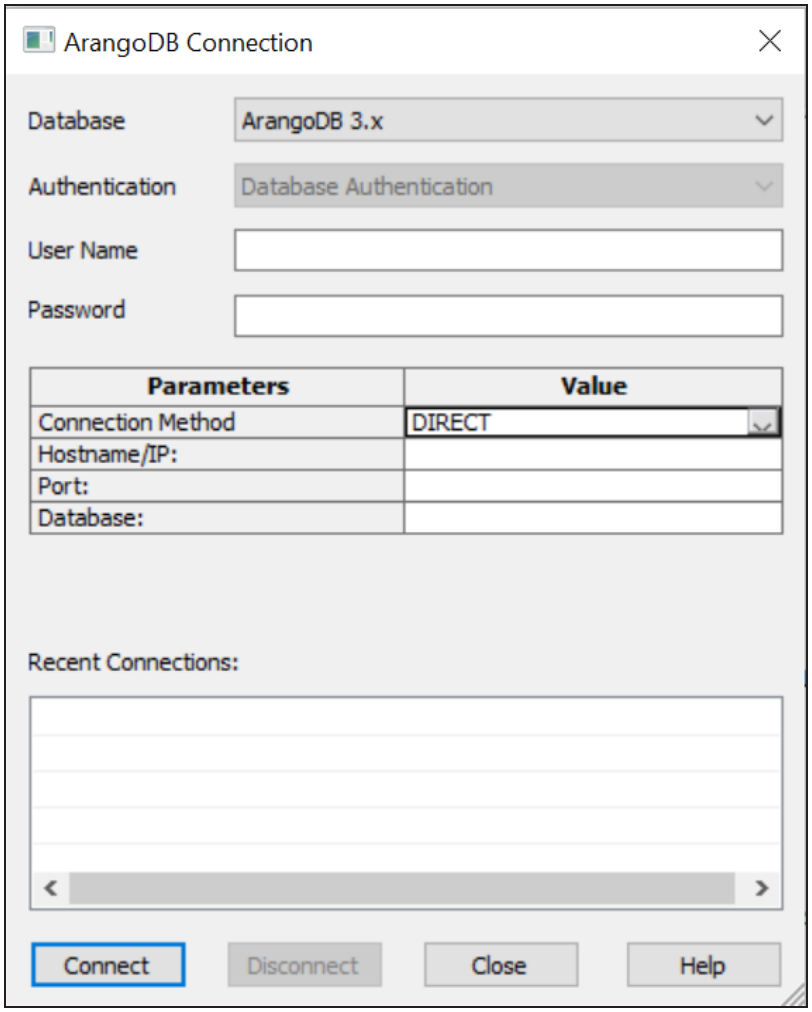
- **Auto Error Check**: Select this option to enable auto error check by the forward engineering wizard.
- **Error Check** (): Use this option to run an error check. Based on the results, you can correct the generated script.
- **Text Options** (): Use this option to configure the preview text editor's 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**.

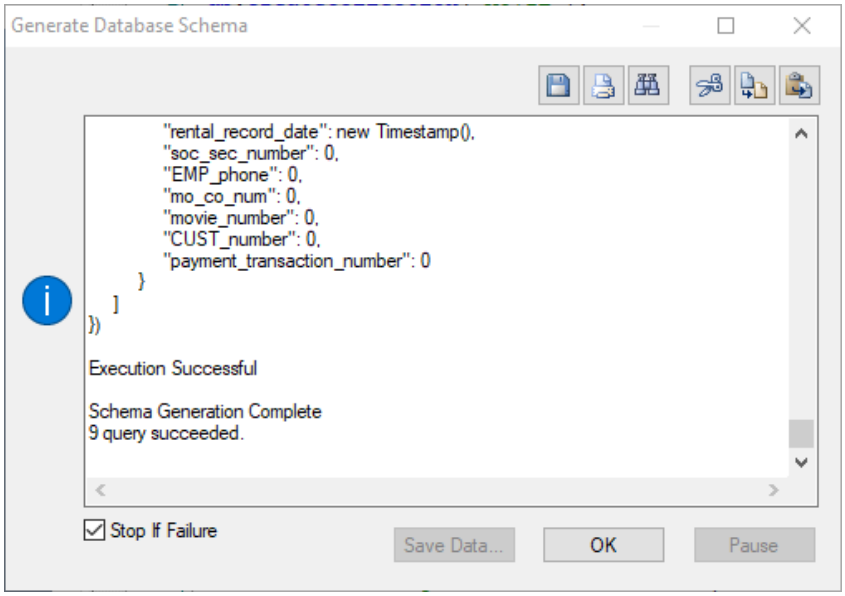
   The ArangoDB Connection screen appears.



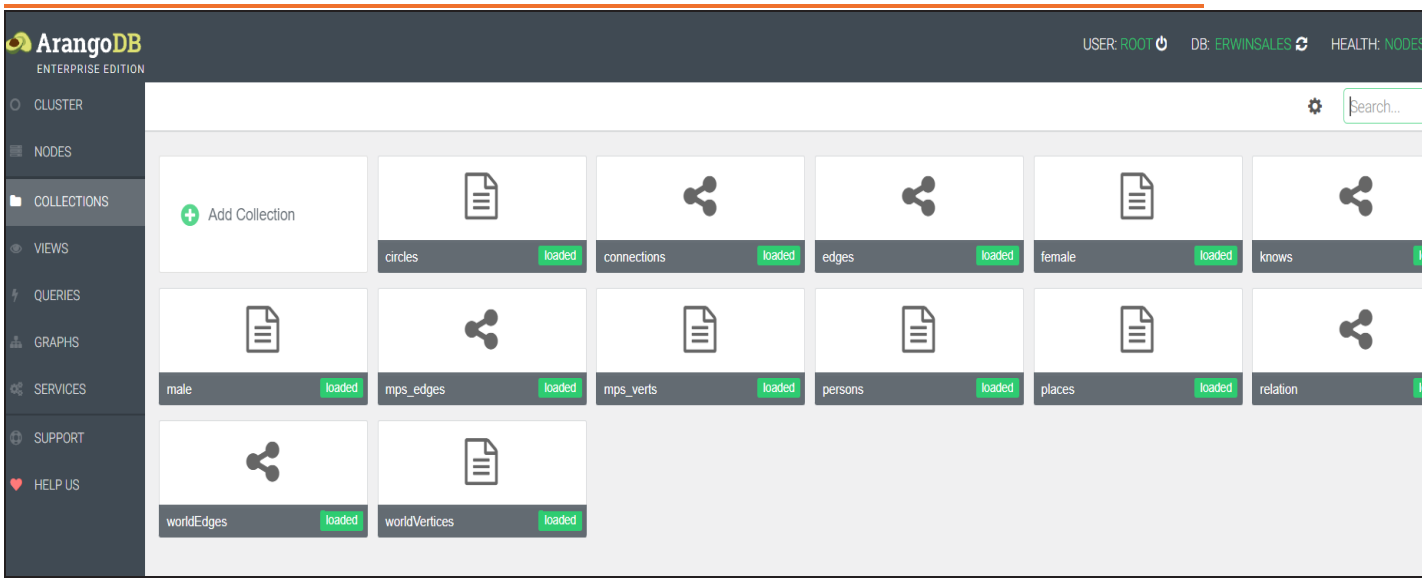
8. Enter user name, password, and appropriate connection parameters to connect the required database. Then, click **Connect**.

The objects move to a database entered on the ArangoDB Connection page irrespective of the databases entered on the object editor pages. If you want to move objects to databases as entered on object editors page then do not enter any database on the ArangoDB Connection page.

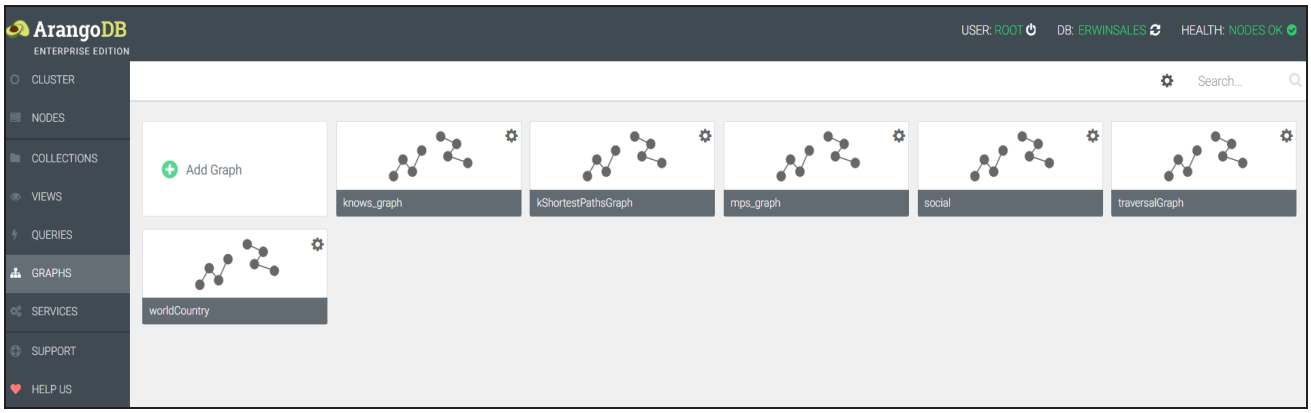
The forward engineering process starts. The script generates your physical database schema. You can access your database and verify the newly generated schema.



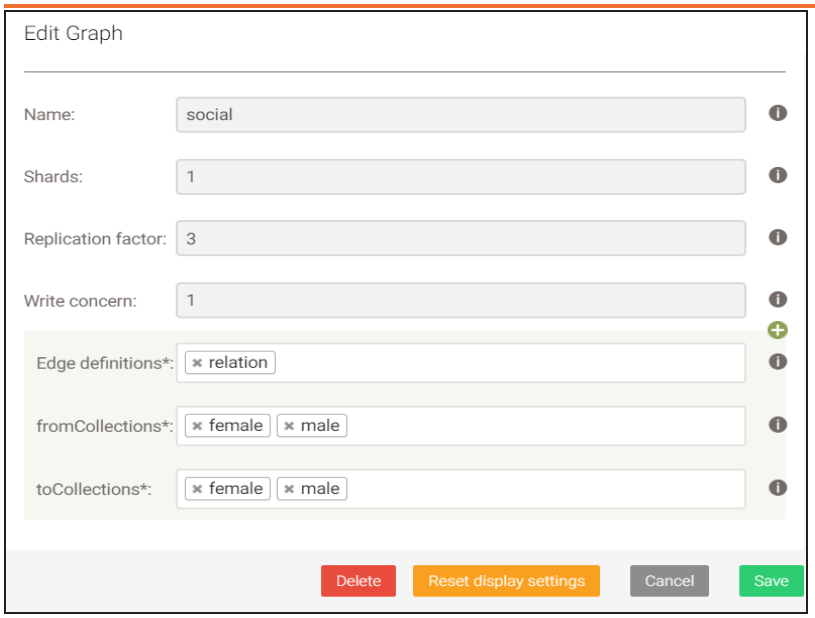
The forward engineering process creates database objects in the database entered on the ArangoDB Connection page. For example, in the following image, the forward engineering process creates 13 collections in the ERWINSALES database. ArangoDB represents edge collections using whereas document collections are represented using .



This process creates six graphs in ERWINSALES database.



You can view graph properties by clicking the required graph. For example, the following Edit Graph page displays the properties of the social graph.


Edit Graph
Name:
social
Shards:
1
Replication factor:
3
Write concern:
1
Edge definitions*:
relation
fromCollections*:
female
male
toCollections*:
female
male
Delete
Reset display settings
Cancel
Save

Forward Engineering Options for ArangoDB

Following are the forward engineering options for ArangoDB.

Option Selection

Parameter	Description	Additional Information
Option Set	Specifies the option set template for forward engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save a configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at some external location. **Delete**: Use this option to delete an option set.
Database Template	Specifies the database template for controlling schema generation	**Browse**: Use this option to browse and select a database template. **Edit**: Use this option to edit a template in the Template Editor. **Reset**: Use this option to reset the Database Template option.
Script Option	Specifies the script option for the schema generation	**Pre-Script**: Indicates whether pre-scripts attached to the schema are executed **Post-Script**: Indicates whether the post-scripts attached to the schema are executed
Database Syntax Option	Specifies the database syntax options for the schema generation	**Use DB**: Indicates whether the Use DB syntax for databases is executed **Create**: Indicates whether the Create syntax for databases is executed **Drop**: Indicates whether the Drop syntax for databases is executed

Collection Syntax Option	Specifies the collection syntax options for the schema generation	**Create**: Indicates whether the Create syntax for collections is executed **Drop**: Indicates whether the Drop syntax for collections is executed **Insert**: Indicates whether the Insert syntax for collections is executed **Blank Value**: Indicates whether the Blank Value syntax for collections is executed
View Syntax Option	Specifies the view syntax options for the schema generation	**Create**: Indicates whether the Create syntax for views is executed **Drop**: Indicates whether the Drop syntax for views is executed
Index Syntax Option	Specifies the index syntax options for the schema generation	**Create**: Indicates whether the Create syntax for indexes is executed **Drop**: Indicates whether the Drop syntax for indexes is executed
Tasks Syntax Option	Specifies the task syntax options for the schema generation	**Register**: Indicates whether the Register syntax for tasks is executed **Unregister**: Indicates whether the Unregister syntax for tasks is executed
User Syntax Option	Specifies user syntax options for the schema generation	**Create**: Indicates whether the Create syntax for users is executed **Drop**: Indicates whether the Drop syntax for users is executed **Permission**: Indicates whether the Permission syntax for users is executed
Graph Syntax Option	Specifies graph syntax options for the schema generation	**Create**: Indicates whether the Create syntax for graphs is executed **Drop**: Indicates whether the Drop syntax for graphs is executed

Collection Filter

Parameter	Description	Additional Information
Collections	Specifies the selected Collections for the schema generation	
Display either Logical Names or Physical Names		**Logical Names**: Indicates that only logical names of the collections are included in the generated schema **Physical Names**: Indicates that only physical names of the collections are included in the generated schema **Physical Names, show owner**: Indicates that physical names and owners of the collections are included in the generated schema **Physical Names, show owner using User**: Indicates that the physical names and owners of the collections are included in the generated schema. Owners of the collections are displayed using User.
Select all of the items in the list	Use this option to select all the collections in the list.	
Unselect all of the items in the list	Use this option to unselect all the collections.	
Select all unselected items, and unselect all selected items	Use this option to select all the unselected collections and unselect all the previously selected collections.	

Preview

Forward Engineering Options for ArangoDB

Parameter	Description	Additional Information
Viewer	Displays the schema in the viewer editor	**Collapse All**: Use this option to collapse all the nodes. **Search**: Use this option to search a text entered in the search box. **Find Previous**: Use this option to navigate to previous search string in the search results **Find Next**: Use this option to navigate to next search string in the search result.
Text	Displays the schema in the text editor	**Save**: Use this option to save the generated schema. **Search**: Use this option to search through the generated schema. **Print**: Use this option to print the generated schema. **Replace**: Use this option to find and replace text in the generated schema. **Copy**: Use this option to copy the selected text in the schema. **Text Options**: Use this option to edit window settings, fonts, and syntax color. **Error Check**: Use this option to check errors in the forward engineering script. **Git**: Use this option to commit the FE script to a Git repository.

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.

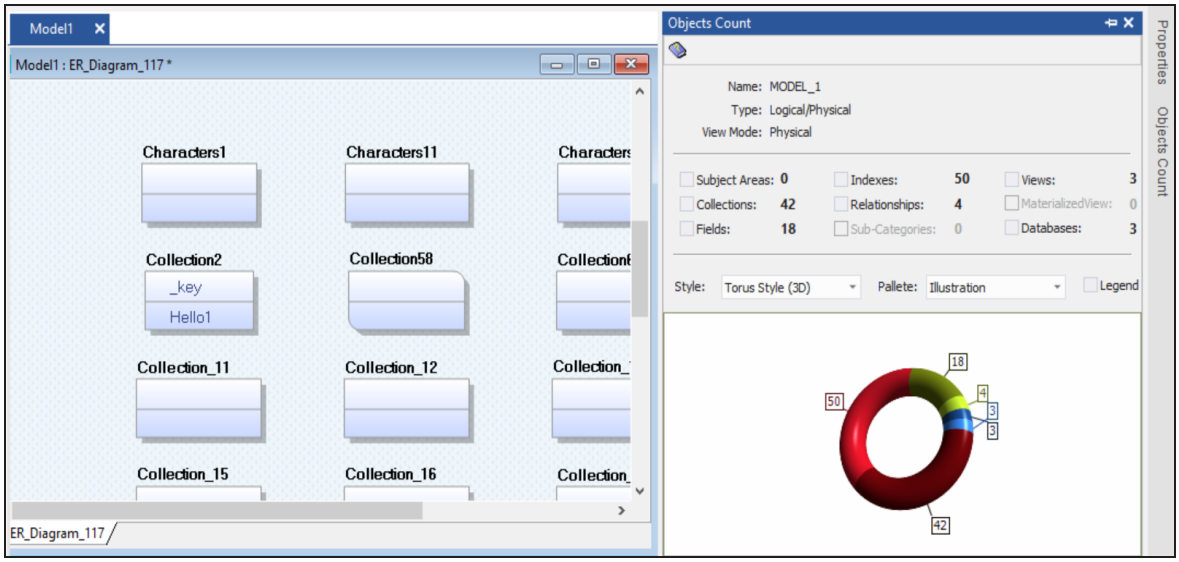
This topic walks you through the steps to compare an ArangoDB model with database.

To compare models with database:

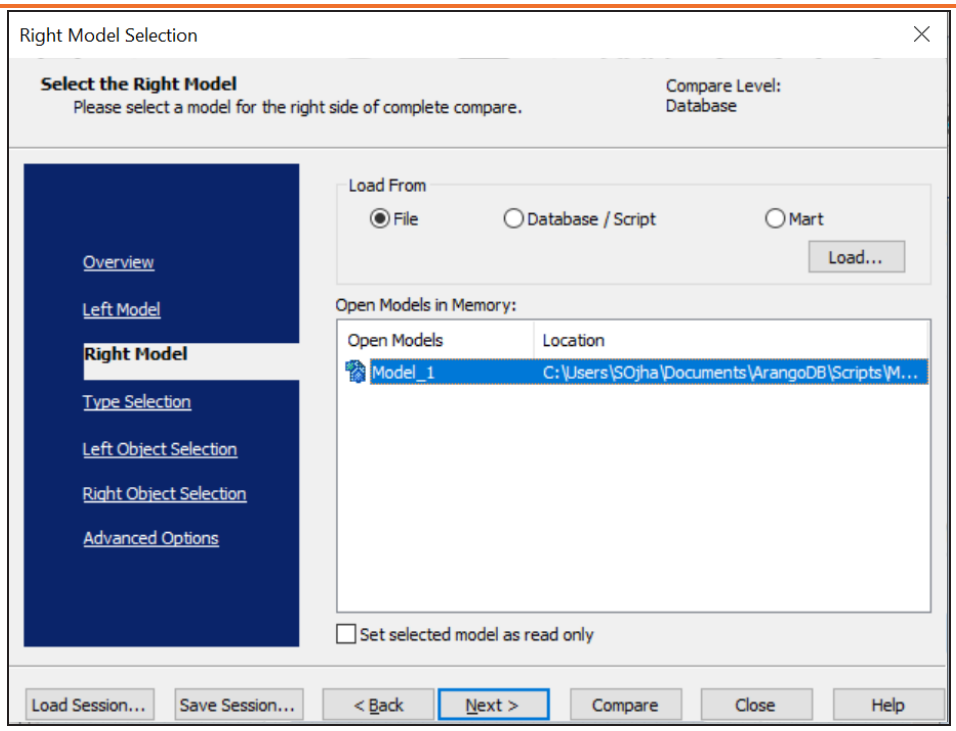
1. Open your ArangoDB model.

   Ensure that you are in the Physical mode.

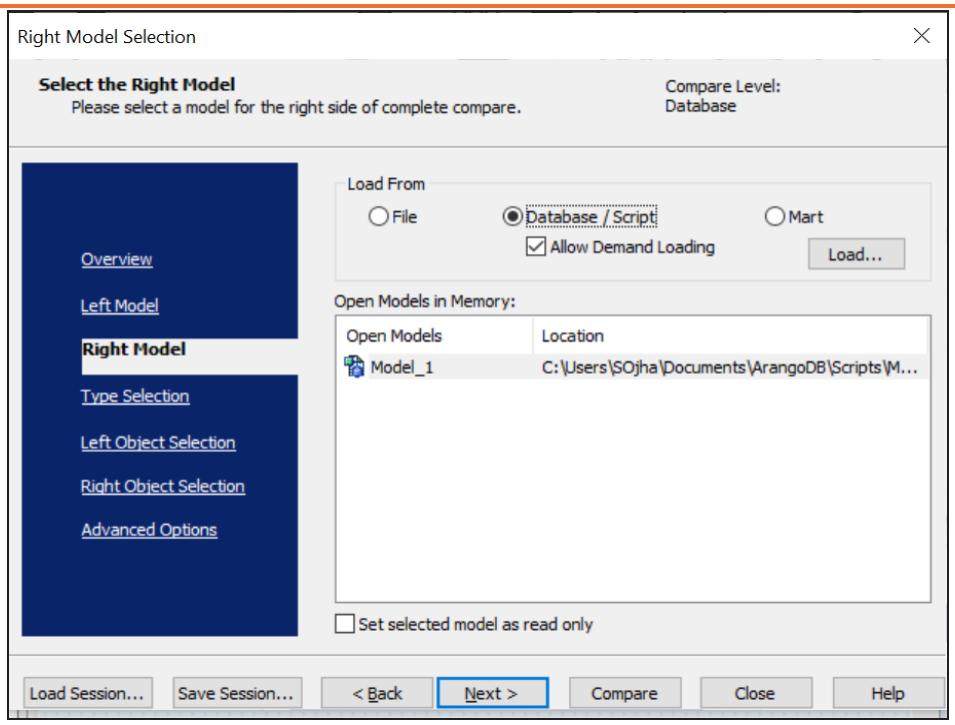
   For example, the following image uses an ArangoDB model with 42 collections.



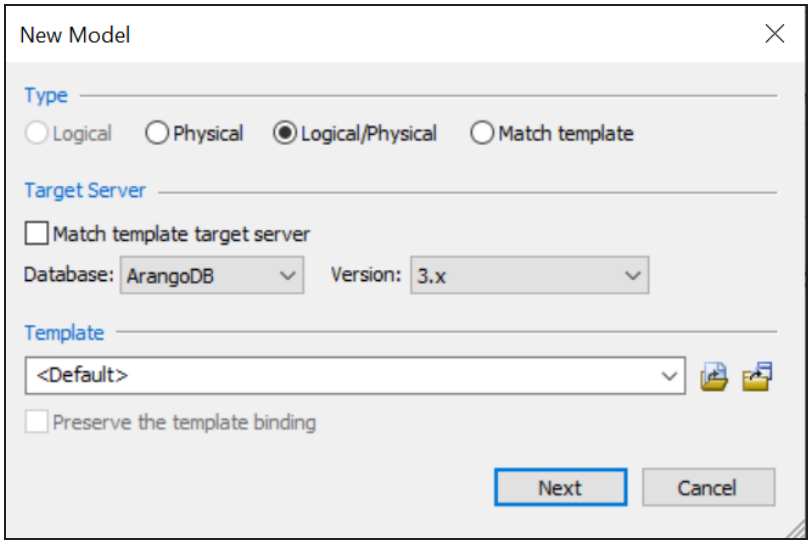
2. Click **Actions** > **Complete Compare**.
   By default, the Complete Compare wizard assigns the open model as the Left Model. Hence, the Right Model tab appears.



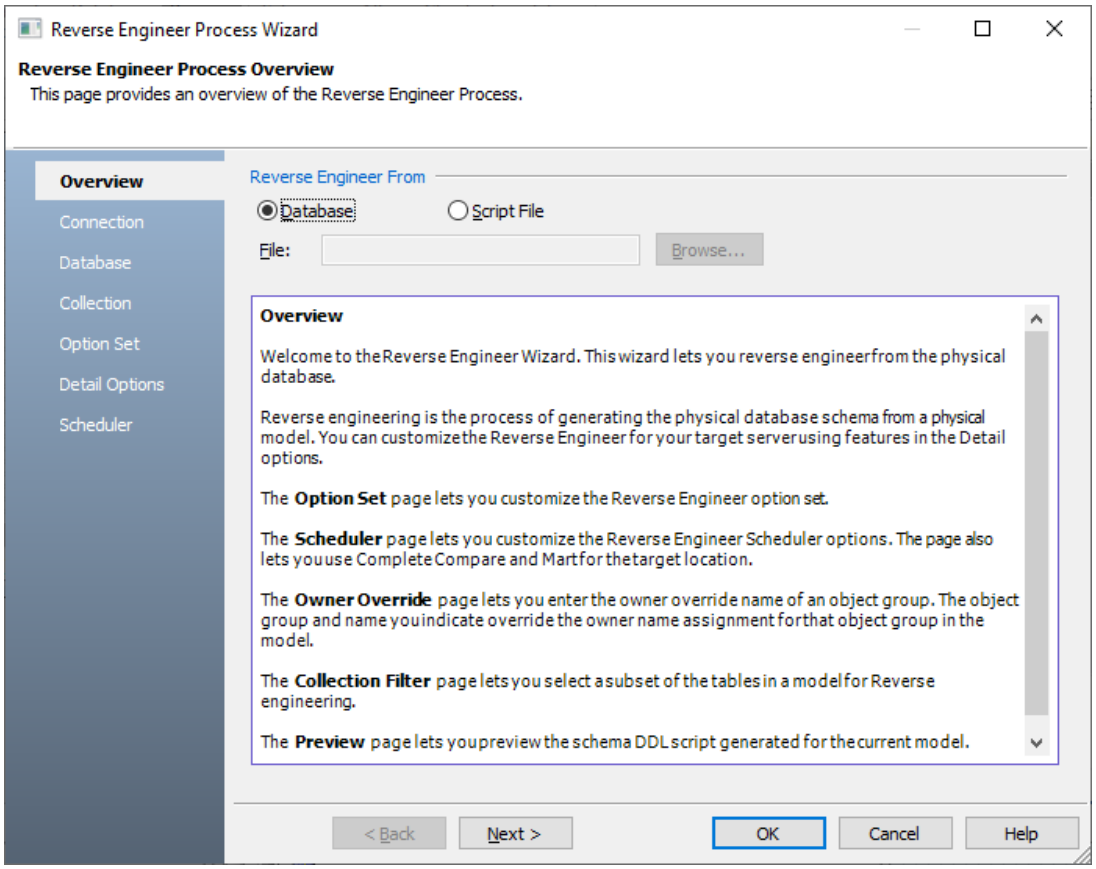
3. Click **Database/Script**.
   By default, the Allow Demand Loading option is selected.



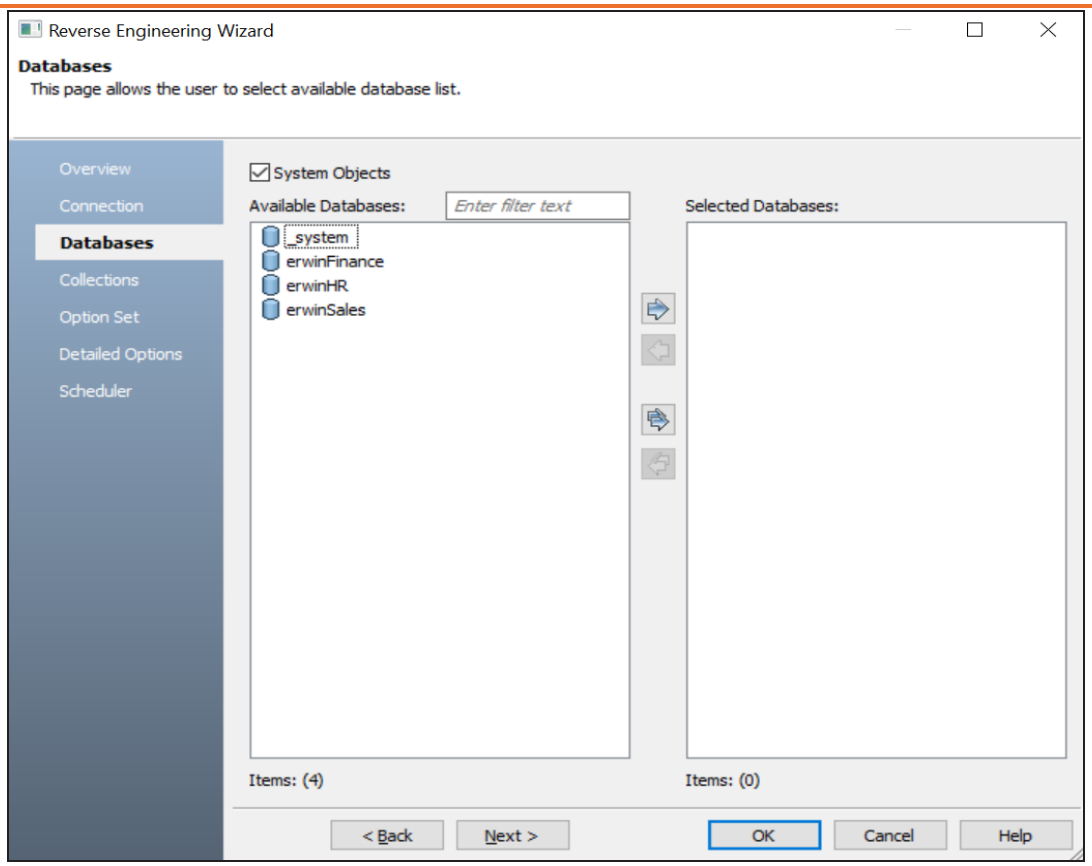
4. Click **Load**.
   The New Model dialog box appears. This starts the reverse engineering process to pull a model from the database to compare.



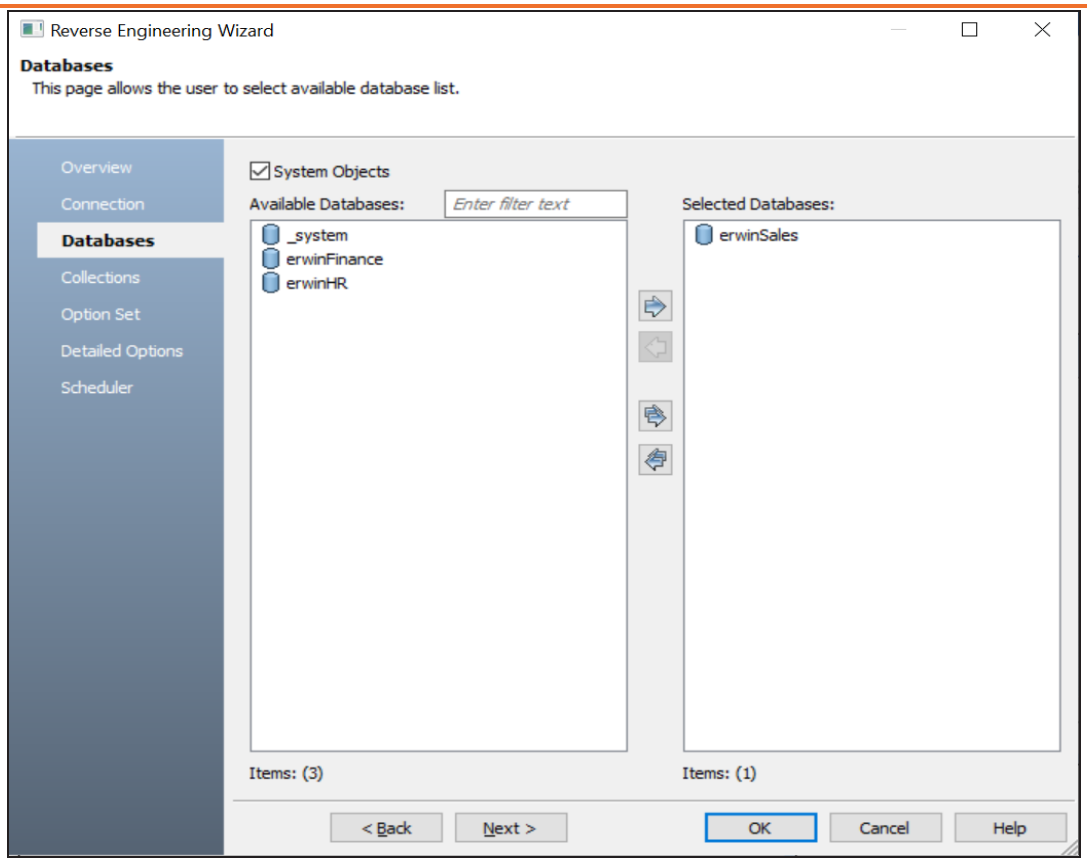
5. Ensure that the Database is set to the correct one. In this case, ArangoDB. Then, click **Next**.
   The Reverse Engineer Process Wizard appears.



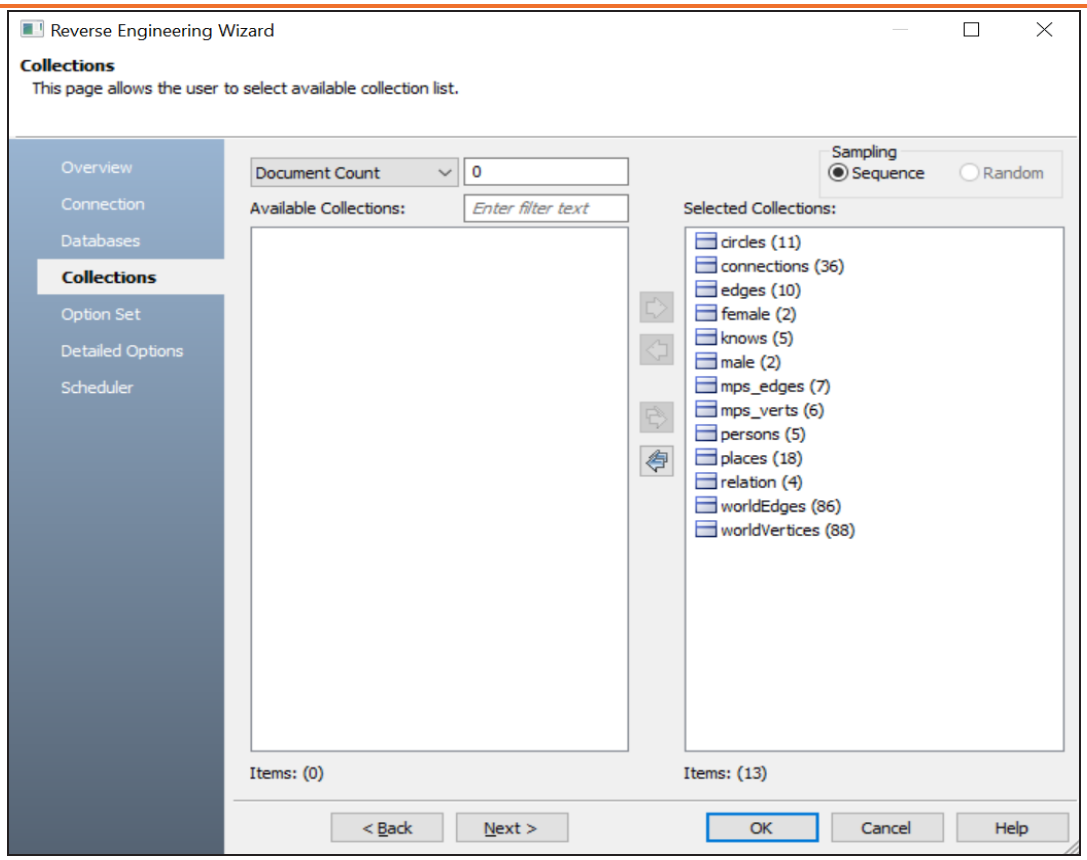
6. Click **Database**. Then, click **Next**.
   The Connection tab appears. Use this tab to connect to the database from which you want to reverse engineer the model.
7. After connection is established, click **Next**.
   The Databases tab appears. It displays a list of available databases.



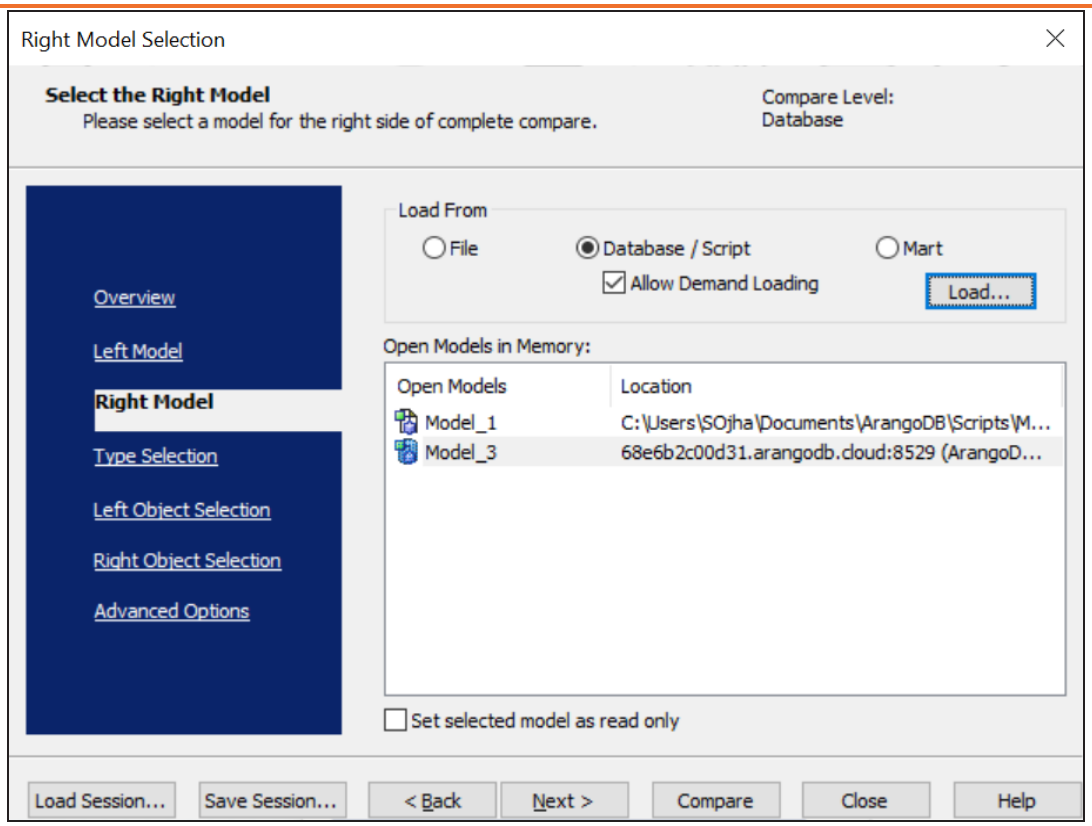
8. Under **Available Databases**, select the databases that you want to reverse engineer. Then, click  .
   This moves the selected databases under Selected Databases.



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

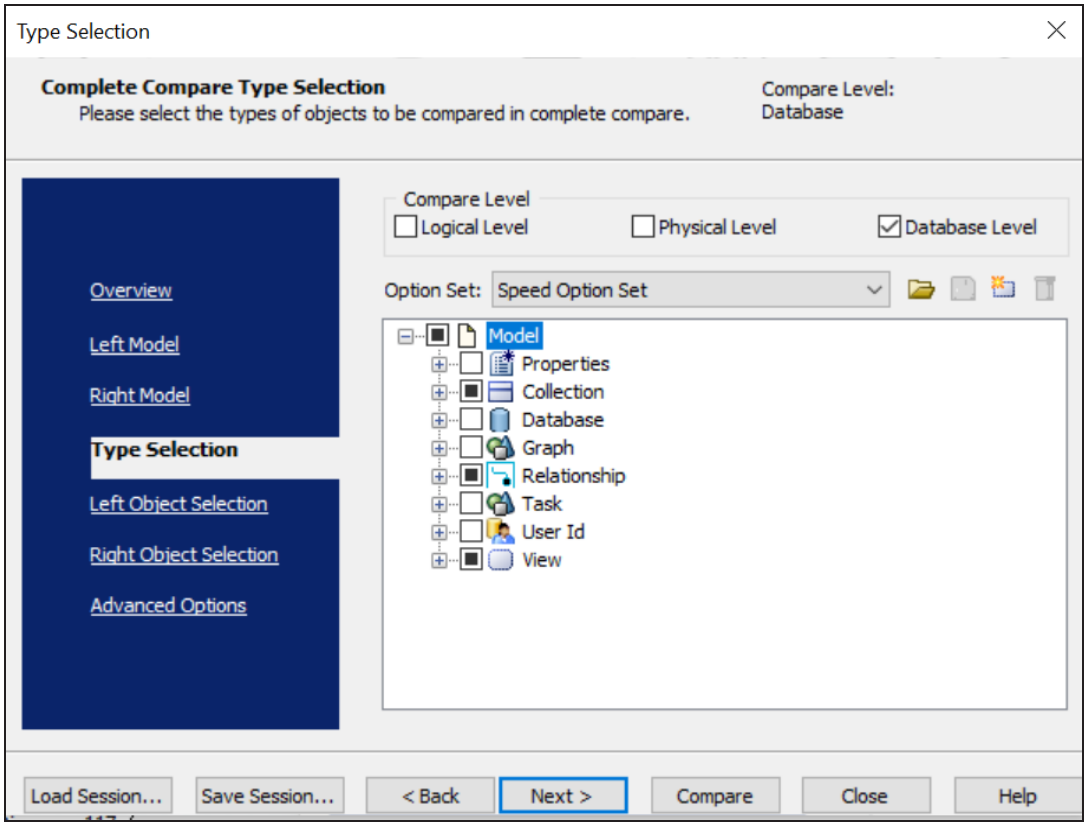


10. Click **Next** and on the Option Set tab, keep the default configuration.
11. Click **Next** and on the Detailed Options tab, keep the default configuration.
12. Click **OK**.
    The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse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



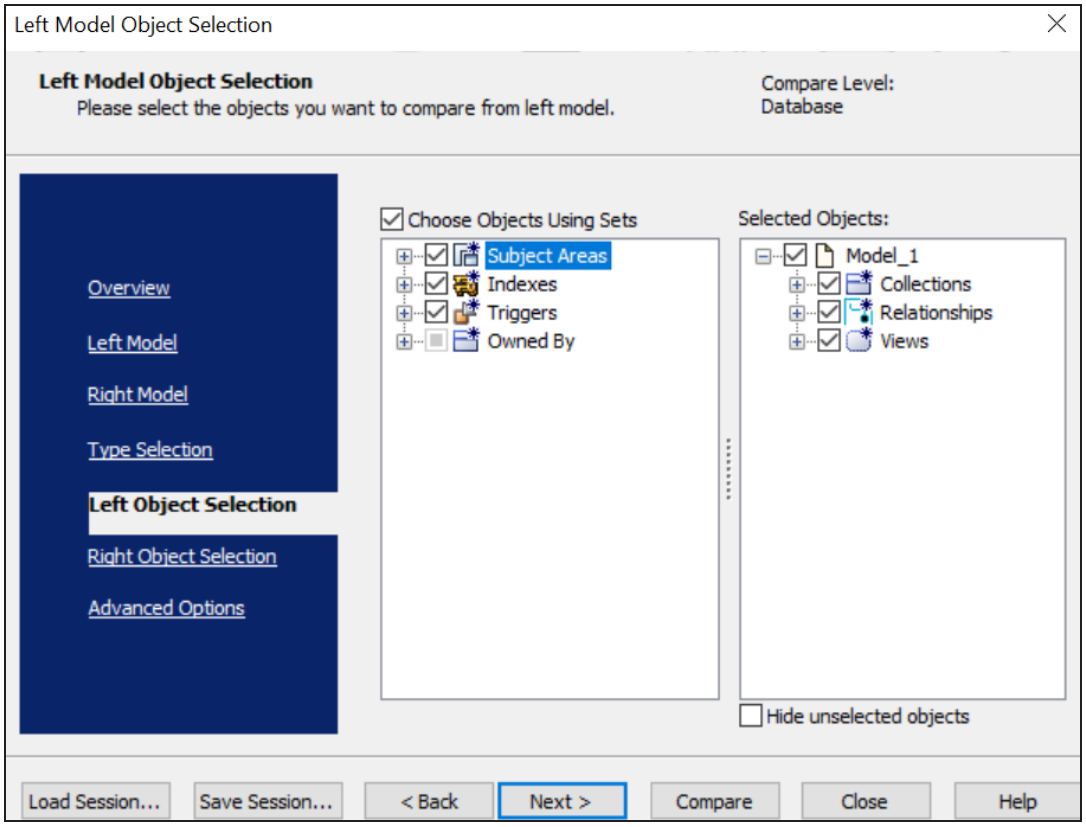
13. Click **Next** and on the Type Selection tab, select the appropriate options.

For example, the following image shows the default options.



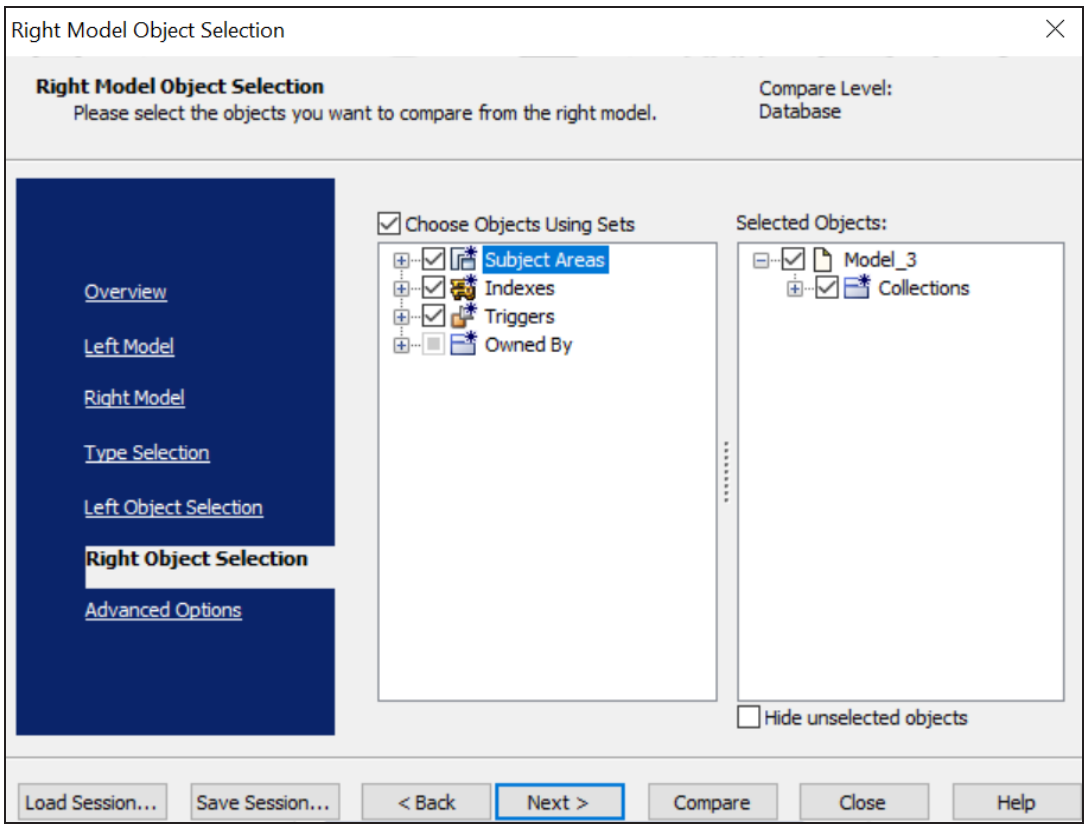
14. Click **Next** and on the Left Object Selection tab, select the appropriate options.

For example, the following image shows the default options.

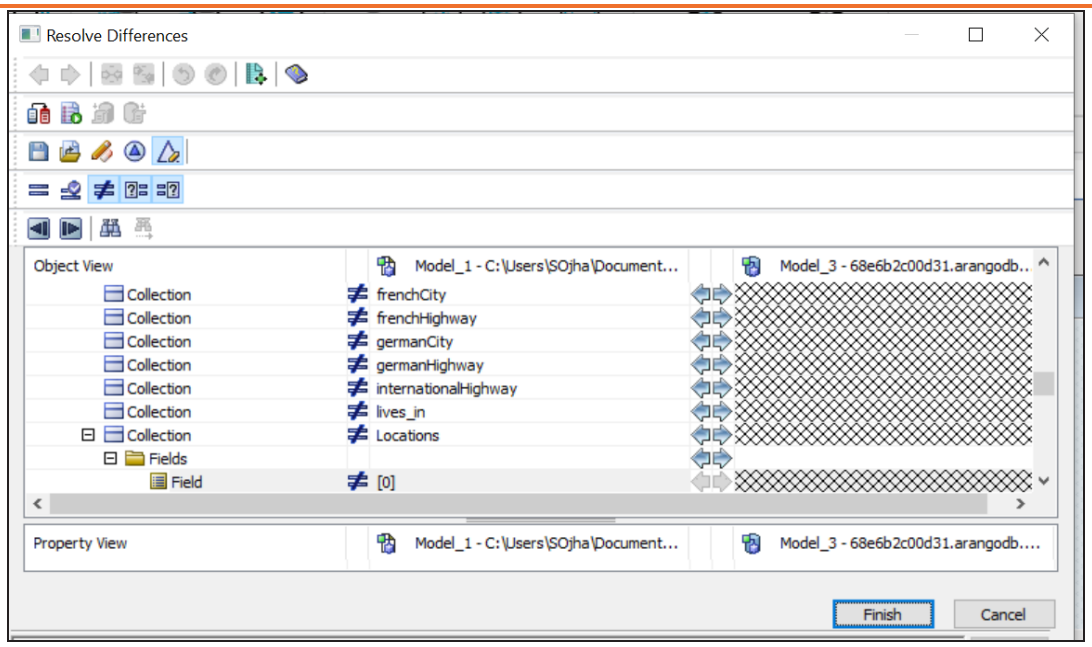


15. Click **Next** and on the Right Object Selection tab, select the appropriate options.

For example, the following image shows the default options.



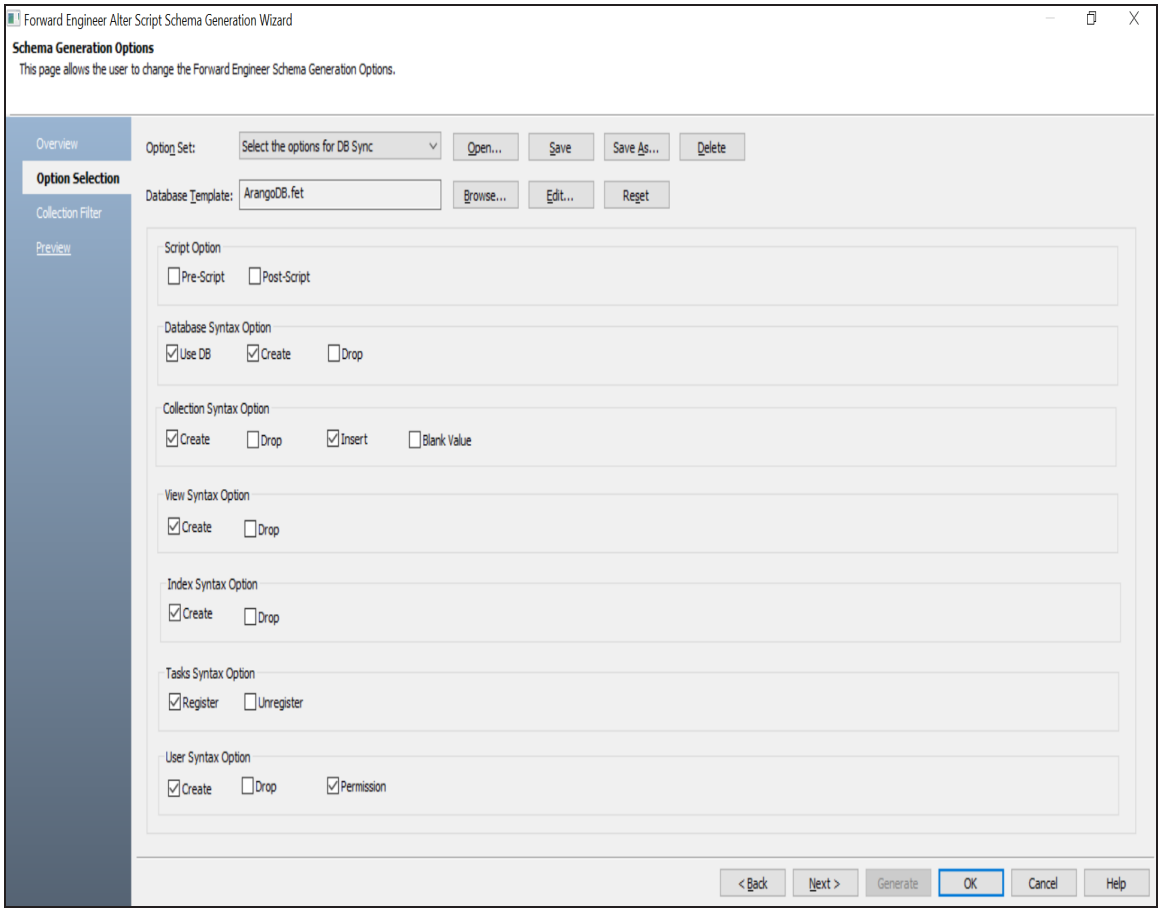
16. Click **Compare**.
    The comparison process runs, and the Resolve Differences dialog box appears. It displays the differences between your model and database.
    For example, the following image shows that the frenchCity collection is available in your model but not in the database.



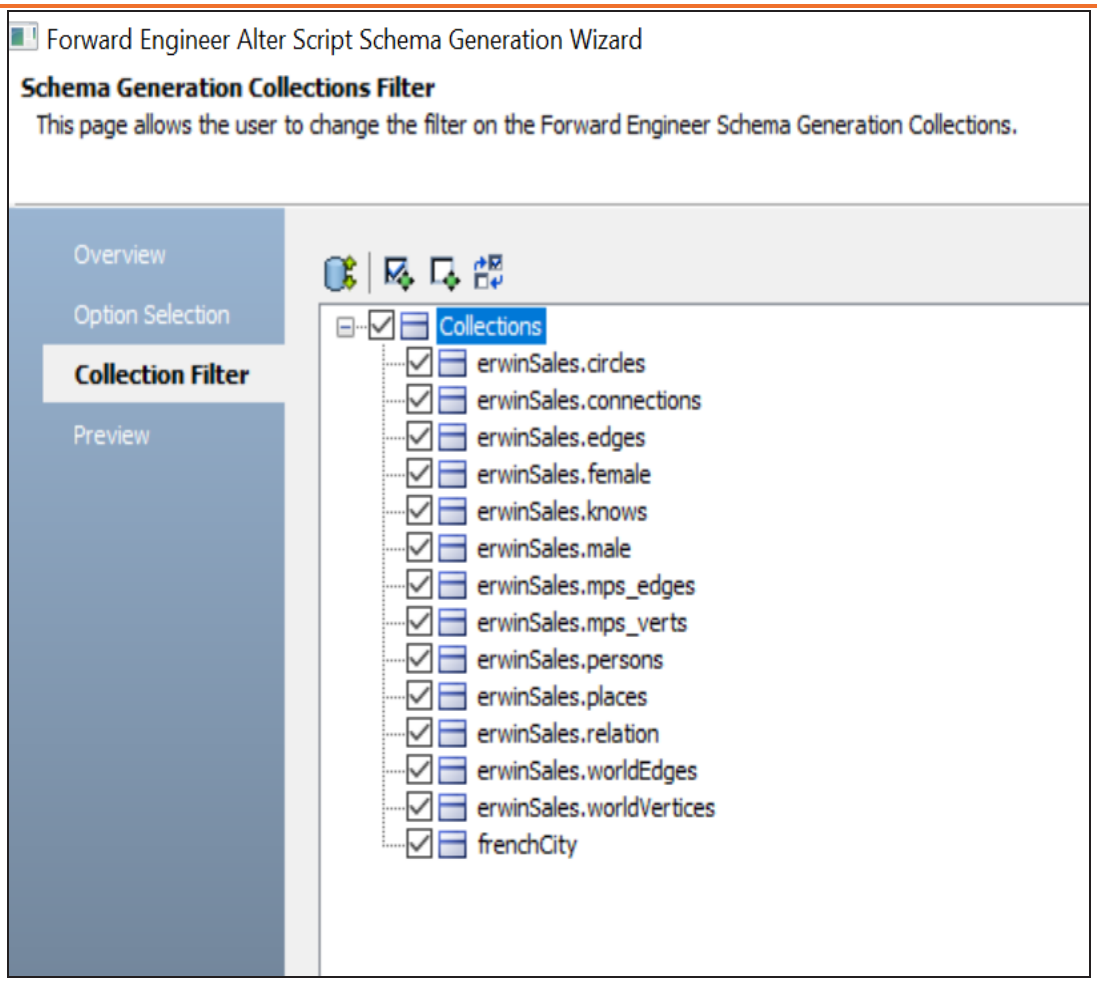
Select the frenchCity collection and click . This will move the frenchCity collection to the right model (from the database). Similarly, resolve other differences.

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 ArangoDB 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**.

Migrating Relational Models to ArangoDB Models

You can migrate your relational models to ArangoDB 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 an ArangoDB model.

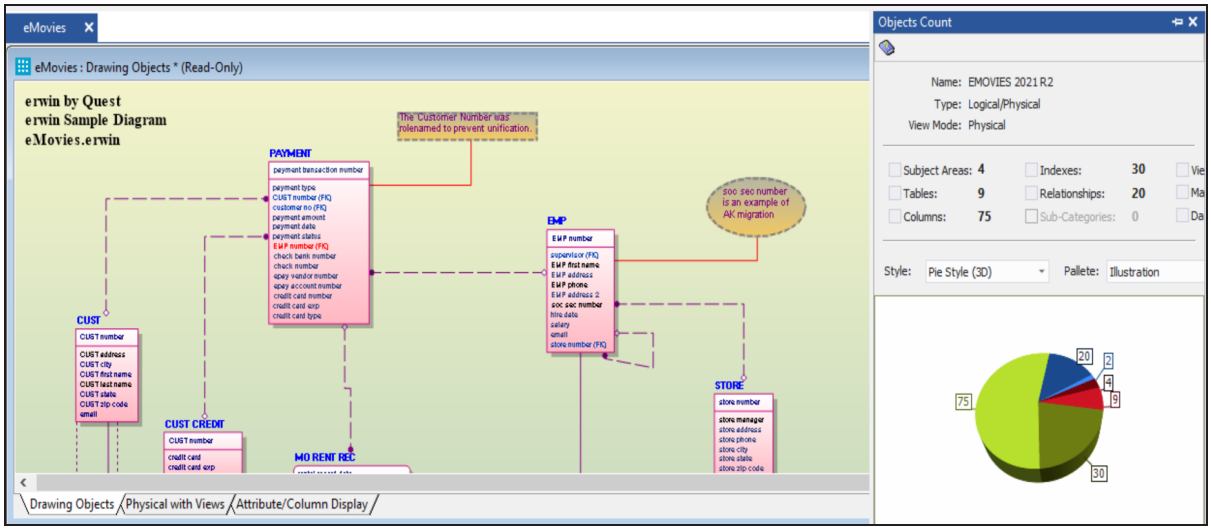
Migration by Changing the Target Database

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

1. Open your relational model in erwin Data Modeler (DM).

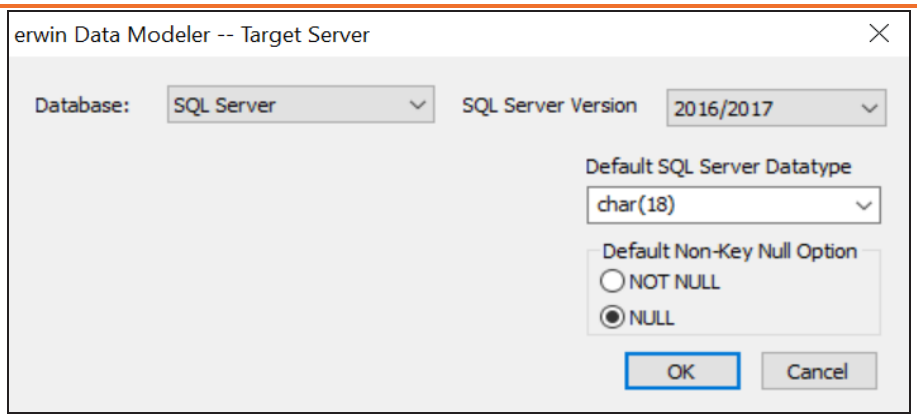
   Ensure that you are in the Physical mode.

   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.

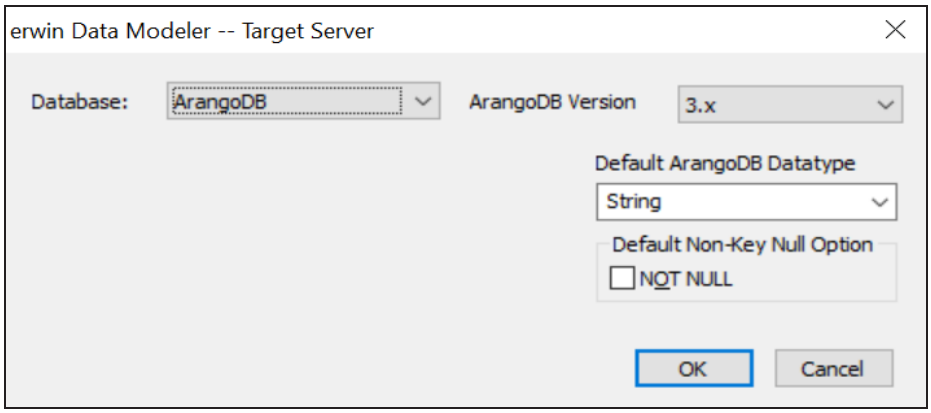


2. On the ribbon, click **Actions** > **Target Database** or on the status bar, click the database name.

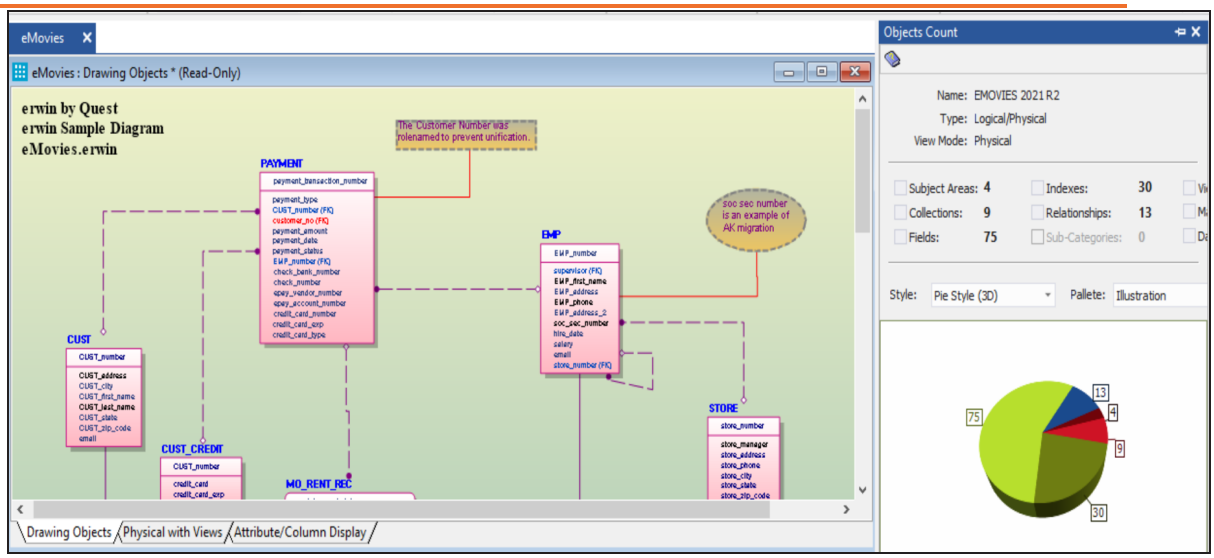
   The erwin Data Modeler -- Target Server screen appears.



3. In the **Database** drop-down list, select ArangoDB.



4. Click **OK**.
   The conversion process starts.
   Once the conversion is complete, the existing model in migrated to an ArangoDB database.



In the **Objects Count** pane, note that instead of tables and columns, we now have collections and fields. The migration process converts and merges multiple tables, columns, and relationships to the ArangoDB format.

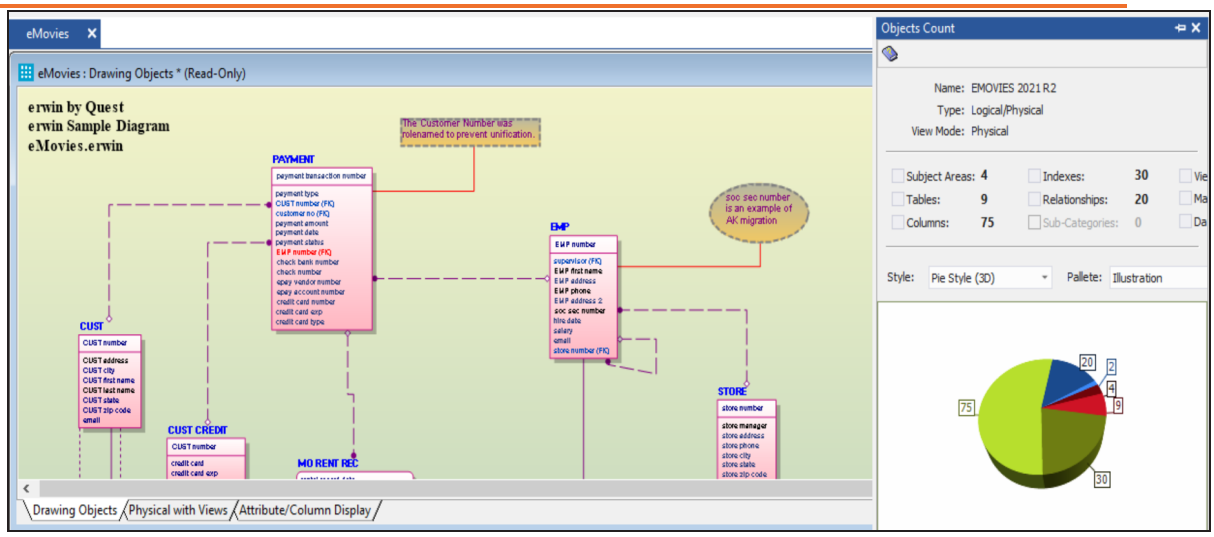
Migration by Deriving a Model

To migrate by deriving a model, follow these steps:

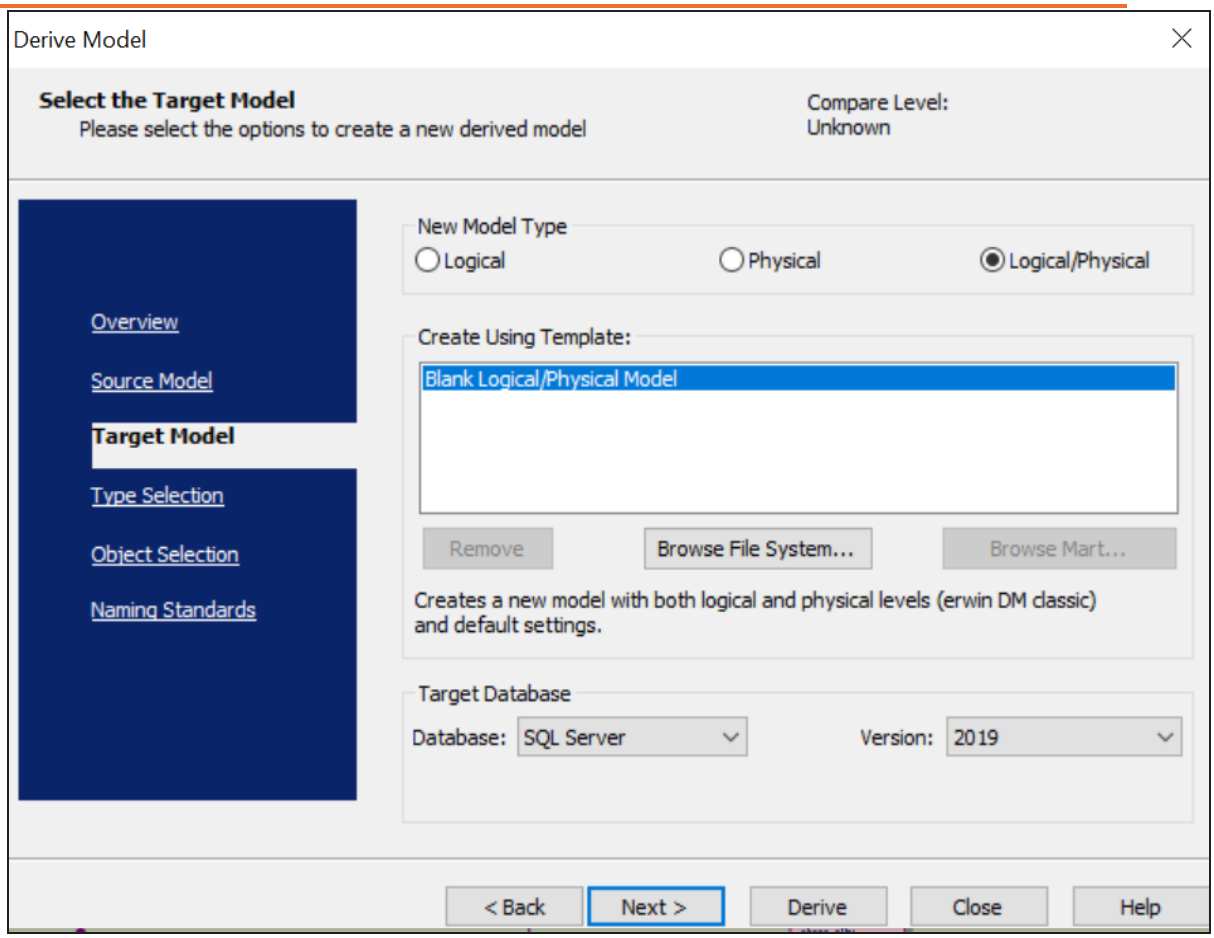
1. Open your relational model in erwin Data Modeler (DM).

   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.



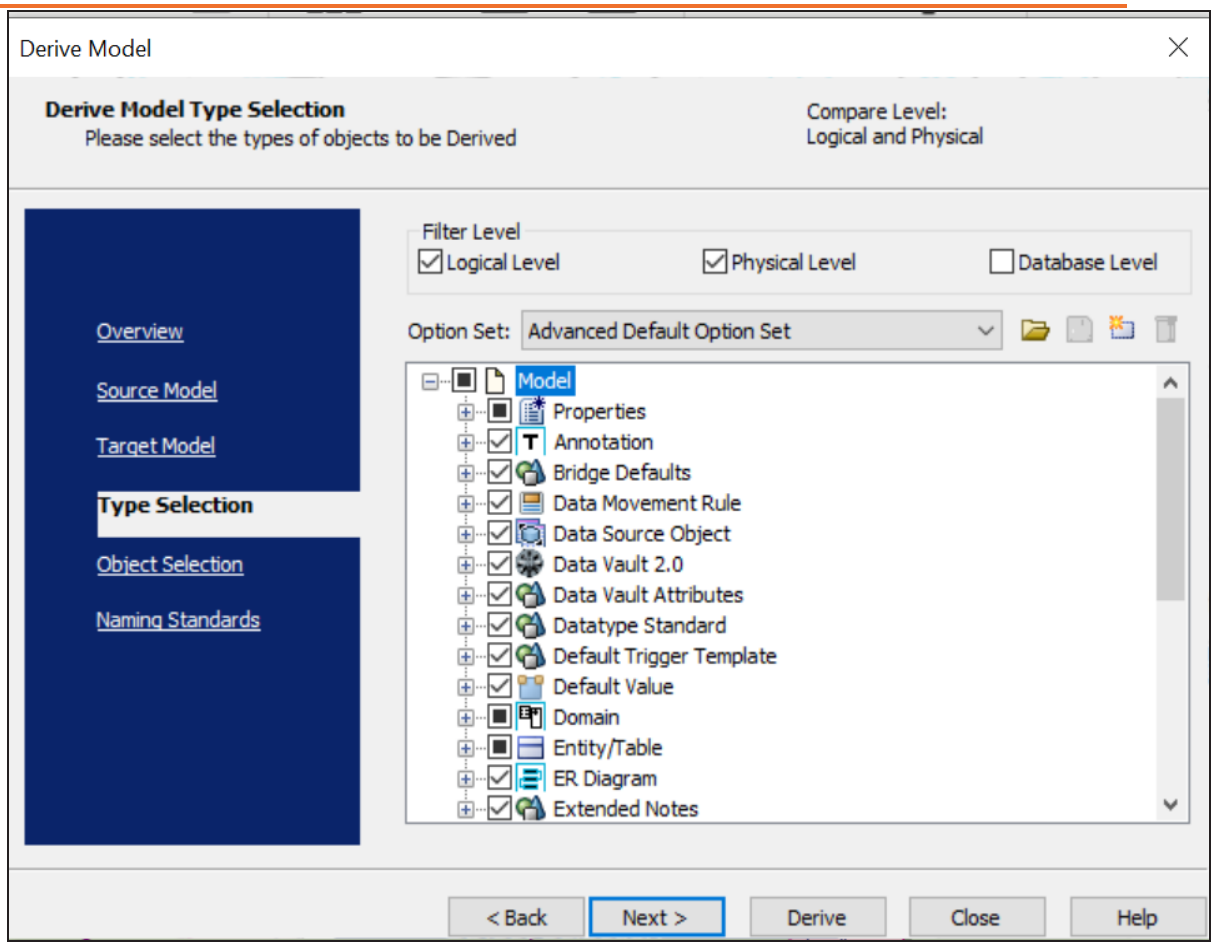
2. On the ribbon, click **Actions** > **Design Layers** > **Derive New Model**.
   The Derive Model screen appears. By default, the Source Model is set to your current model.



3. In the **Database** drop-down list, select **ArangoDB**.
4. Click **Next**.

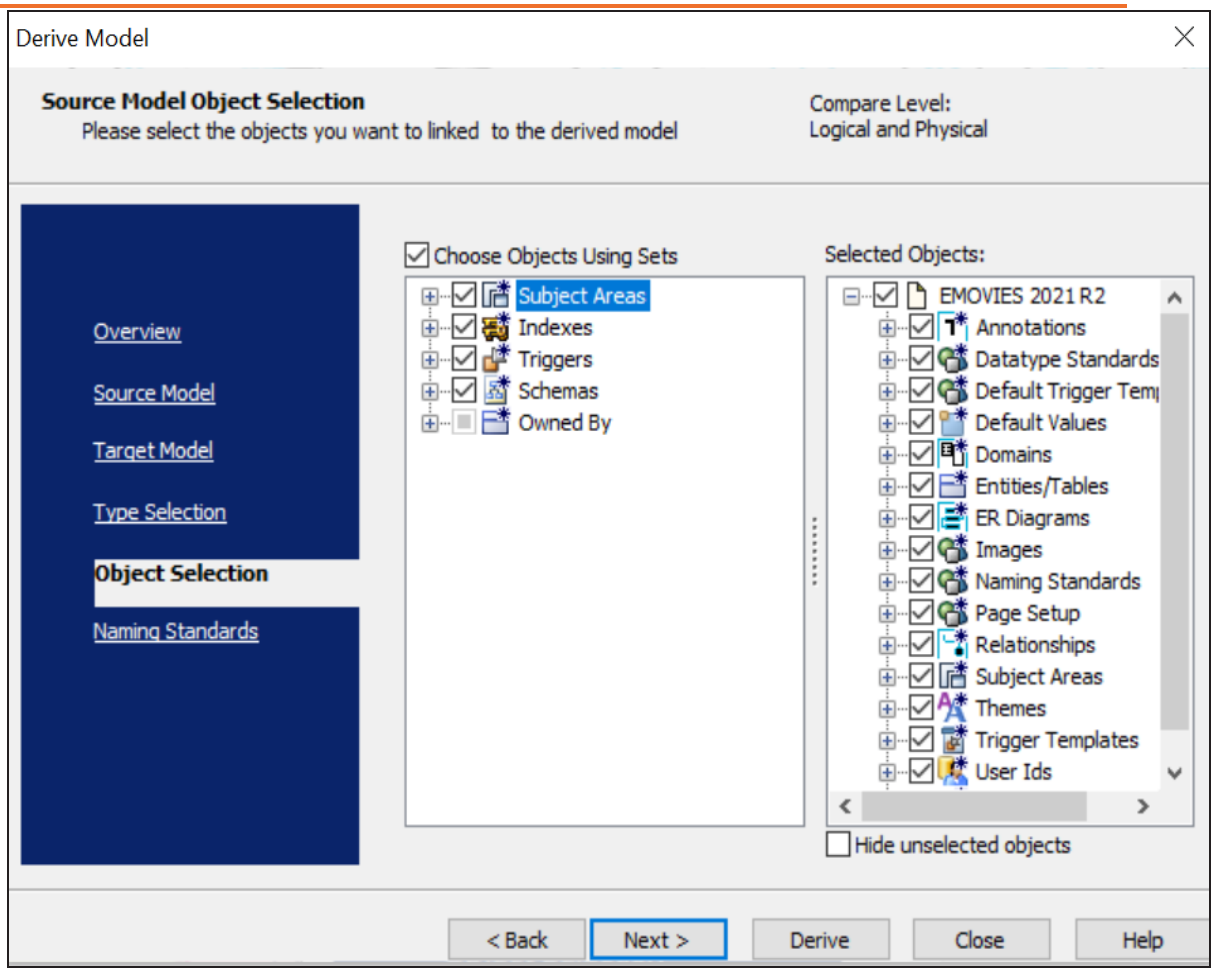
> If the Type Resolution screen appears, click **Finish**.

The Type Selection section appears.



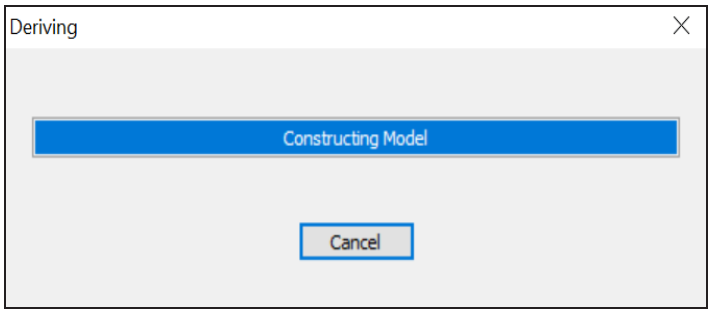
5. Select the types of objects that you want to derive into the target ArangoDB model.

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

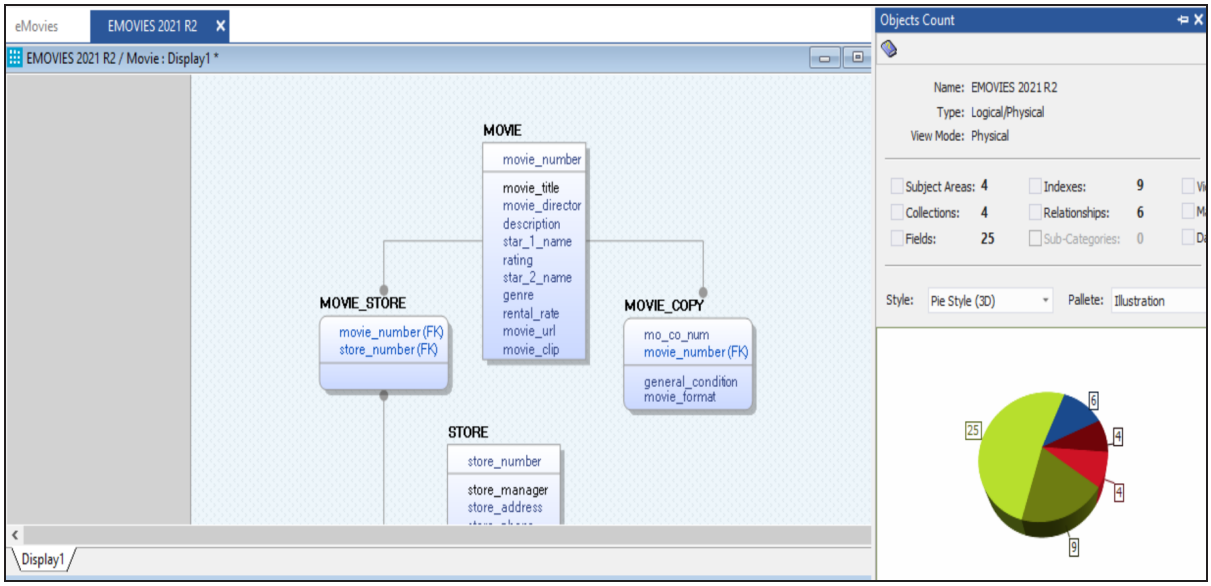


7. Select the objects that you want to derive into the target ArangoDB model.

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. The migration process converts and merges multiple tables, columns, and relationships to the ArangoDB format.

Amazon Keyspaces Support

erwin Data Modeler (DM) now supports Amazon Keyspaces as a target database. This implementation supports the following objects:

- Keyspaces
- Tables
  - Columns
  - Indexes

Following are the supported data types:

- ASCII
- BIGINT
- BLOB
- BOOLEAN
- COUNTER
- DATE
- DECIMAL
- DOUBLE
- FLOAT
- INET
- INT
- LIST
- MAP
- SET
- SMALLINT
- TEXT
- TIME

- TIMESTAMP
- TIMEUUID
- TINYINT
- TUPLE
- UUID
- VARCHAR
- VARINT

Amazon Keyspaces implementation supports all erwin DM features and functions. The following sections walk you through these features:

- Reverse engineering models from database and script
- Forward engineering models to database
- Comparing changes using Complete Compare
- Converting relational models to Amazon Keyspaces models

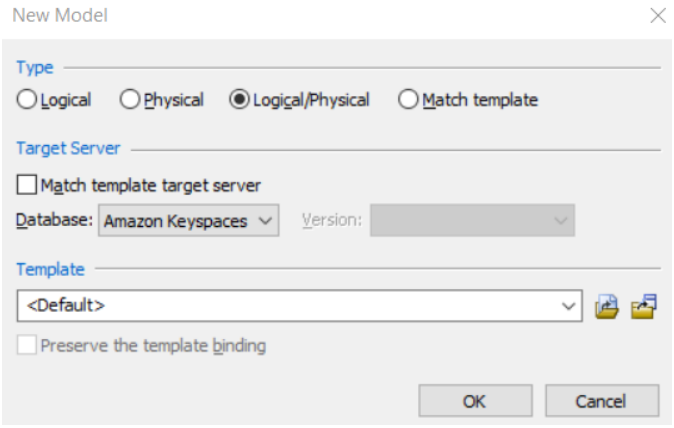
Reverse Engineering Models

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

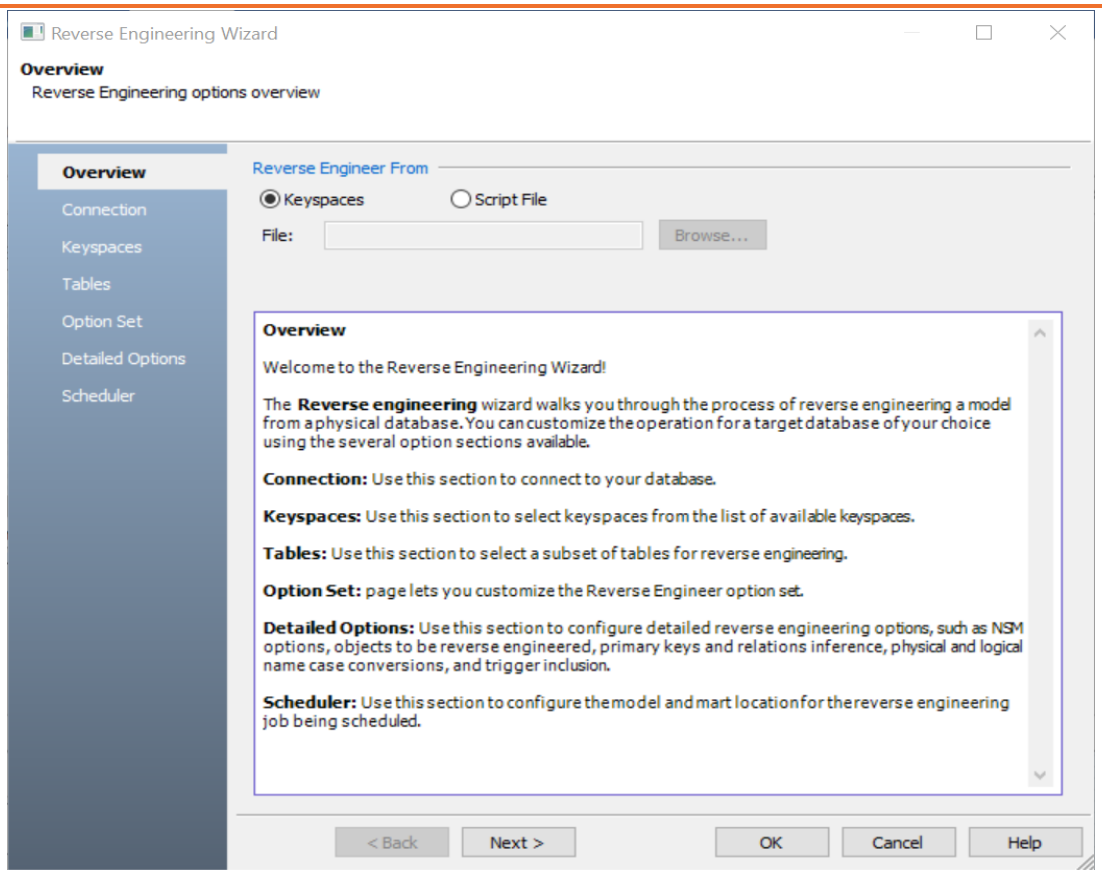
This topic walks you through the steps to reverse engineer an Amazon Keyspaces model. For detailed description of reverse engineering options, refer to the Reverse Engineering Options topic.

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 Amazon Keyspaces.



3. Click **Next**.
   The Reverse Engineering Wizard appears.



4. Click one of the following options:

    - **Keyspaces**: Use this option to reverse engineer a model from your database.

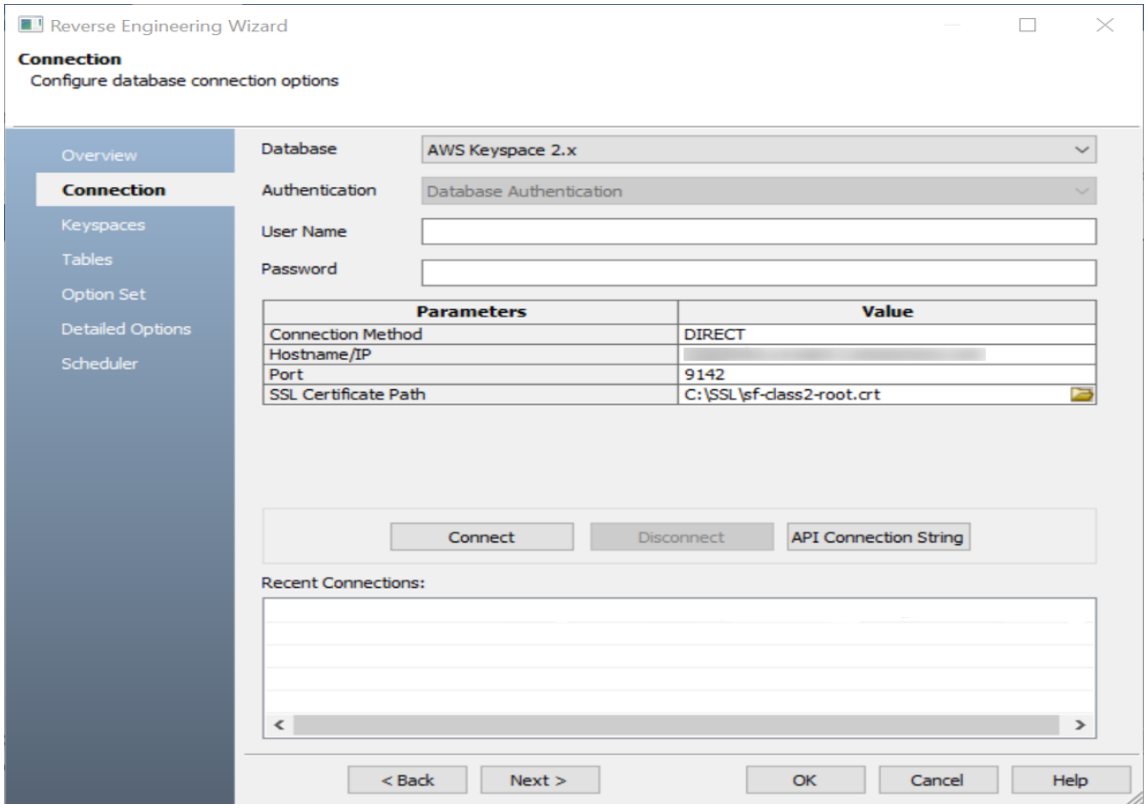
      If you click **Keyspaces**, continue to step 5.

    - **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.

      If you click **Script File**, see step 13 below.

5. Click **Next**.

The Connection tab appears.



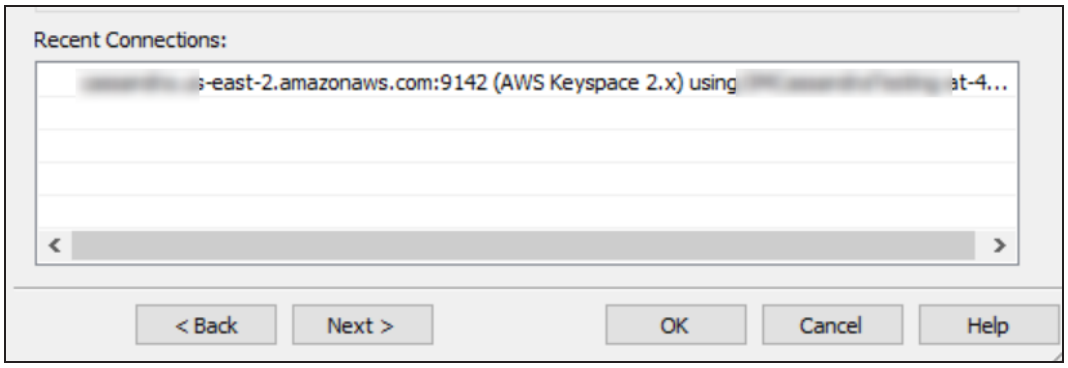
6. Enter your **User Name** and **Password**.

   The following table explains the connection parameters.

Parameter	Description	Additional Information
Connection Method	Specifies the type of connection you want to use. Select **Direct** to connect to your database directly.	
Hostname/IP	Specifies the hostname or IP address of the server where your database is hosted in the following format: *cassandra.<region>.amazonaws.com*	For example, *cassandra.us-east-2.amazonaws.com* This option is available when Connection Method is set to Direct.

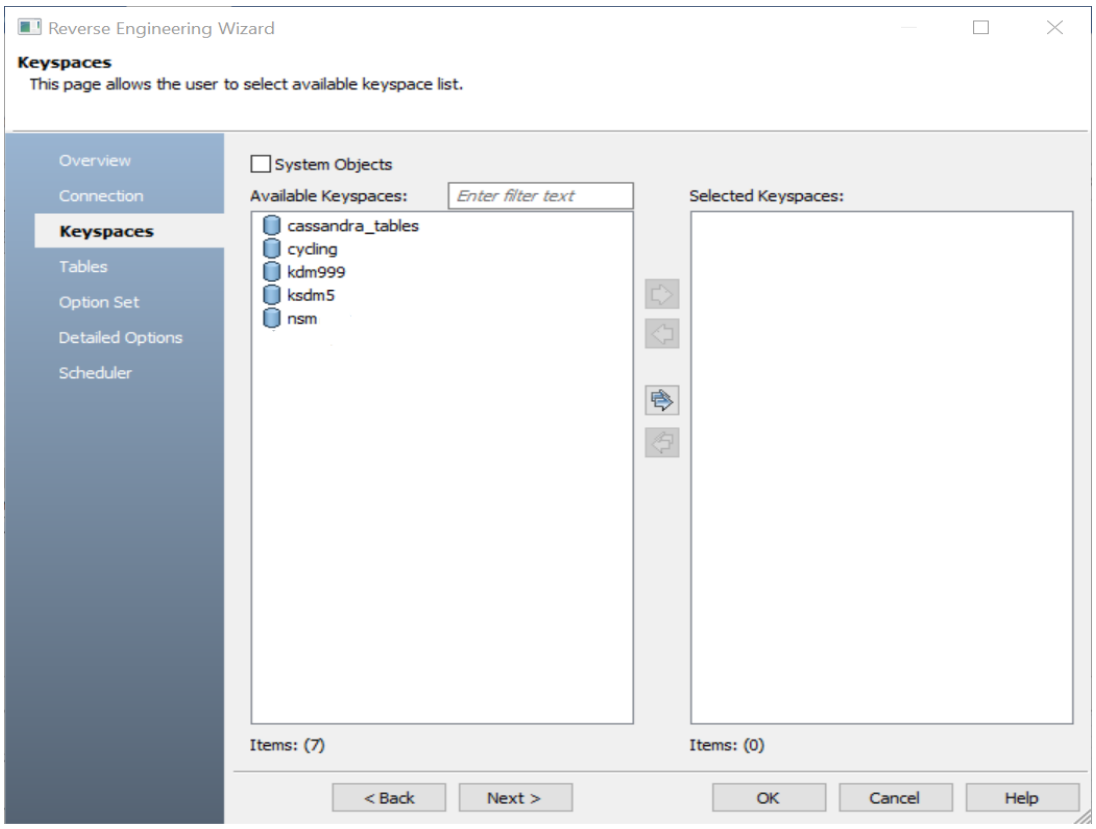
Port	Specifies the port configured for your database	For example, *9142* This option is available when Connection Method is set to Direct.
SSL Certificate Path	Specifies the path to the SSL certificate in the following format: *C:\<file name>.crt*	For example, *C:\SSL\sf-class2-root.crt* This option is available when Connection Method is set to Direct.

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

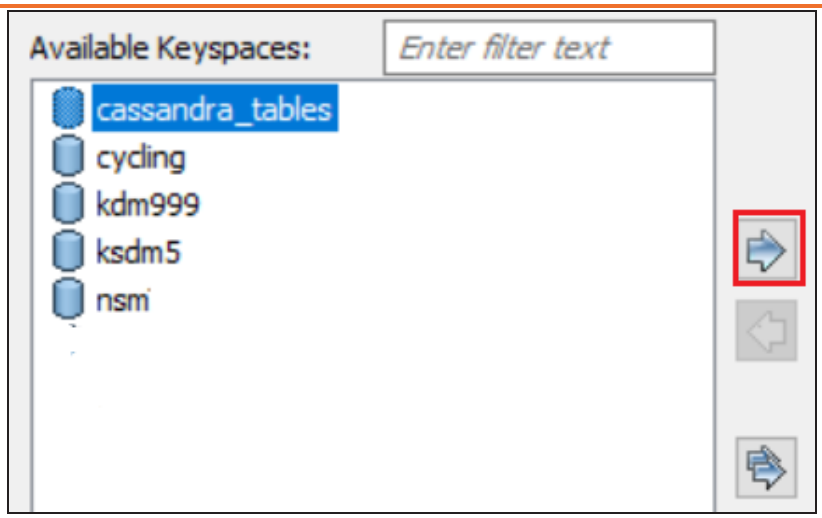


8. Click **Next**.

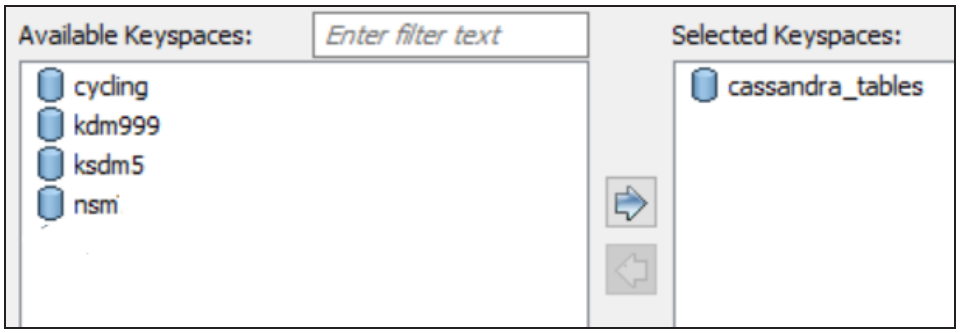
The Keyspaces tab appears. It displays a list of available keyspaces.



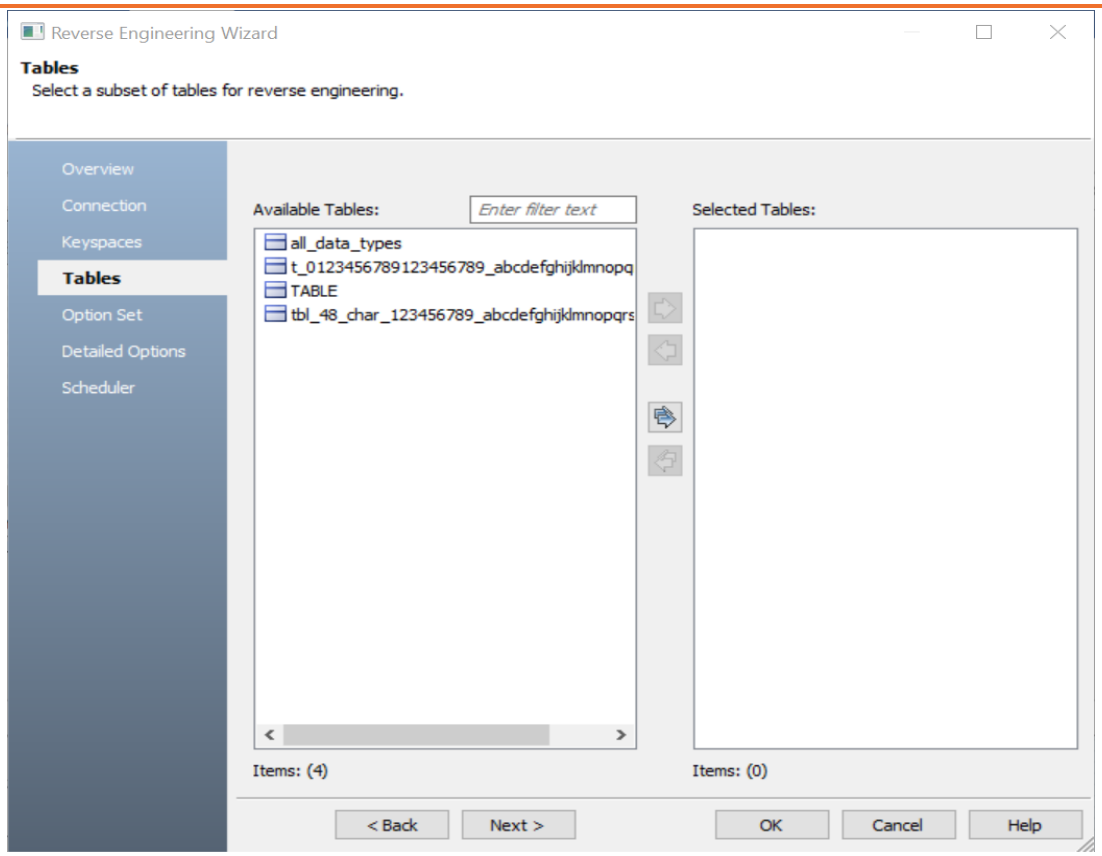
9. Under **Available Keyspaces**, select the keyspaces that you want to reverse engineer. Then, click  .



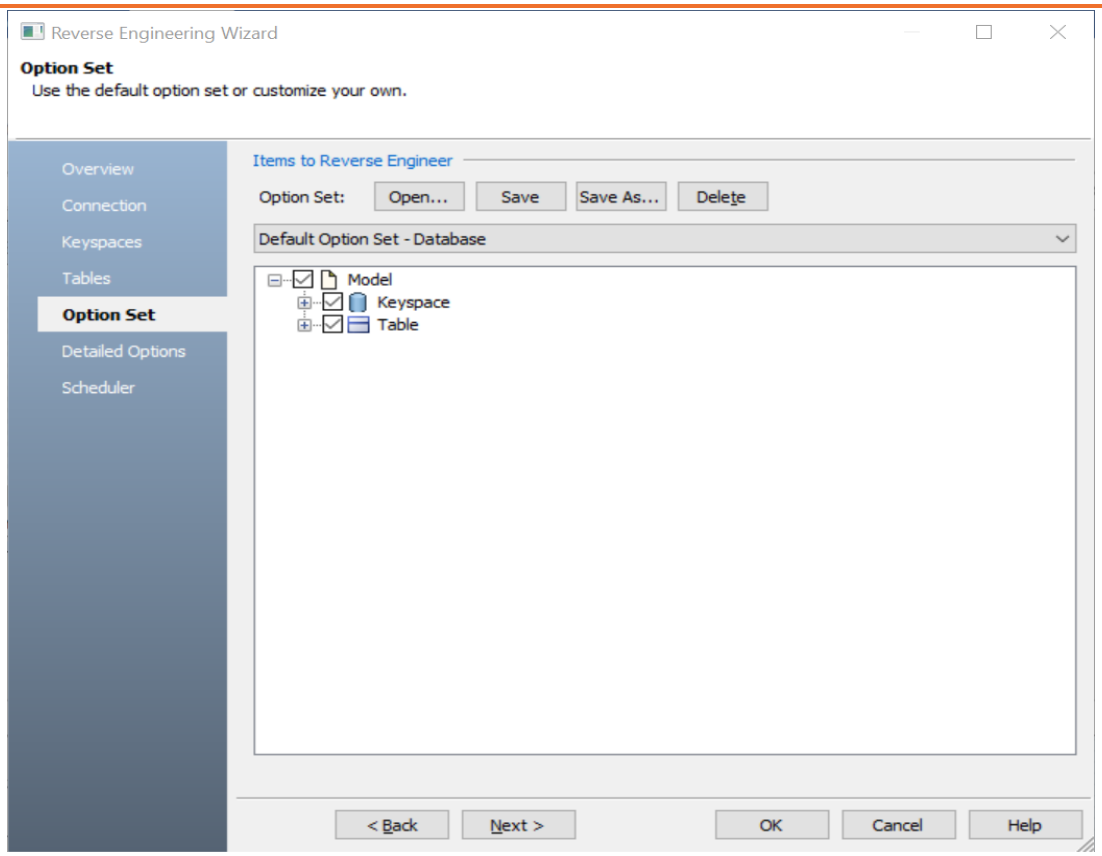
This moves the selected databases under Selected Keyspaces.



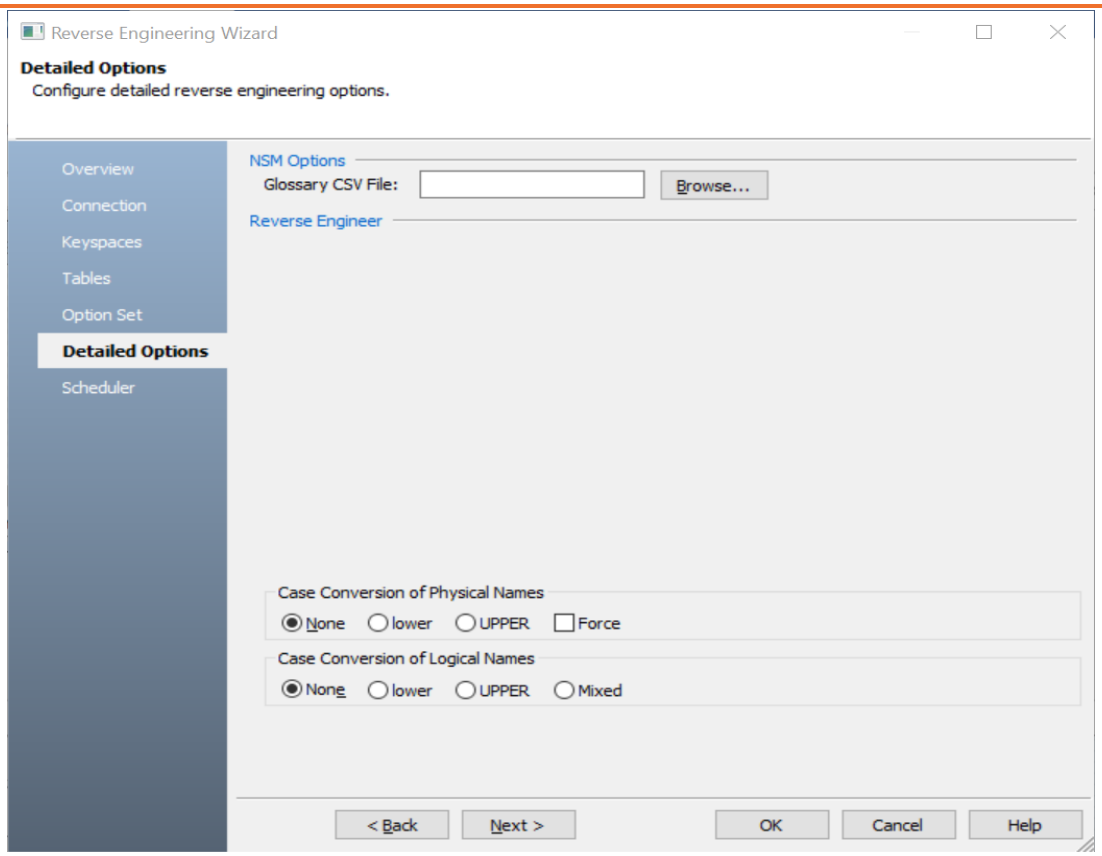
10. Click **Next**.
    The Tables tab appears. It displays a list of available tables in the keyspaces that you selected in step 8.



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

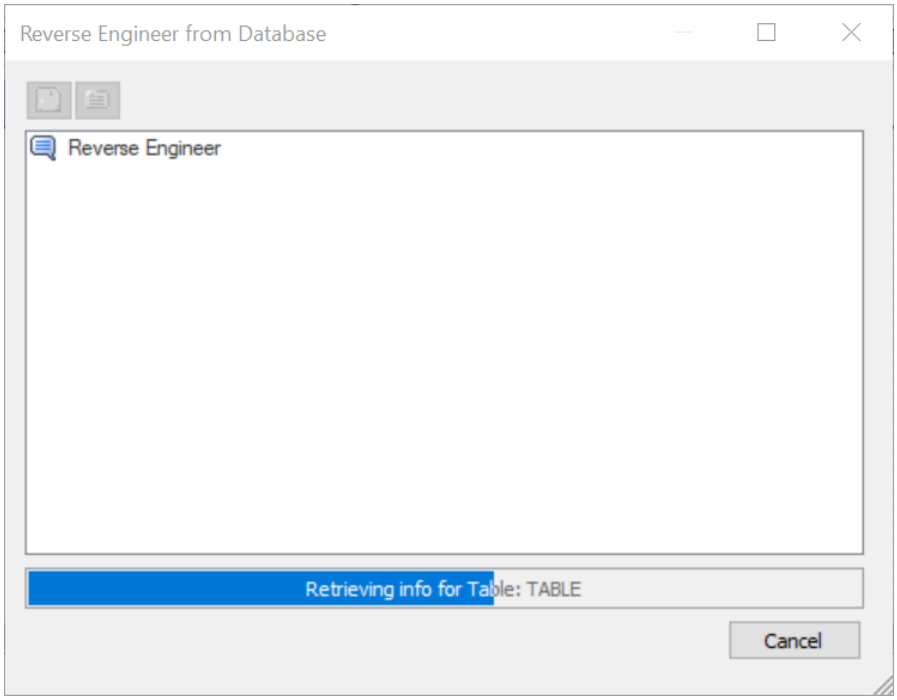


12. Click **Next**.
    The Detail Options tab appears. Set up appropriate options based on your requirement.

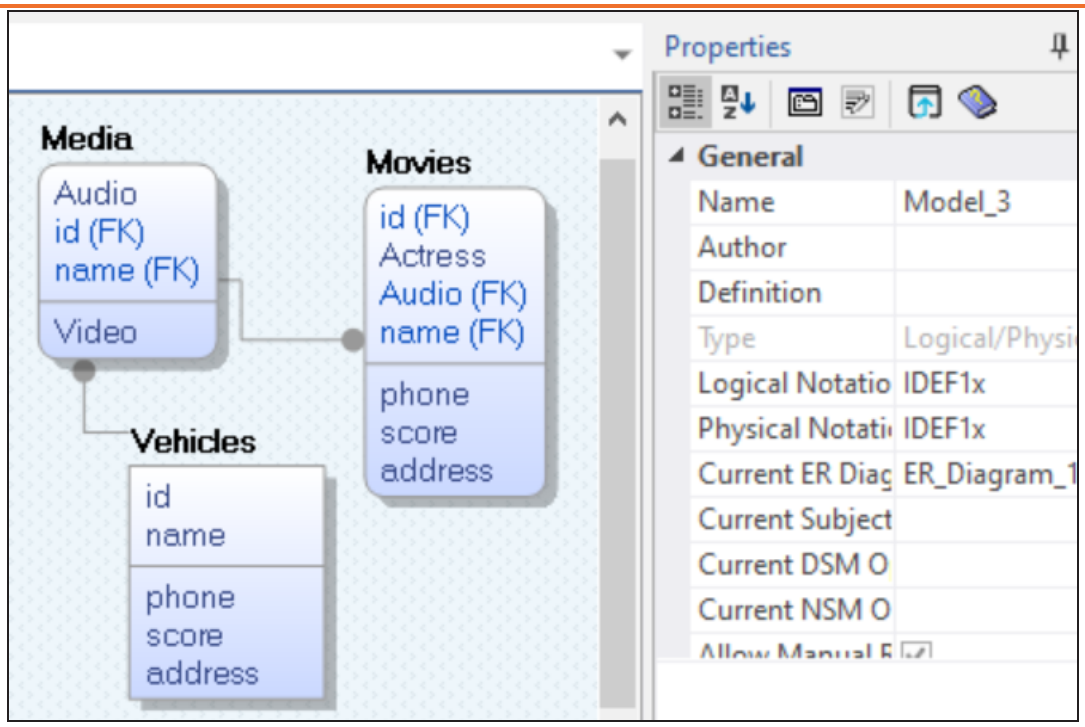

Reverse Engineering Wizard
Detailed Options
Configure detailed reverse engineering options.
Overview
Connection
Keyspaces
Tables
Option Set
Detailed Options
Scheduler
NSM Options
Glossary CSV File:
Browse...
Reverse Engineer
Case Conversion of Physical Names
None
lower
UPPER
Force
Case Conversion of Logical Names
None
lower
UPPER
Mixed
< Back
Next >
OK
Cancel
Help

13. Click **OK**.

The reverse engineering process starts.

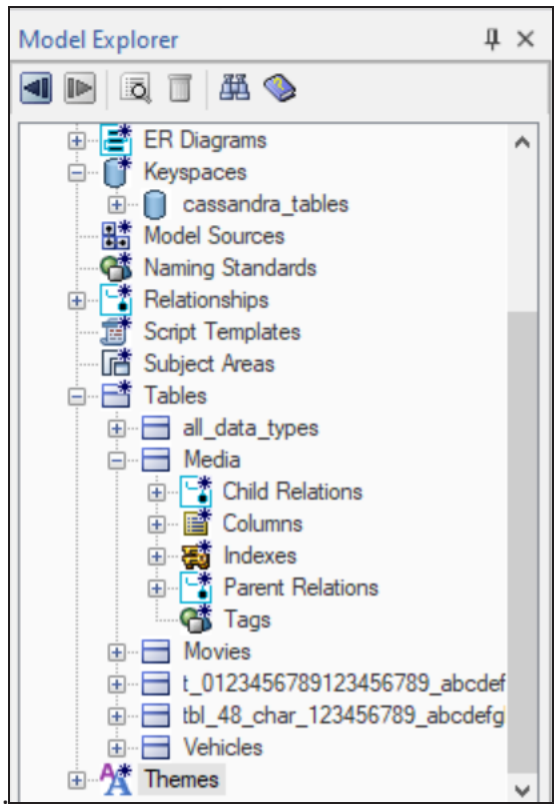


Once the process is complete, based on your selections, a schema is generated and a model is created.



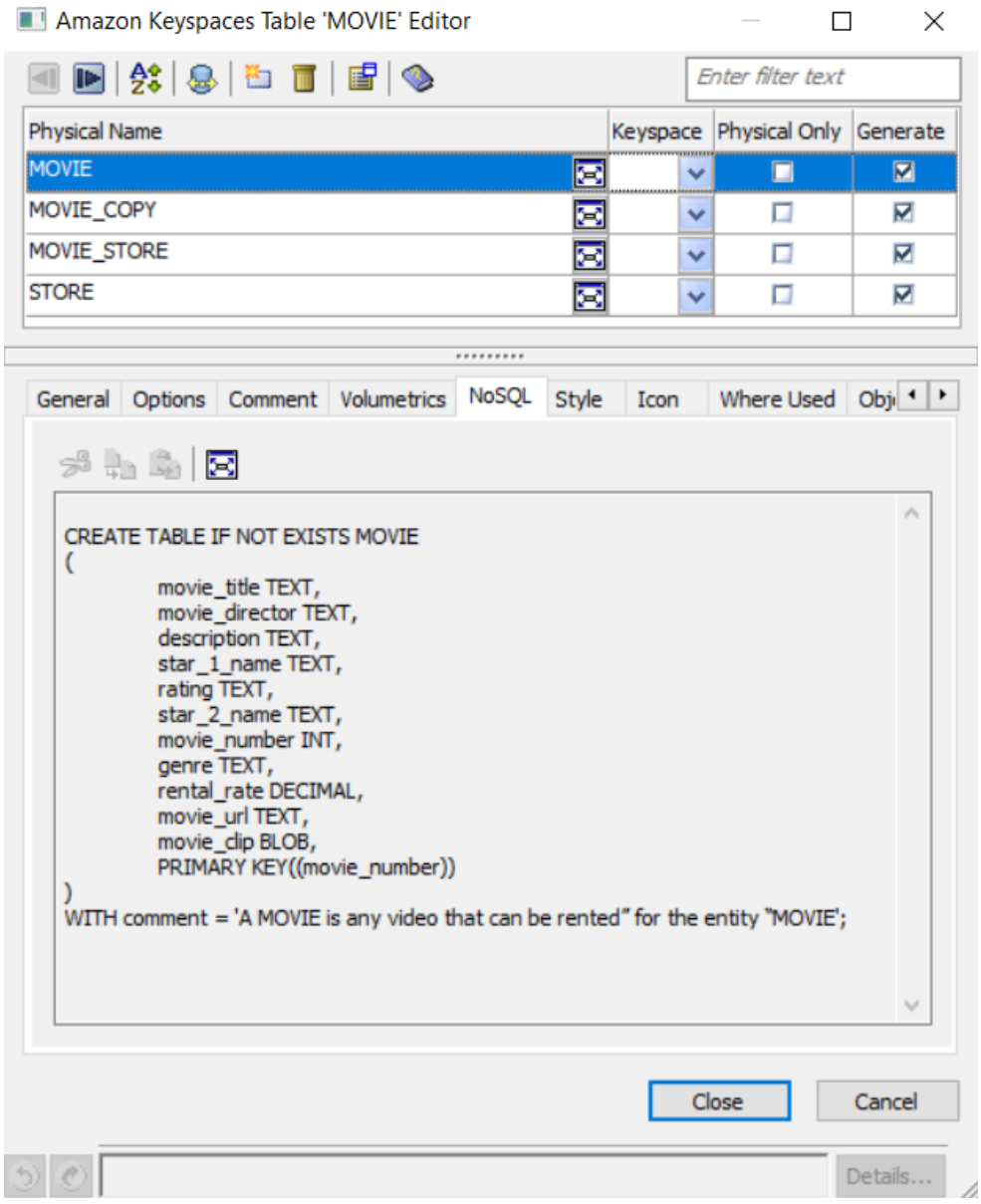
You can edit the shape of the nodes to look like the standard table-like structure. On the ribbon click **View** > **Field**. You can also change label color, size, and caption using the properties pane.

Along with Keyspaces and Tables, other object such as Columns and Indexes are also retrieved.



You can view these objects via the model diagram or view their properties via the Model Explorer. Right-click an object and then, click the required Properties option. For example, on the model diagram, right click a table and then, click Table Properties. The Amazon Keyspaces table editor appears. You can view the table's CREATE statement on the No SQL tab, as seen , the table, Media has four properties, audio, video, id, and name to

store additional information.



Reverse Engineering Options for Amazon Keyspaces

The following are the reverse engineering options for Amazon Keyspaces in erwin DM.

Overview

Parameter	Description	Additional Information
Reverse Engineer From	Specifies whether you want to reverse engineer from a script or keyspace	**Keyspaces**: Indicates that the model is reverse engineered from keyspaces **Script File**: Indicates that the model is reverse engineered from a script
File	Specifies the script file location	This option is available when Script File is selected.

Connection

Parameter	Description	Additional Information
Connection Method	Specifies the type of connection you want to use. Select **Direct** to connect to your database directly.	
Hostname/IP	Specifies the hostname or IP address of the server where your database is hosted in the following format: *cassandra.<region>.amazonaws.com*	For example, *cassandra.us-east-2.amazonaws.com* This option is available when Connection Method is set to Direct.
Port	Specifies the port configured for your data-base	For example, *9142* This option is available when Connection Method is set to Direct.
SSL Certificate Path	Specifies the path to the SSL certificate in the following format:	For example, *C:\SSL\sf-class2-root.crt*

	C:\<file name>.crt	This option is available when Connection Method is set to Direct.

Keyspaces

Parameter	Description	Additional Information
System Objects	Specifies whether system objects are available under Available Keyspaces	
Available Keyspaces	Specifies a list of available keyspaces	
Selected Keyspaces	Specifies a list of selected keyspaces for reverse engineering	

Tables

Parameter	Description	Additional Information
Available Tables	Specifies a list of available tables	
Selected Tables	Specifies a list of selected tables for reverse engineering	

Option Sets

Parameter	Description	Additional Information
Option Set	Specifies the option set template for reverse engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save the configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at some external location.

		Delete: Use this option to delete an option set.
<Option Set Name>	Specifies the objects to be reverse engineered according to the selected option set. You can edit this list.	

Detailed Options

Parameter	Description	Additional Information
NSM Options	Specifies the naming standard glossary file in the .CSV format	
Case Conversion of Physical Names	Specifies how the case conversion of physical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Force**: Indicates whether the physical name property of all the logical/physical models is overridden. If this option is enabled, the logical/physical link is broken between the logical and physical name. If this option is not enabled, all logical and physical names are set to the same value after the process completes.
Case Conversion of Logical Names	Specifies how the case conversion of logical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Mixed**: Indicates that the mixed-case logical names are preserved

Scheduler

Reverse Engineering Options for Amazon Keyspaces

Parameter	Description	Additional Information
Model	Specifies the location and name of the reverse engineered model	For example: C:\Scheduler\<Model Name>.erwin When you schedule a job on a remote server, ensure the model path is same for remote and local server.
Mart Folder	Specifies the location or library in your mart where the reverse engineered model is saved	To use this option, ensure that you are connected to a mart. For more information, refer to the Connecting to Mart topic.
Complete Compare	Specifies whether the Complete Compare (CC) process should run while reverse engineering	
Output File	Specifies the location of the CC output file generated	
File	Specifies that the target model location is on the local system	
Mart	Specifies that the target model location is in the mart	
Using Latest Version	Specifies whether the target model is the latest version of the model in the mart	This option is available only when Mart is selected.
Save To Mart	Specifies whether the reverse engineered model is saved to the mart	This option is available only when Using Latest Version is selected.
Target Model	Specifies the location of the target model for CC	
Option Set	Specifies the option set that is used for CC	**Advanced Default Option Set**: Indicates that all erwin DM metadata is included. CC works slowest with this option. **Speed Option Set**: Indicates that only the essen-

		tial metadata is included. CC works the fastest with this option set. **Standard Default Option Set**: Indicates that standard metadata is included. CC works fast with this option set compared to the Advanced option set.

Forward Engineering Models

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

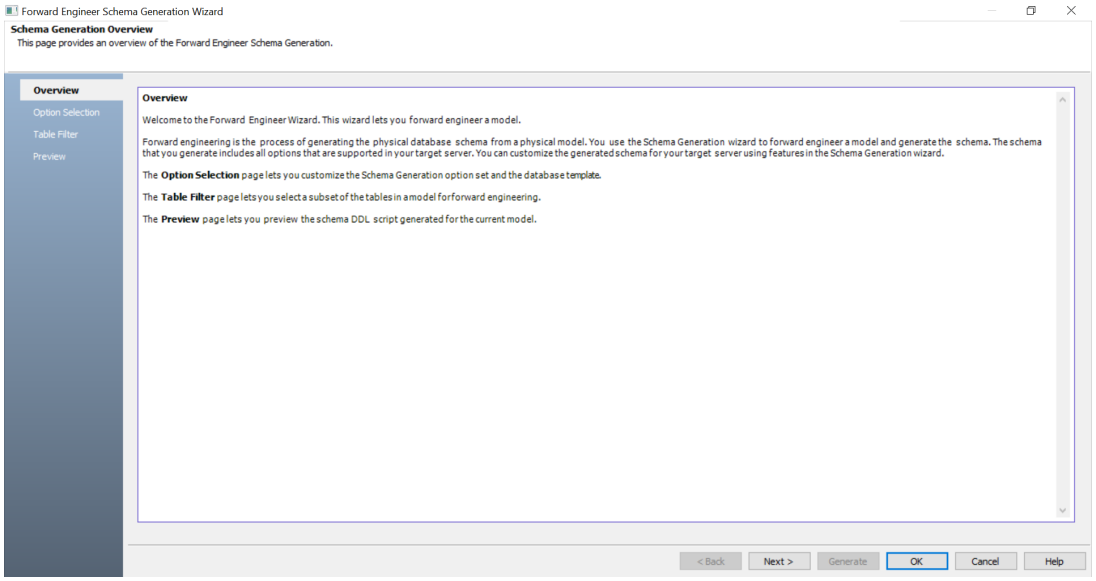
This topic walks you through the steps to forward engineer an Amazon Keyspaces model. For detailed description of forward engineering options, refer to the Forward Engineering Options topic.

To forward engineer an Amazon Keyspaces model:

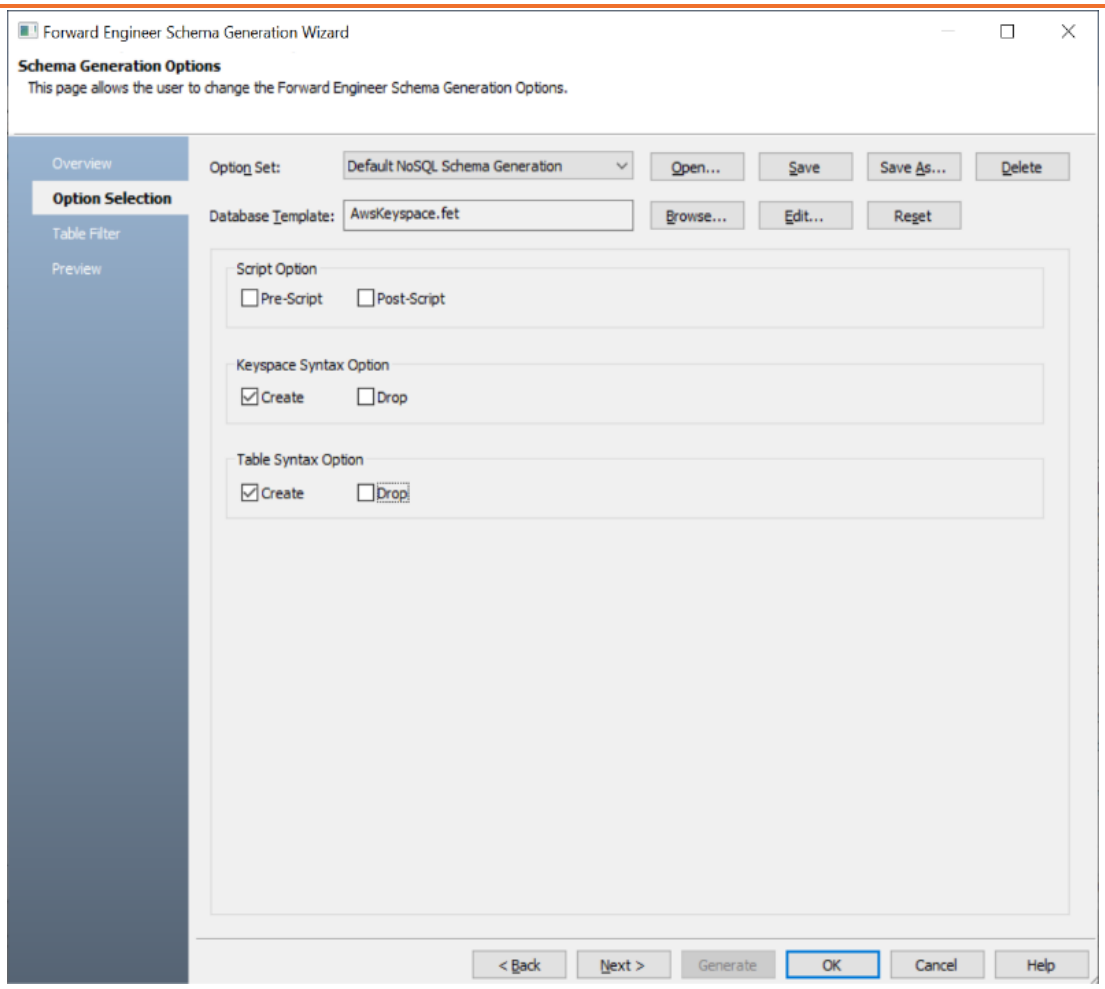
1. Open your Amazon Keyspaces model in erwin Data Modeler (DM).

   Ensure that you are in the Physical mode.

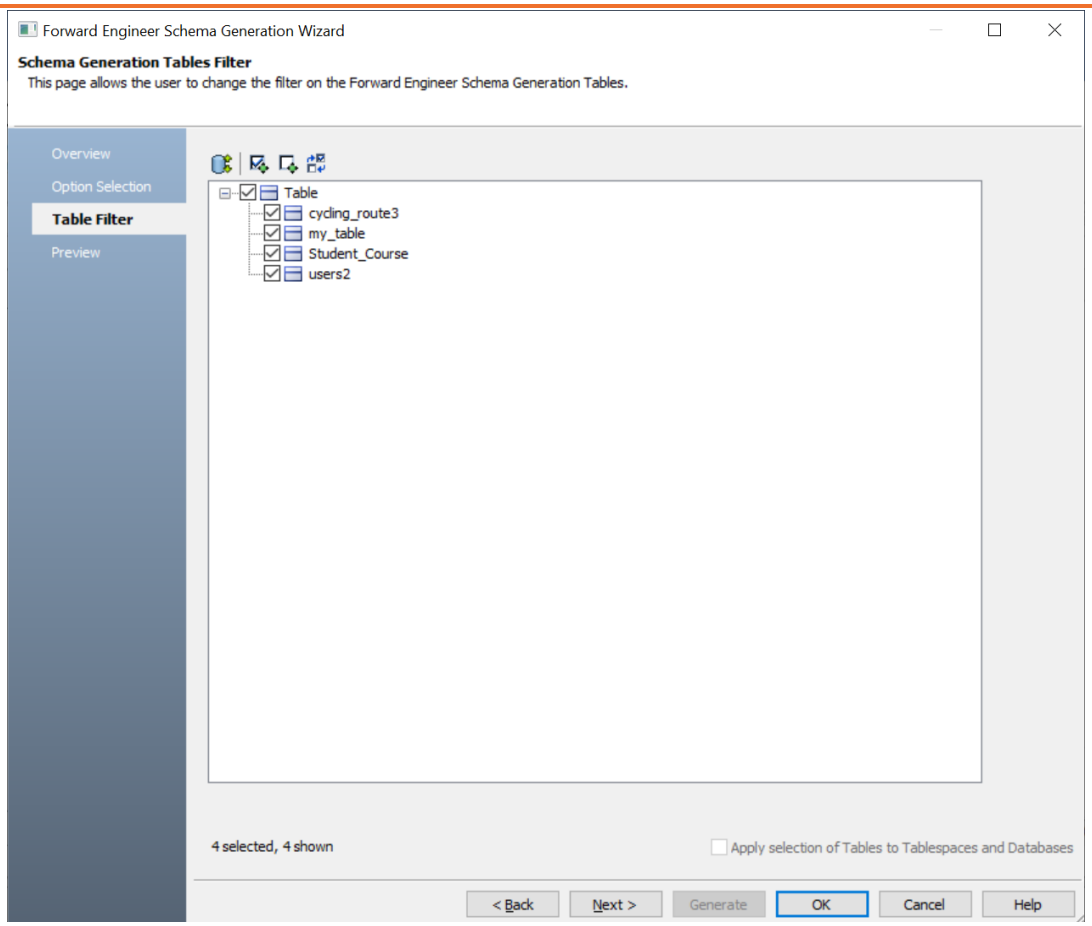
2. Click **Actions** > **Schema**.
   The Forward Engineer Schema Generation Wizard appears.



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

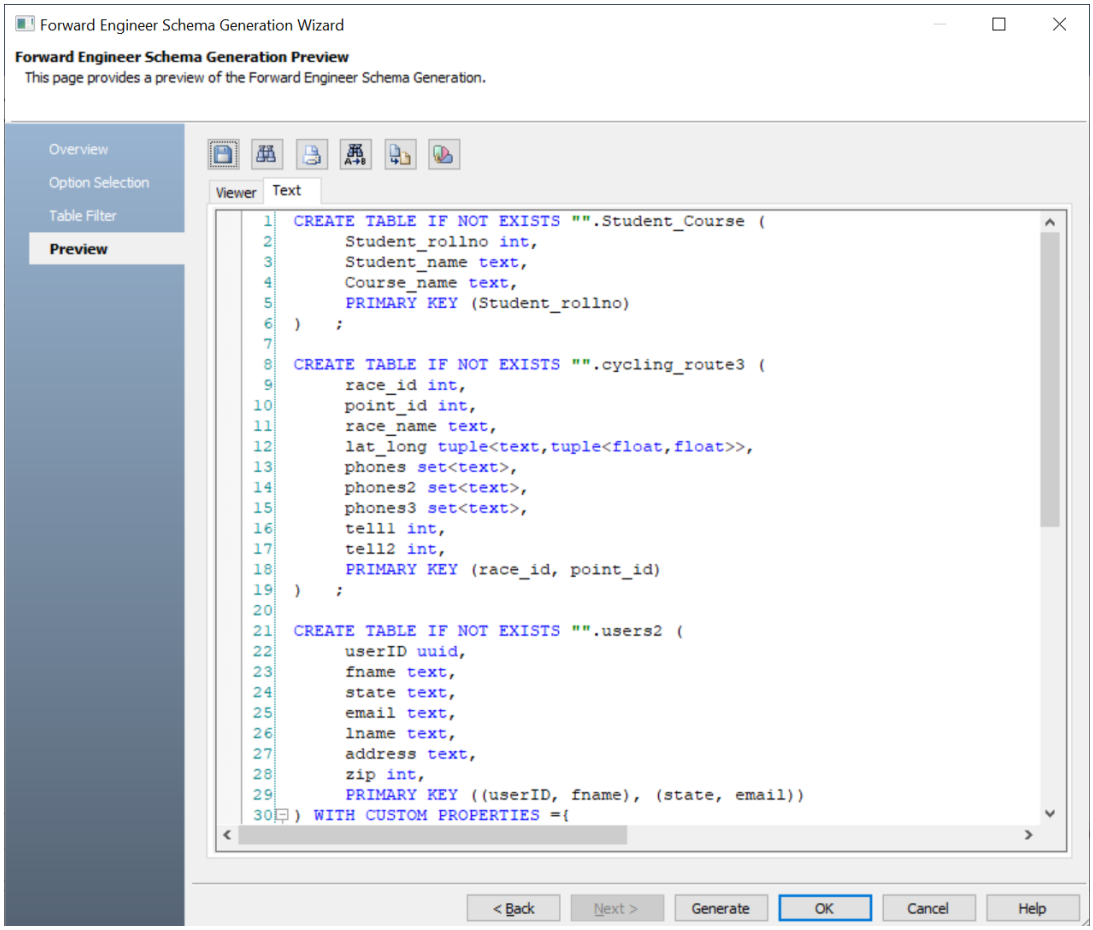


4. Click **Next**.
   The Table Filter tab appears. It displays a list of tables available in your model.



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

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



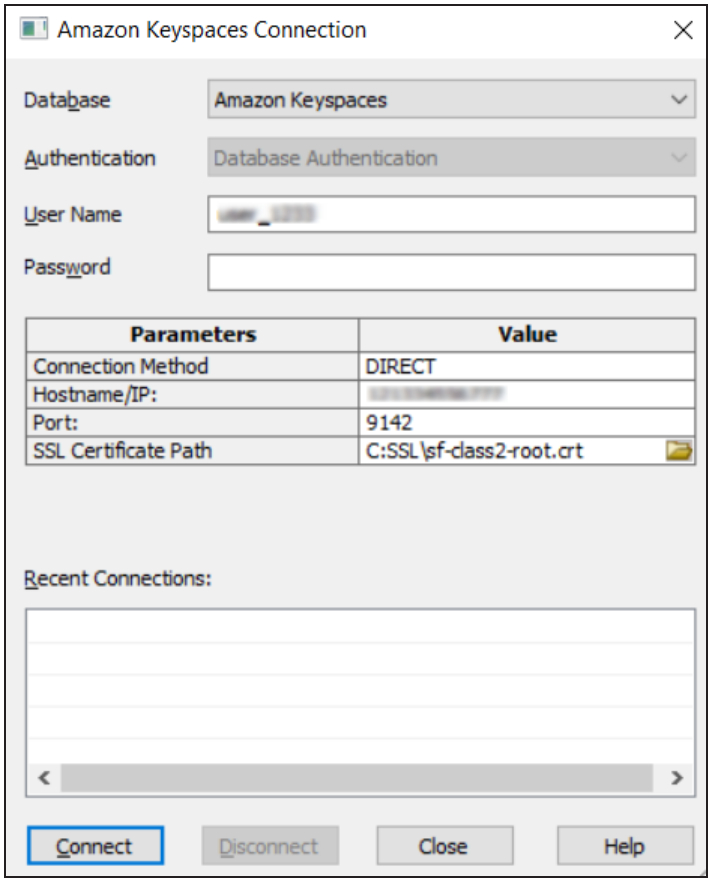
Use the following options:

- **Copy** ( ): Use this option to copy the script.
- **Save** ( ): Use this option to save the generated script in the CQL or SQL format.
- **Search** ( ): Use this option to search through the generated schema.
- **Print** ( ): Use this option to print the generated schema.
- **Replace** ( ): Use this option to find and replace in the generated schema.

- **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.
- **Error Check** (): Use this option to run an error check. Based on the results, you can correct the generated script.

7. Click **Generate**.

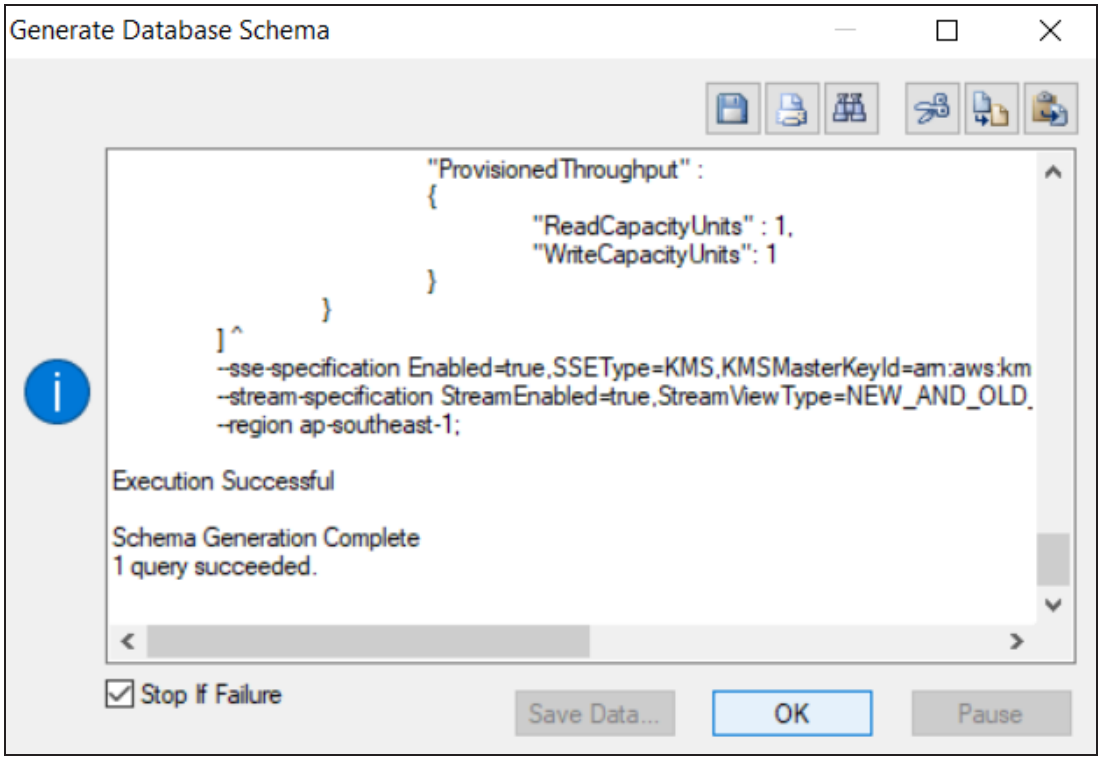
   The Amazon Keyspaces Connection screen appears.



8. Enter username, password, and appropriate connection parameters to connect the required database. Then, click **Connect**.

Objects in your model move to the database mentioned on the Amazon Keyspaces Connection screen irrespective of the databases defined on the object editor screens. If you want to retain objects in their respective databases as defined on the object editor screens, keep the database parameter blank.

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



Forward Engineering Options for Amazon Keyspaces

Following are the forward engineering options for Amazon Keyspaces.

Option Selection

Parameter	Description	Additional Information
Option Set	Specifies the option set template for forward engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save a configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at an external location. **Delete**: Use this option to delete an option set.
Database Template	Specifies the database template for controlling schema generation	**Browse**: Use this option to browse and select a database template. **Edit**: Use this option to edit a template in the Template Editor. **Reset**: Use this option to reset the Database Template option.
Script Option	Specifies the script option for schema generation	**Pre-Script**: Indicates whether pre-scripts attached to the schema are executed **Post-Script**: Indicates whether the post-scripts attached to the schema are executed
Keyspace Syntax Option	Specifies the keyspace syntax options for schema generation	**Create**: Indicates whether the Create syntax for keyspaces is executed **Drop**: Indicates whether the Drop syntax for keyspaces is executed
Table Syntax Option	Specifies the table syntax options for schema generation	**Create**: Indicates whether the Create syntax for tables is executed

		Drop: Indicates whether the Drop syntax for tables is executed

Table Filter

Parameter	Description	Additional Information
Tables	Specifies the selected tables for schema generation	
Display either Logical Names or Physical Names	Specifies the database template for controlling schema generation	**Logical Names**: Indicates that only logical names of the records are included in the generated schema **Physical Names**: Indicates that only physical names of the records are included in the generated schema **Physical Names, show owner**: Indicates that physical names and owners of the records are included in the generated schema **Physical Names, show owner using User**: Indicates that the physical names and owners of the records are included in the generated schema. Owners of the records are displayed using User.
Select all of the items in the list	Use this option to select all the records in the list.	
Unselect all of the items in the list	Use this option to clear all the records.	
Select all unselected items, and unselect all selected items	Use this option to select all the unselected records and clear all the previously selected records.	

Preview

Parameter	Description	Additional Information
Text	Displays the schema in the text editor	**Save**: Use this option to save the generated schema. **Search**: Use this option to search through the generated schema. **Print**: Use this option to print the generated schema. **Replace**: Use this option to find and replace text in the generated schema. **Copy**: Use this option to copy the selected text in the schema. **Text Options**: Use this option to edit window settings, fonts, syntax color.

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.

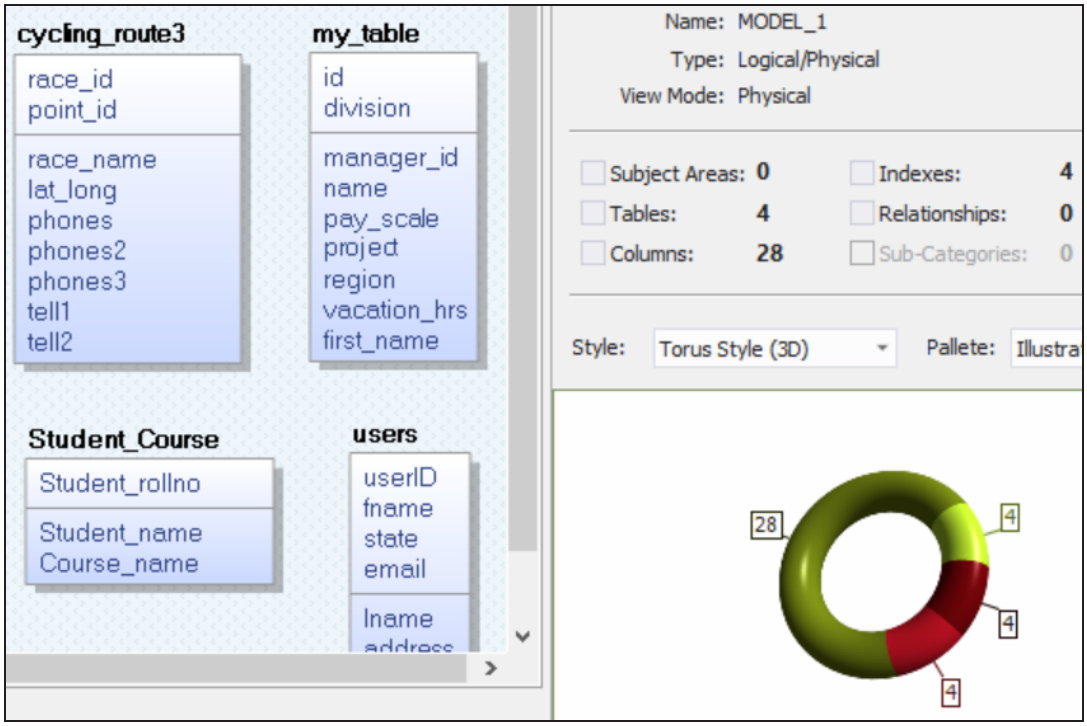
This topic walks you through the steps to compare an Amazon Keyspaces model with database.

To compare models with database:

1. Open your Amazon Keyspaces model.

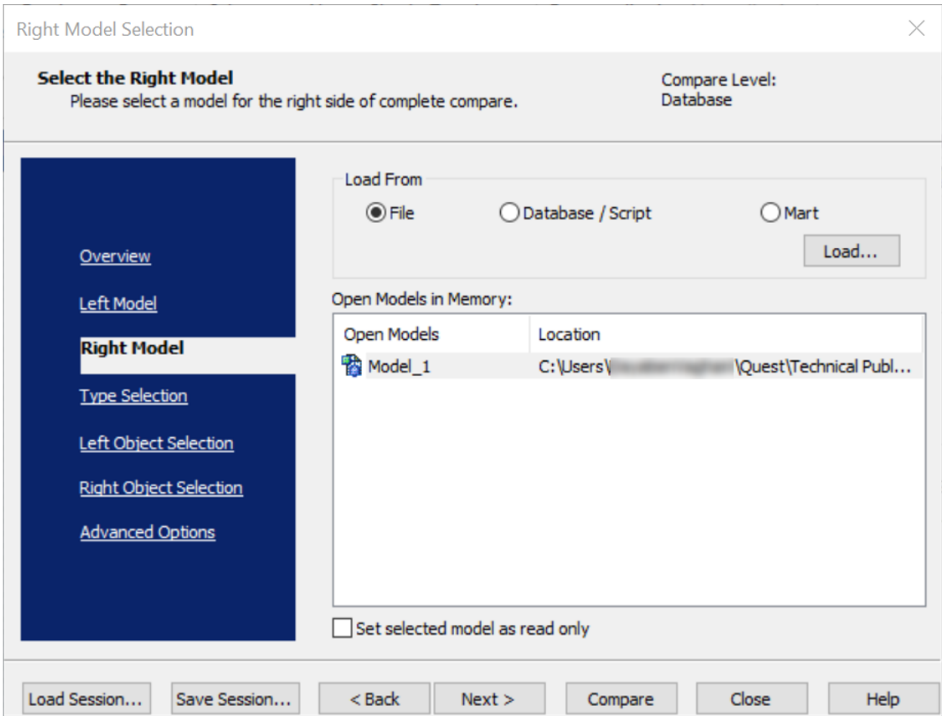
   Ensure that you are in the Physical mode.

   For example, the following image uses an Amazon Keyspaces model with 28 records.

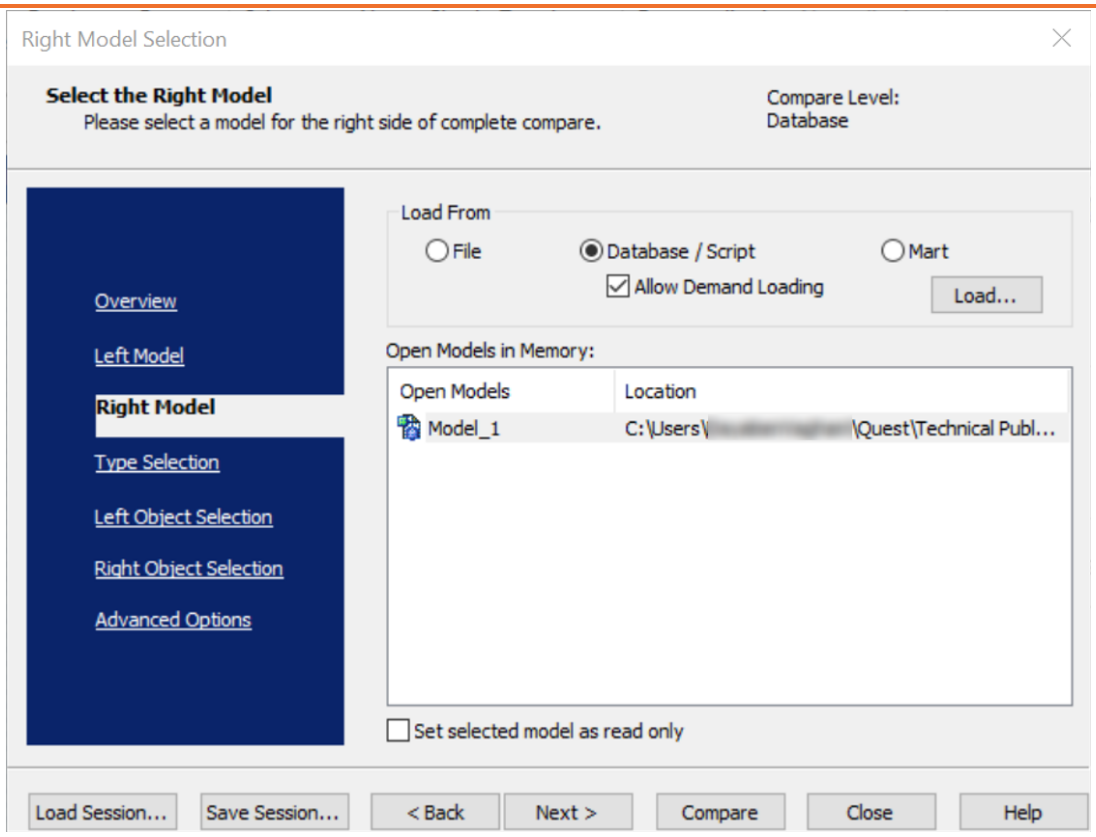


2. Click **Actions** > **Complete Compare**.
   By default, the Complete Compare wizard assigns the open model as the Left Model.

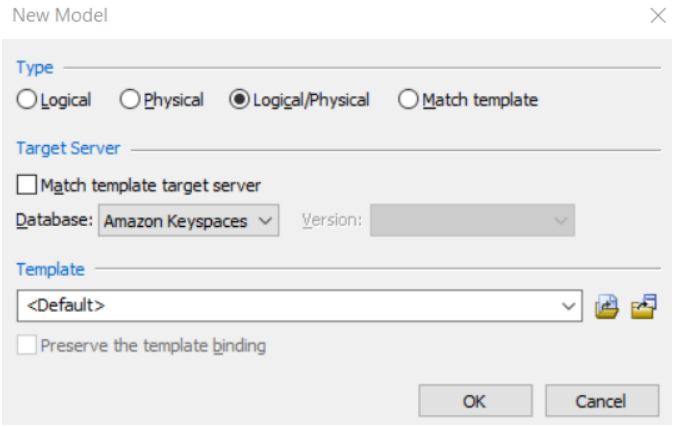
Hence, the Right Model tab appears.



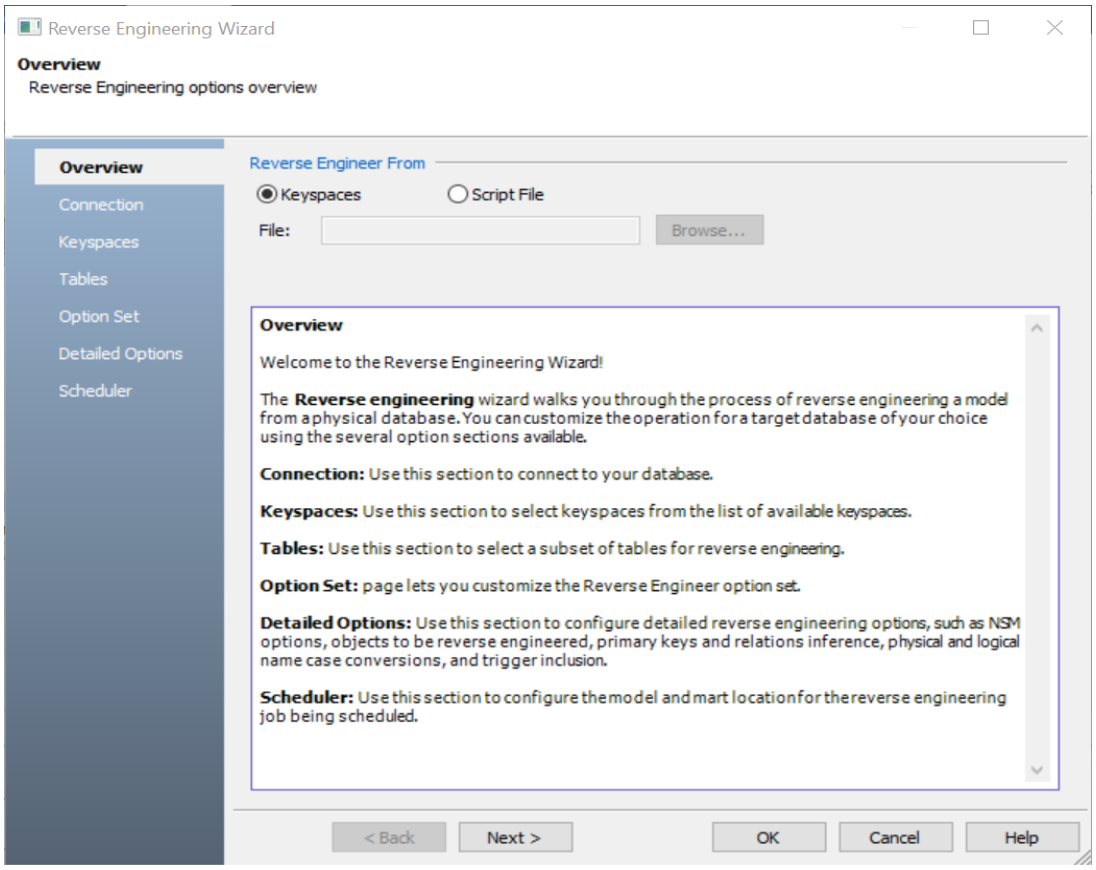
3. Click **Database/Script**.
   By default, the Allow Demand Loading option is selected.



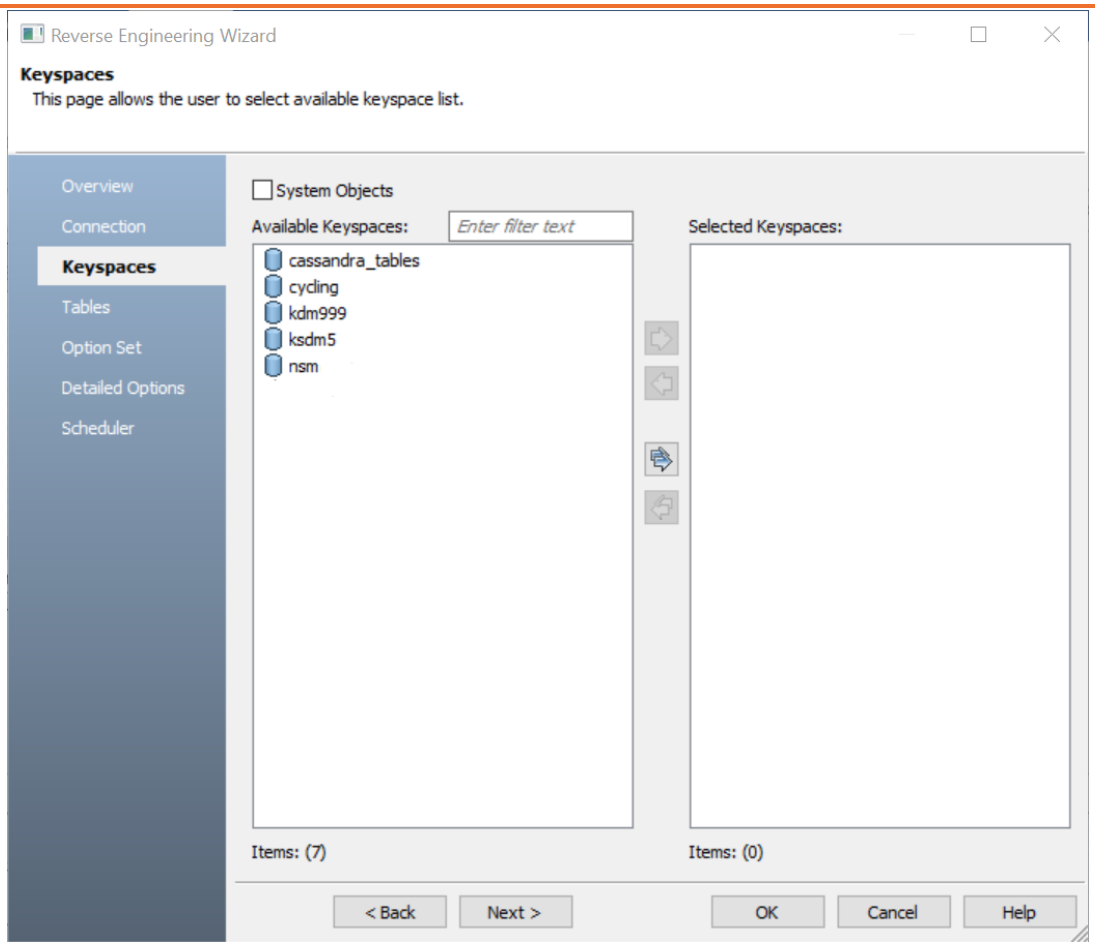
4. Click **Load**.
   The New Model dialog box appears. This starts the reverse engineering process to pull a model from the database to compare.



5. Ensure that the Database is set to Amazon Keyspaces. Then, click **Next**.
   The Reverse Engineering Wizard appears.

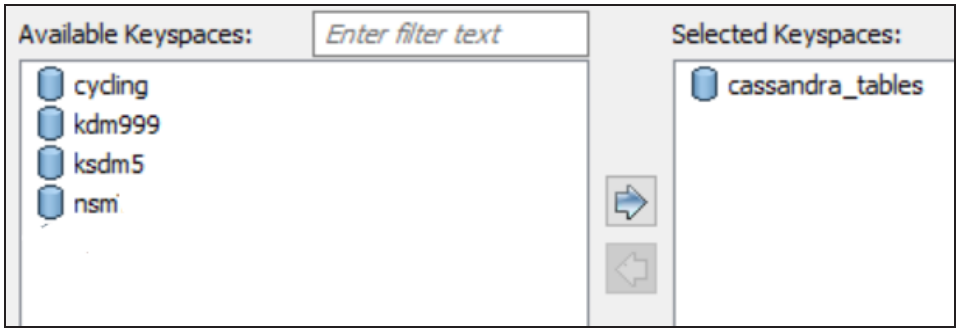


6. Click **Keyspaces**. Then, click **Next**.
   The Connection tab appears. Use this tab to connect to the database from which you want to reverse engineer the model.
7. After connection is established, click **Next**.
   The Keyspaces tab appears. It displays a list of available keyspaces.

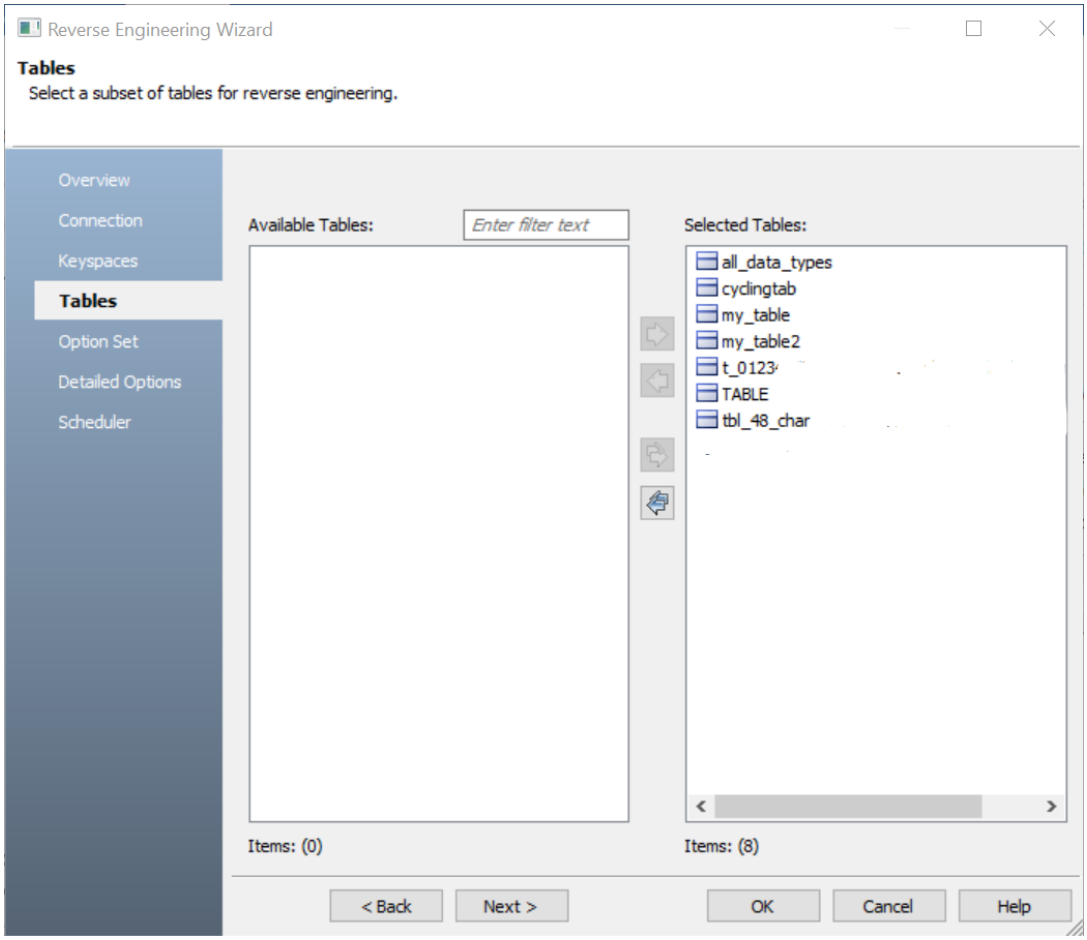


8. Under **Available Keyspaces**, select the keyspaces that you want to reverse engineer. Then, click  .

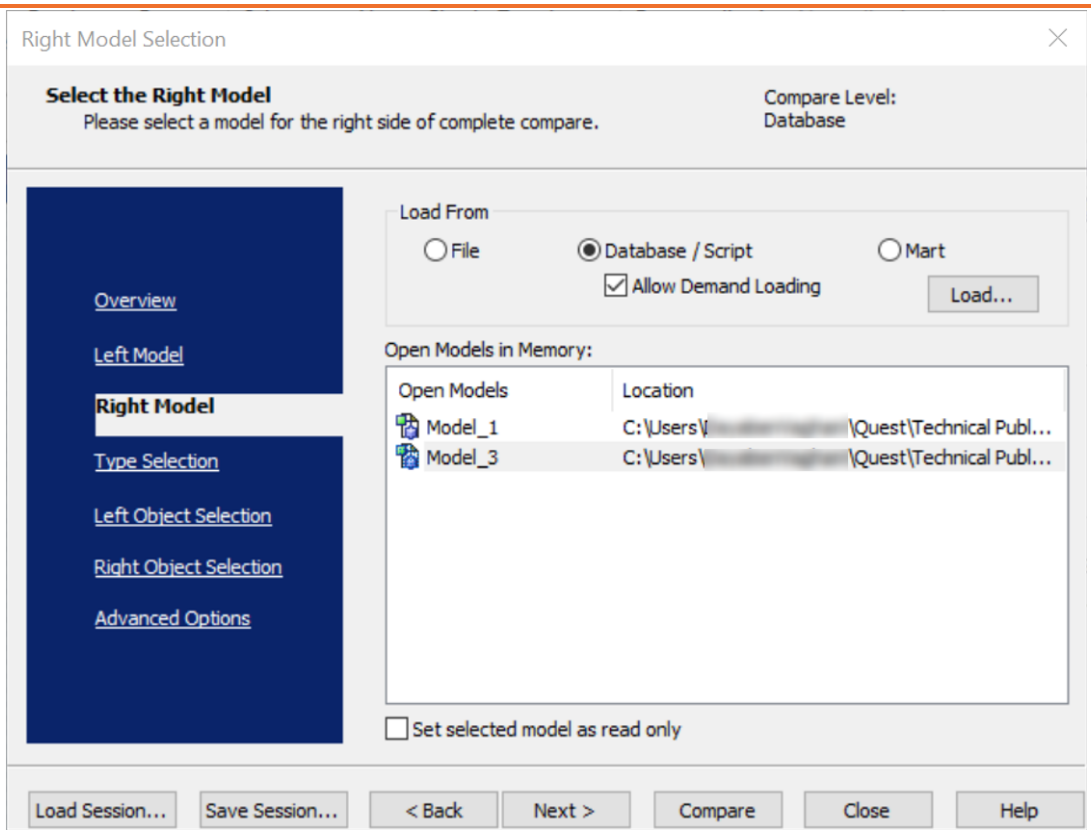
   This moves the selected keyspaces under Selected Keyspaces.



9. Click **Next** and on the Tables tab, click  .
   This selects all the available tables.

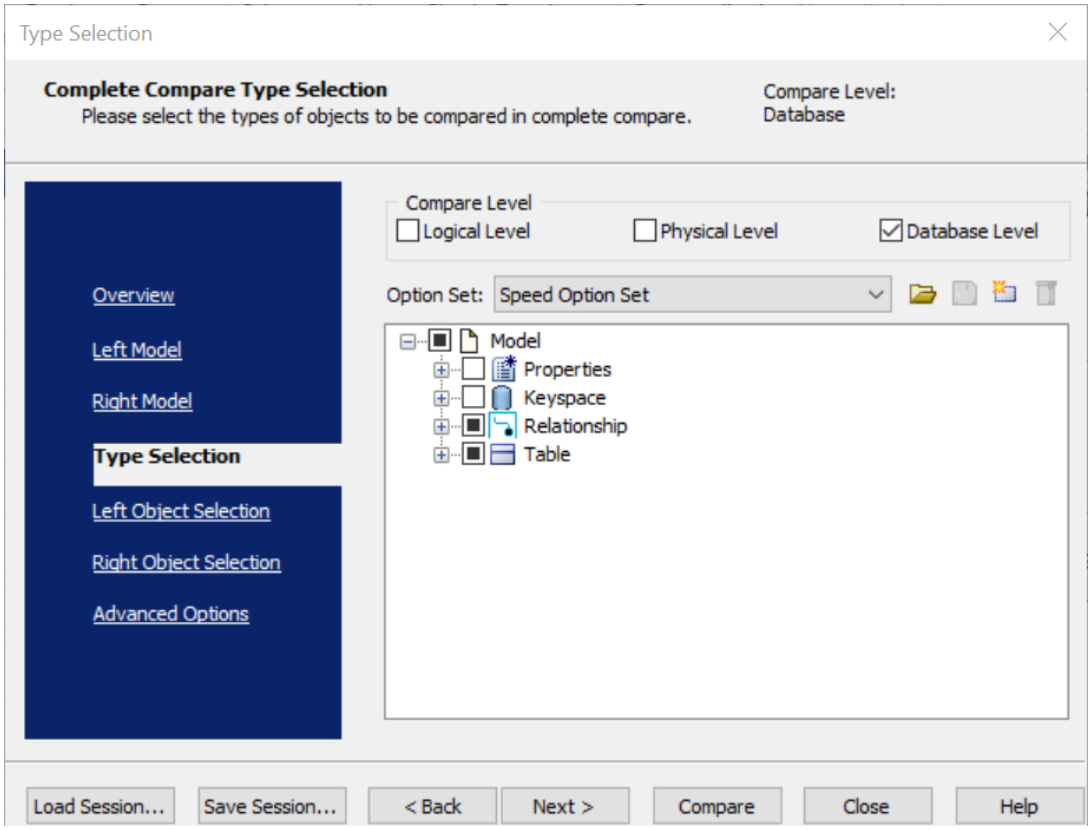


10. Click **Next** and on the Option Set tab, keep the default configuration.
11. Click **Next** and on the Detail Options tab, keep the default configuration.
12. Click **OK**.
    The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse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



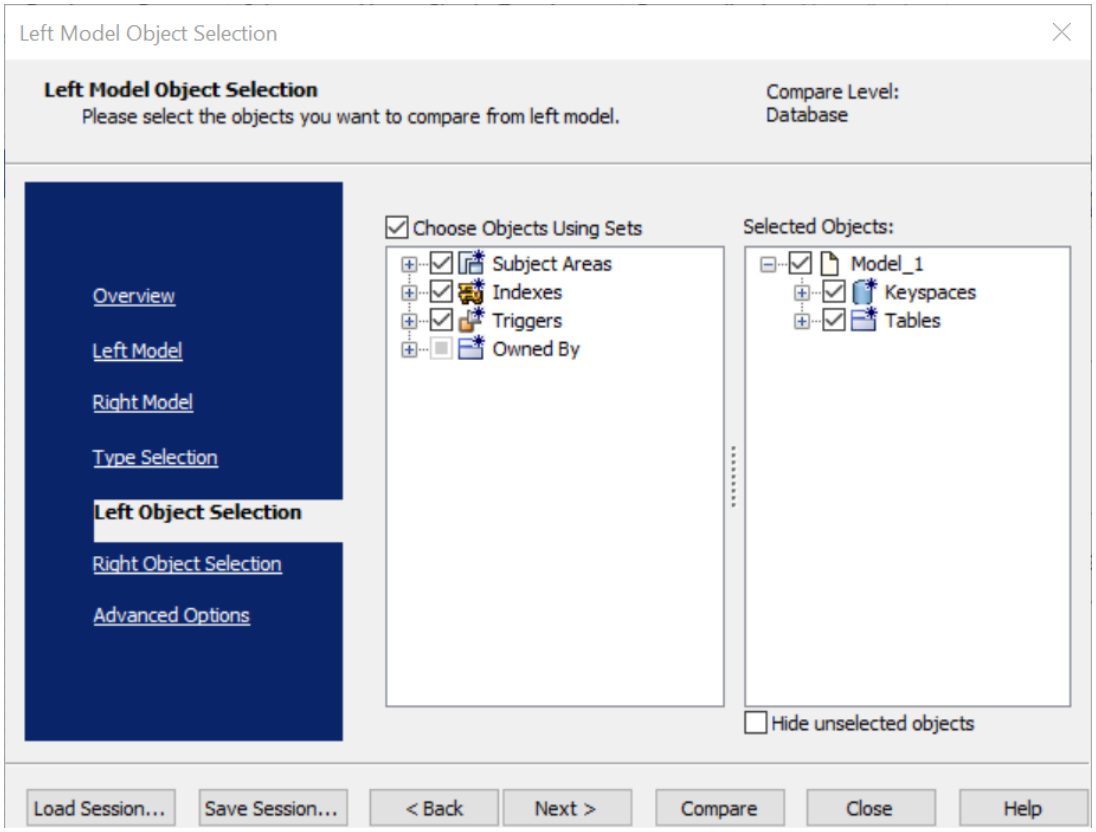
13. Click **Next** and on the Type Selection tab, select the appropriate options.

For example, the following image shows the default options.



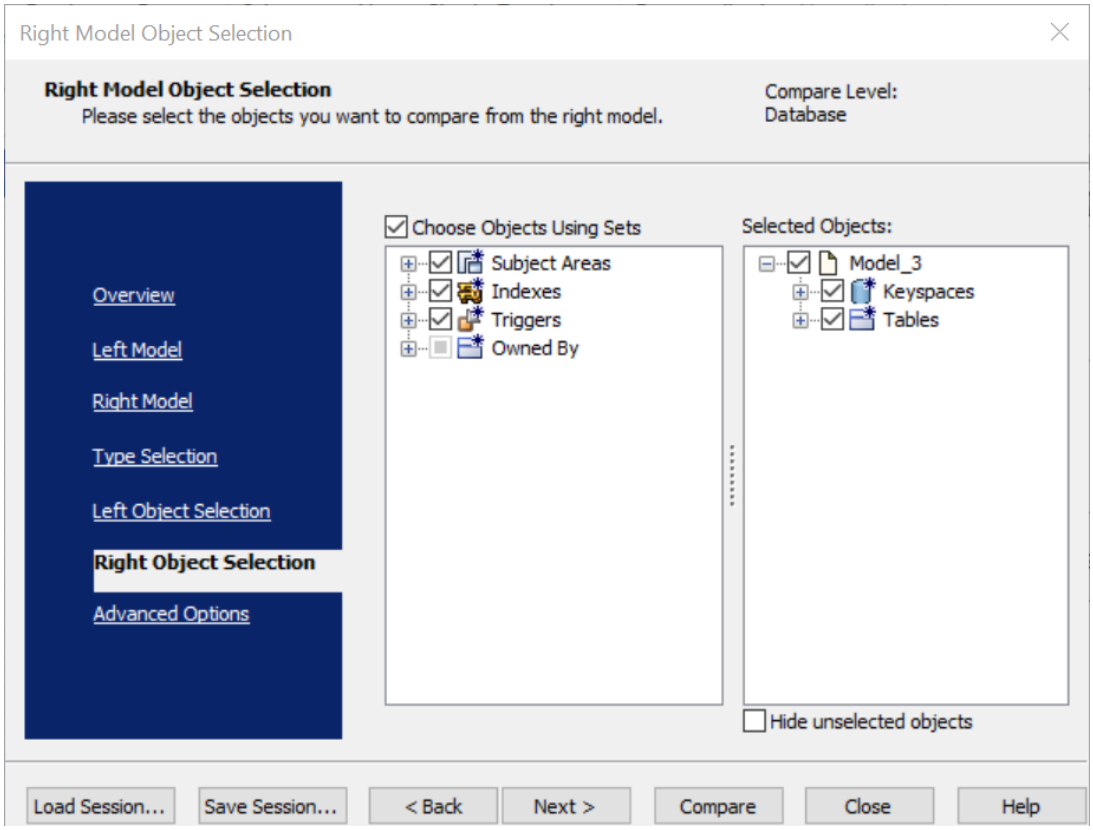
14. Click **Next** and on the Left Object Selection tab, select the appropriate options.

For example, the following image shows the default options.

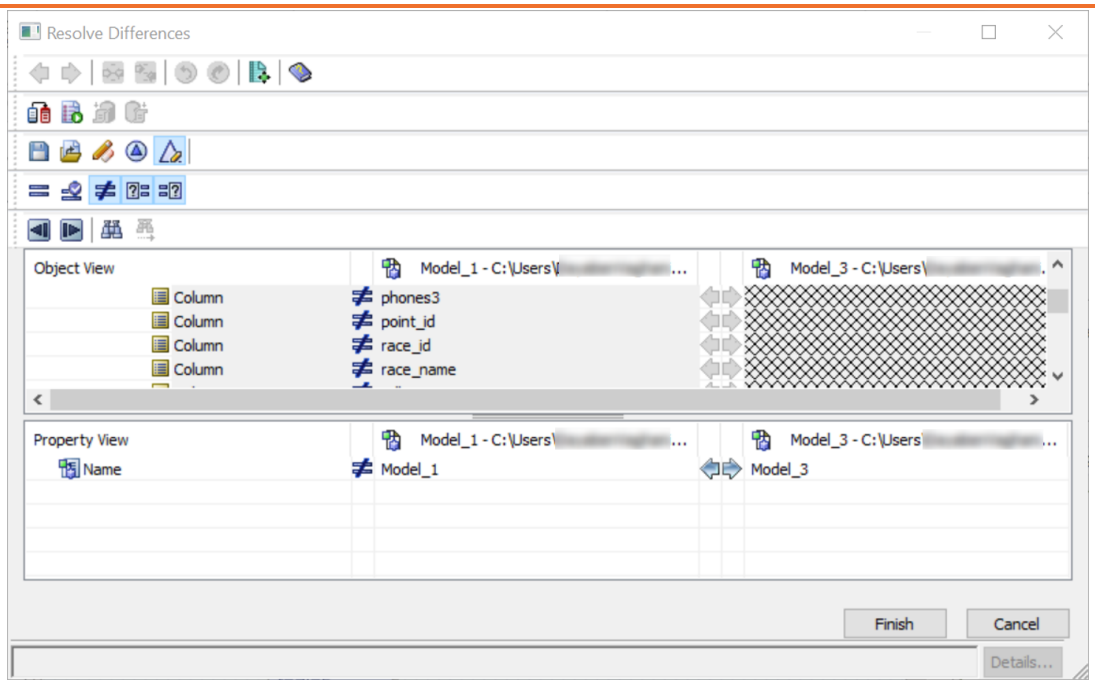


15. Click **Next** and on the Right Object Selection tab, select the appropriate options.

For example, the following image shows the default options.



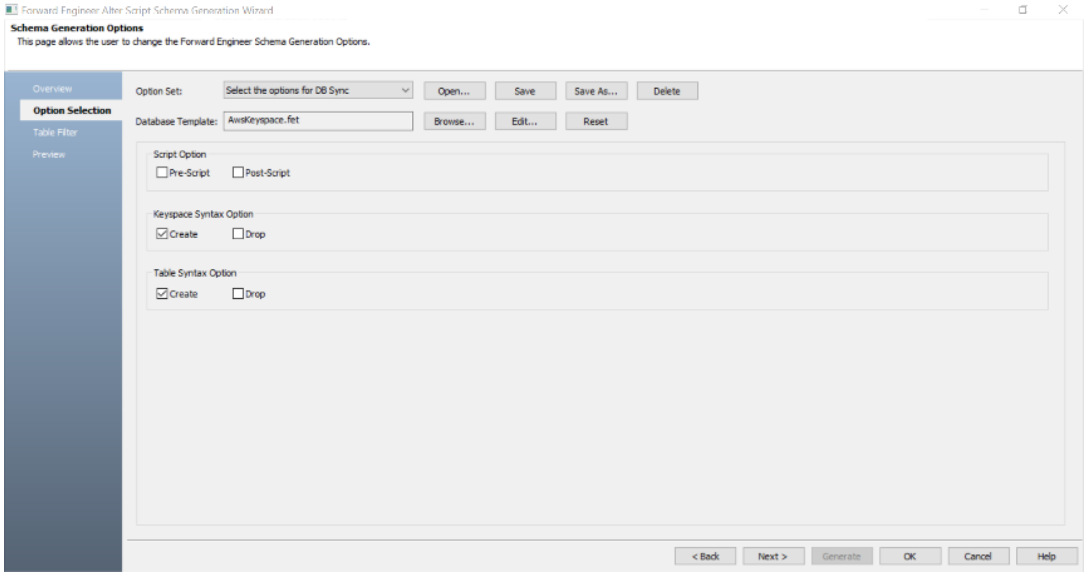
16. Click **Compare**.
    The comparison process runs, and the Resolve Differences dialog box appears. It displays the differences between your model and database.
    For example, the following image shows that the point_id table is available in your model but not in the database.



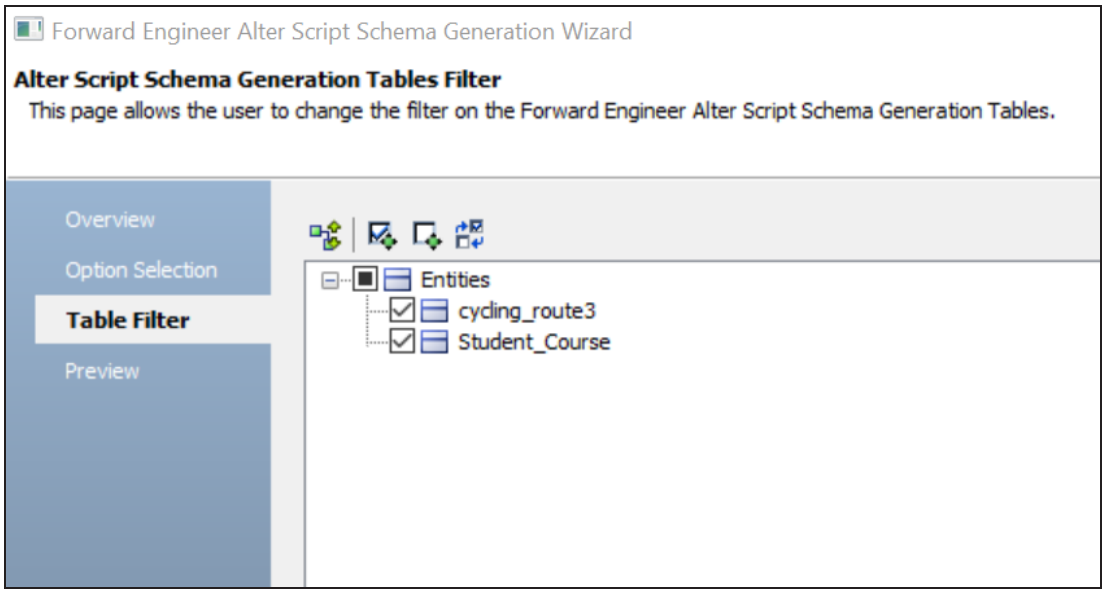
Select the Rating collection and click . This will move the point_id table to the right model (from the database). Similarly, resolve other differences.

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

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



19. Click **Table Filter** and select or verify the tables to be included in the forward engineering script.



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

21. Click **Generate** and connect to your Amazon Keyspaces 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**.

Migrating Relational Models to Amazon Keyspaces Models

You can migrate your relational models to Amazon Keyspaces 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 an Amazon Keyspaces model.

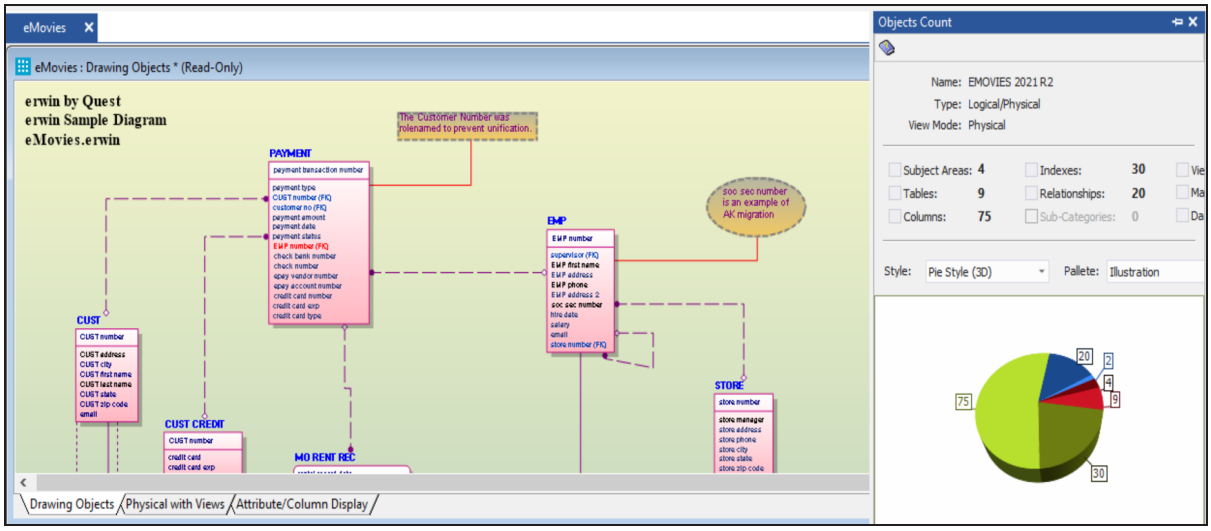
Migration by Changing the Target Database

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

1. Open your relational model in erwin Data Modeler (DM).

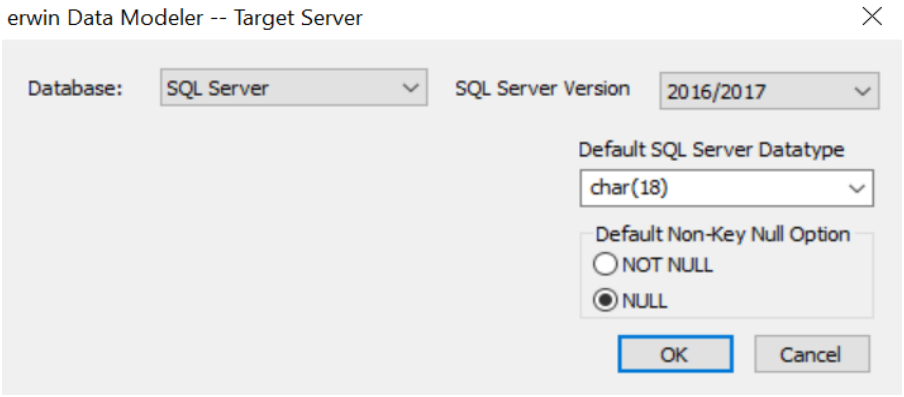
   Ensure that you are in the Physical mode.

   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.

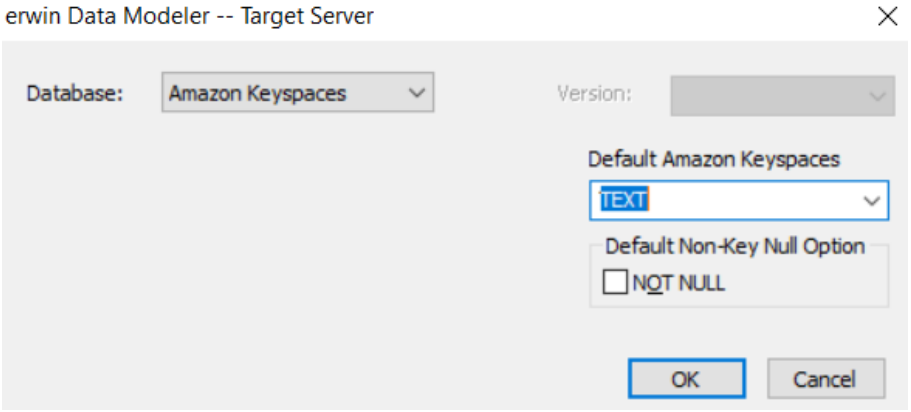


2. On the ribbon, click **Actions** > **Target Database** or on the status bar, click the database name.

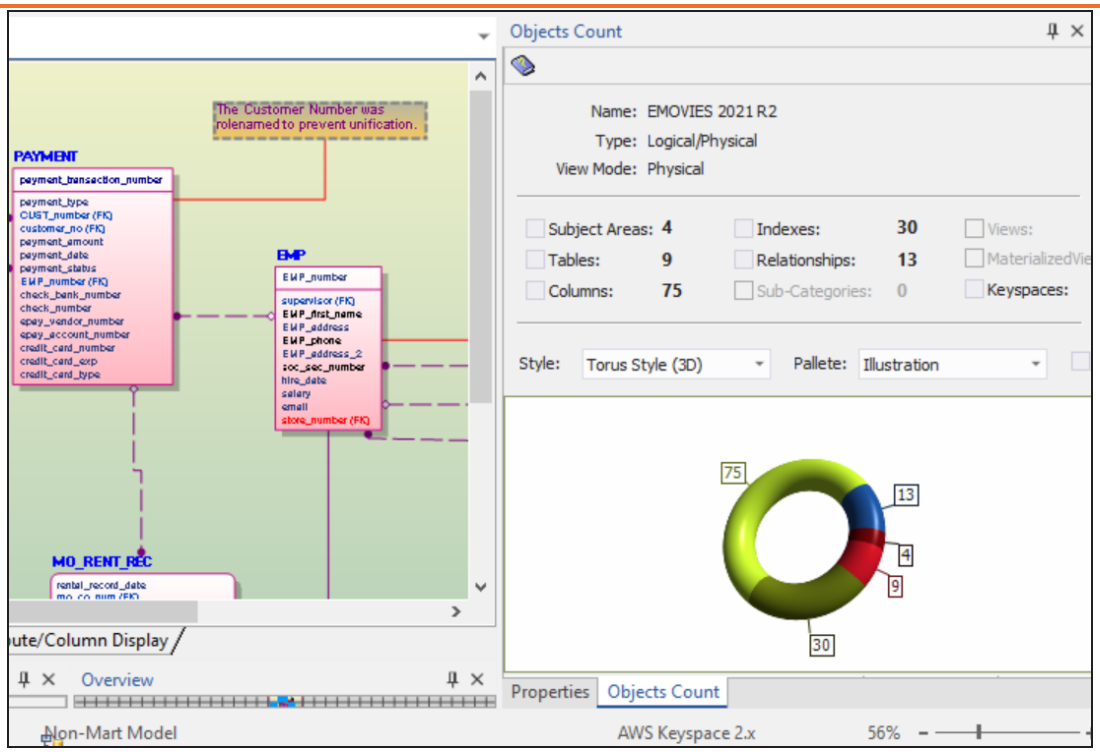
   The erwin Data Modeler -- Target Server screen appears.



3. In the **Database** drop-down list, select Amazon Keyspaces.



4. Click **OK**.
   The conversion process starts.
   Once the conversion is complete, the existing model is migrated to an Amazon Keyspaces database.



In the **Objects Count** pane, note that instead of databases, we now have keyspaces. The migration process converts and merges multiple tables, columns, and relationships to the Amazon Keyspaces format.

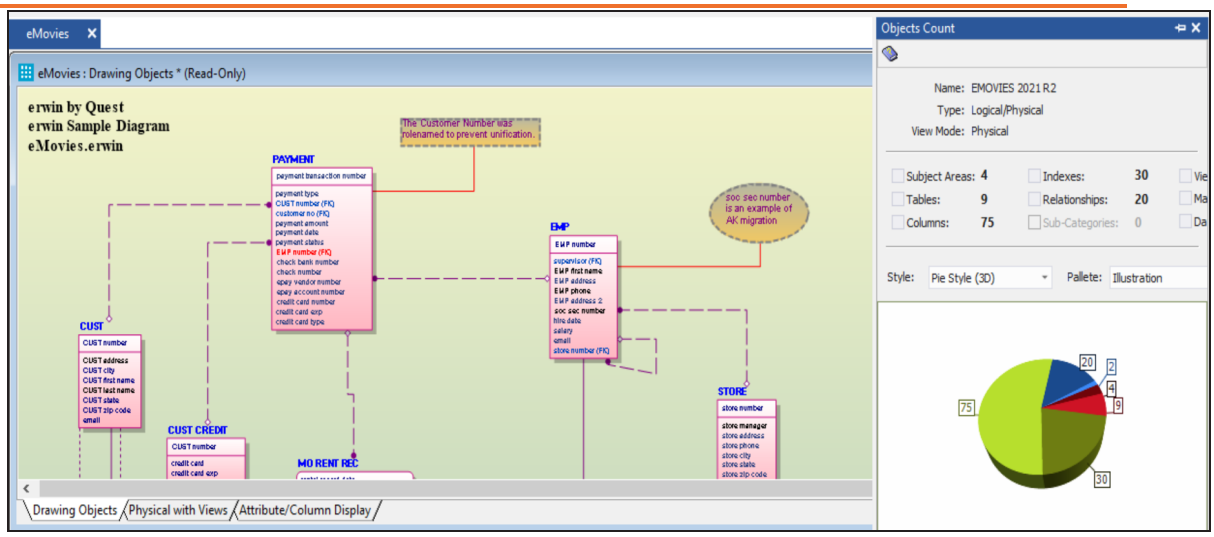
Migration by Deriving a Model

To migrate by deriving a model, follow these steps:

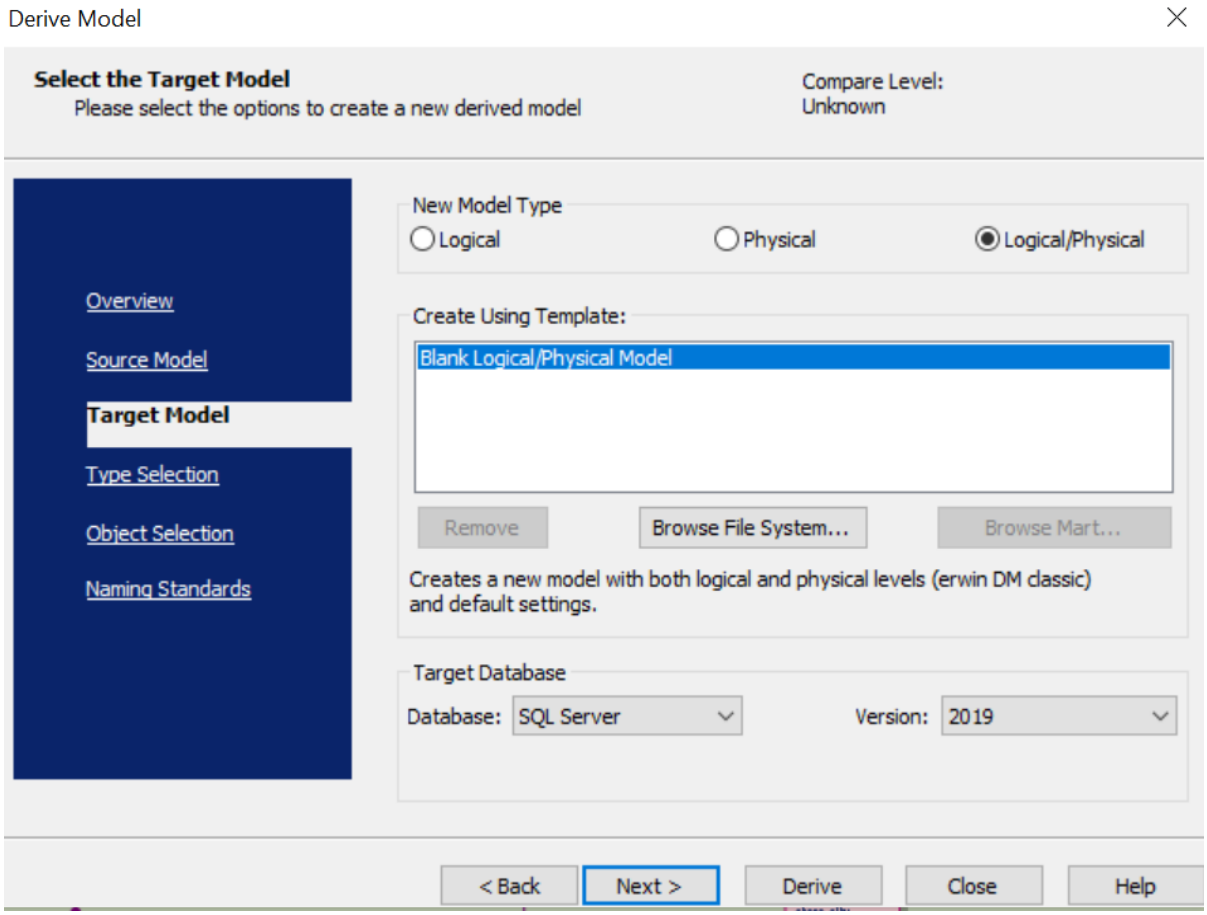
1. Open your relational model in erwin Data Modeler (DM).

   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.



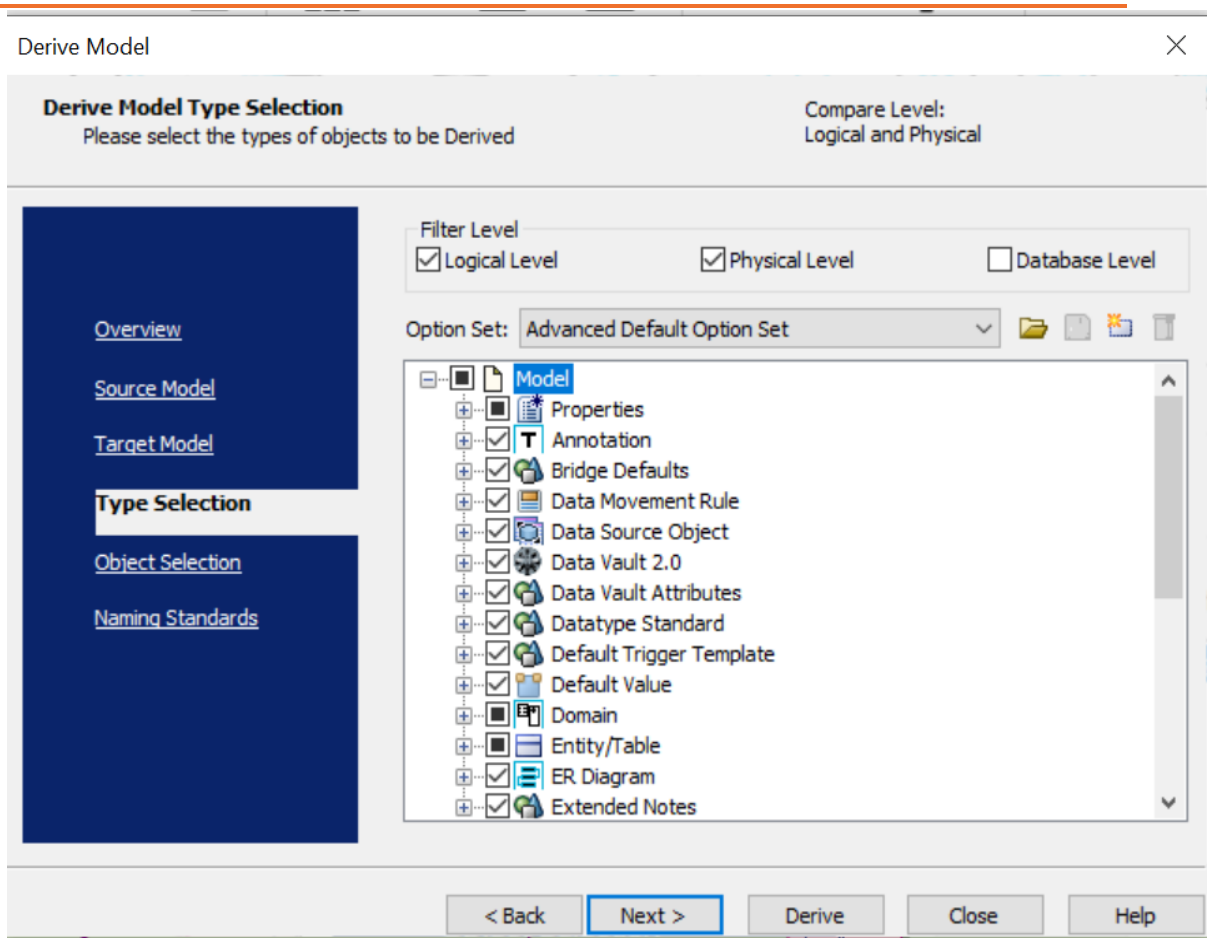
2. On the ribbon, click **Actions** > **Design Layers** > **Derive New Model**.
   The Derive Model screen appears. By default, the Source Model is set to your current model.



3. In the **Database** drop-down list, select **Amazon Keyspaces**.
4. Click **Next**.

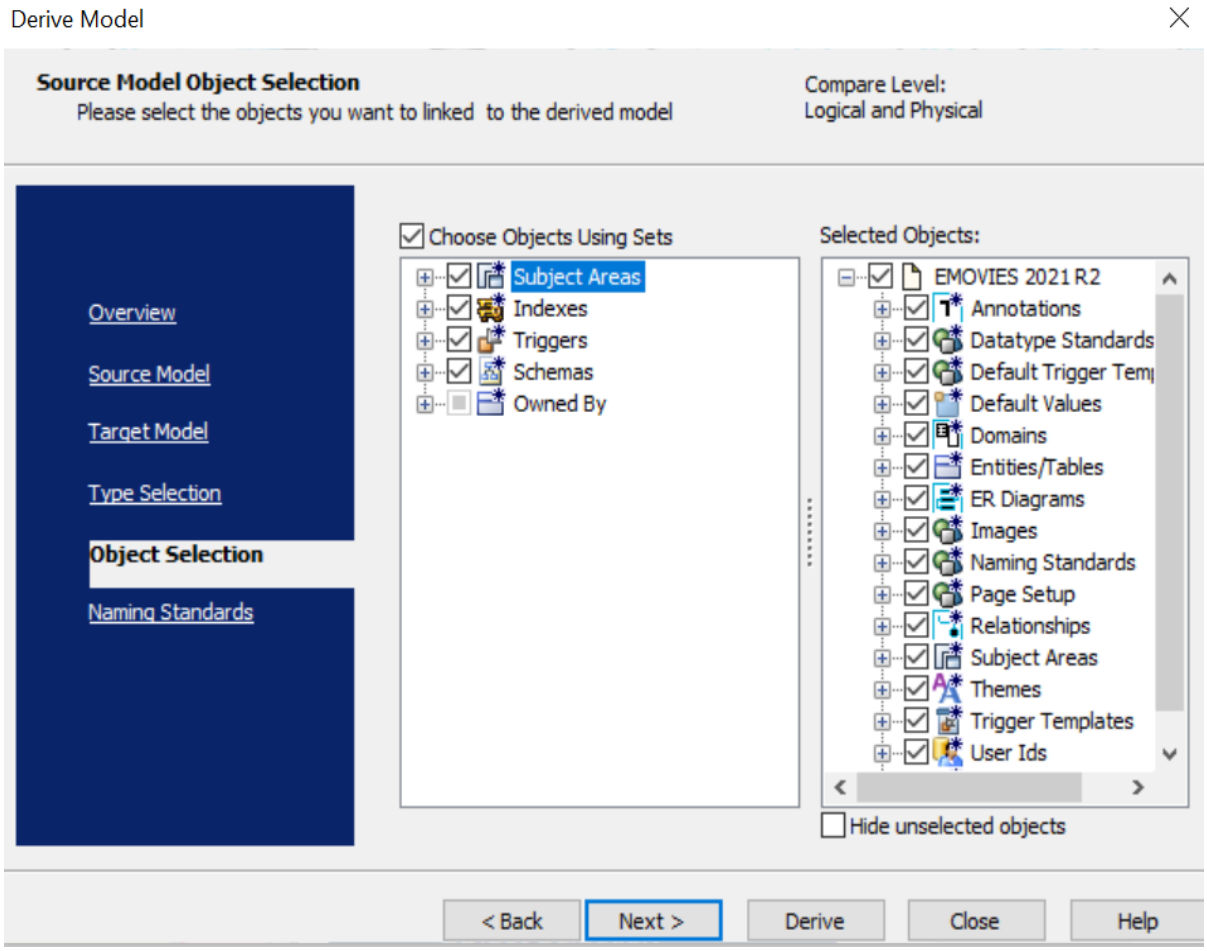
> If the Type Resolution screen appears, click **Finish**.

The Type Selection tab appears.

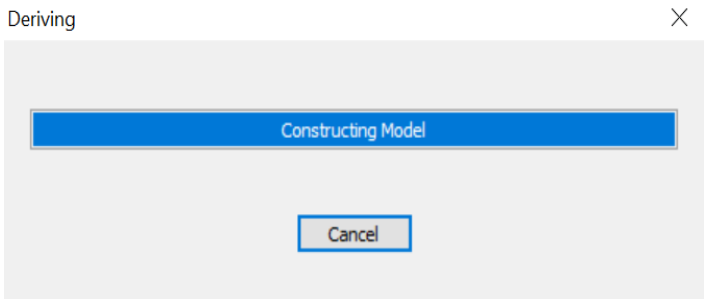


5. Select the types of objects that you want to derive into the target Amazon Keyspaces model.

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

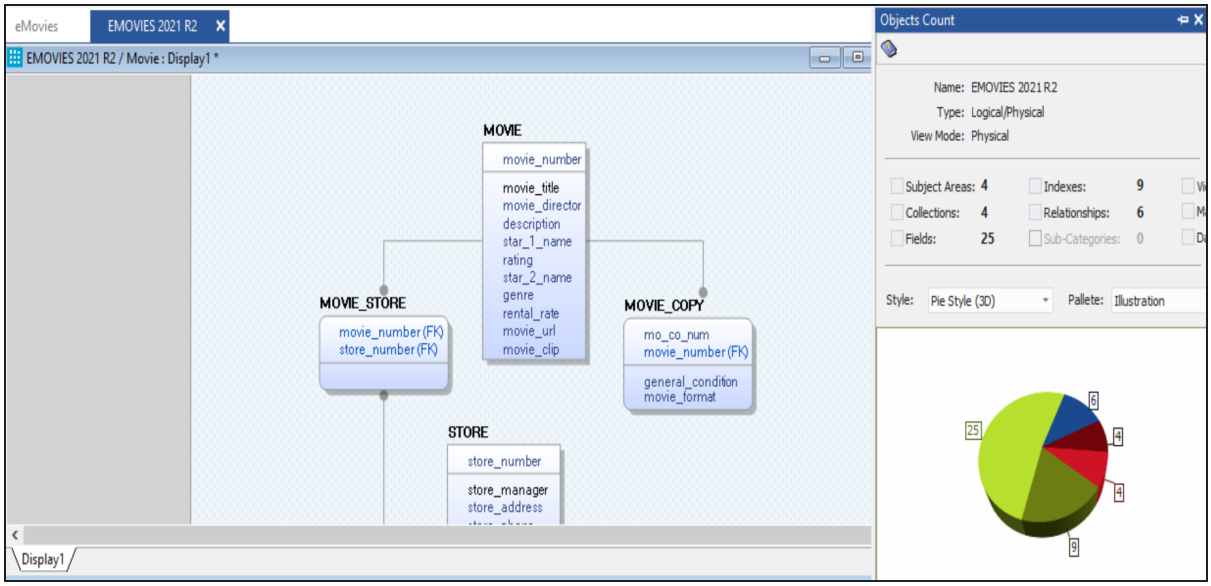


7. Select the objects that you want to derive into the target Amazon Keyspaces model.
8. Click **Derive**.
   The model derivation process starts.



Once the conversion is complete, the existing model is migrated to an

Amazon Keyspaces database.



In the **Objects Count** pane, note that instead of tables and columns, we now have entities and attributes. The migration process converts and merges multiple tables, columns, and relationships to the Amazon Keyspaces format.

Google BigQuery Support

erwin Data Modeler (DM) now supports Google BigQuery as a target database. This implementation supports the following objects:

- Dataset
- Function
- Materialized View
- Stored Procedure
- Table
  - Columns
  - Row Access Policy
- View

Following are the supported data types:

- BIGNUMERIC
- BOOLEAN
- BYTES
- DATE
- DATETIME
- FLOAT
- GEOGRAPHY
- INTEGER
- INTERVAL
- NUMERIC
- RECORD/STRUCT
- STRING

- TIME
- TIMESTAMP

Google BigQuery implementation supports all erwin DM features and functions. The following sections walk you through these features:

- Reverse engineering models from database and script
- Forward engineering models to database
- Comparing changes using Complete Compare

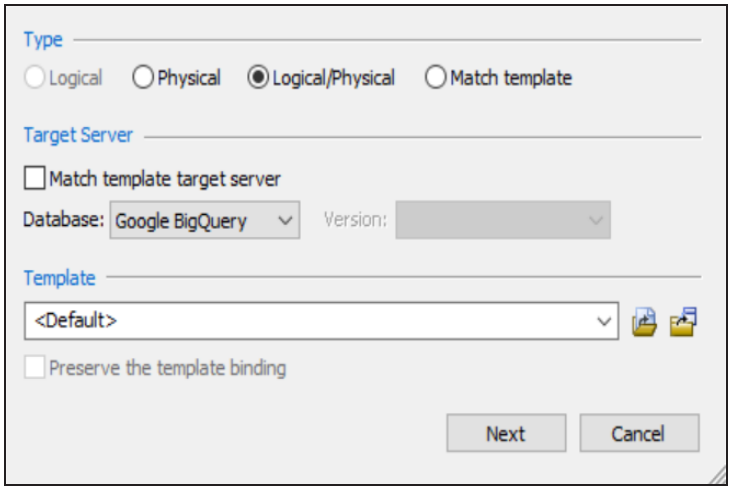
Reverse Engineering Models

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

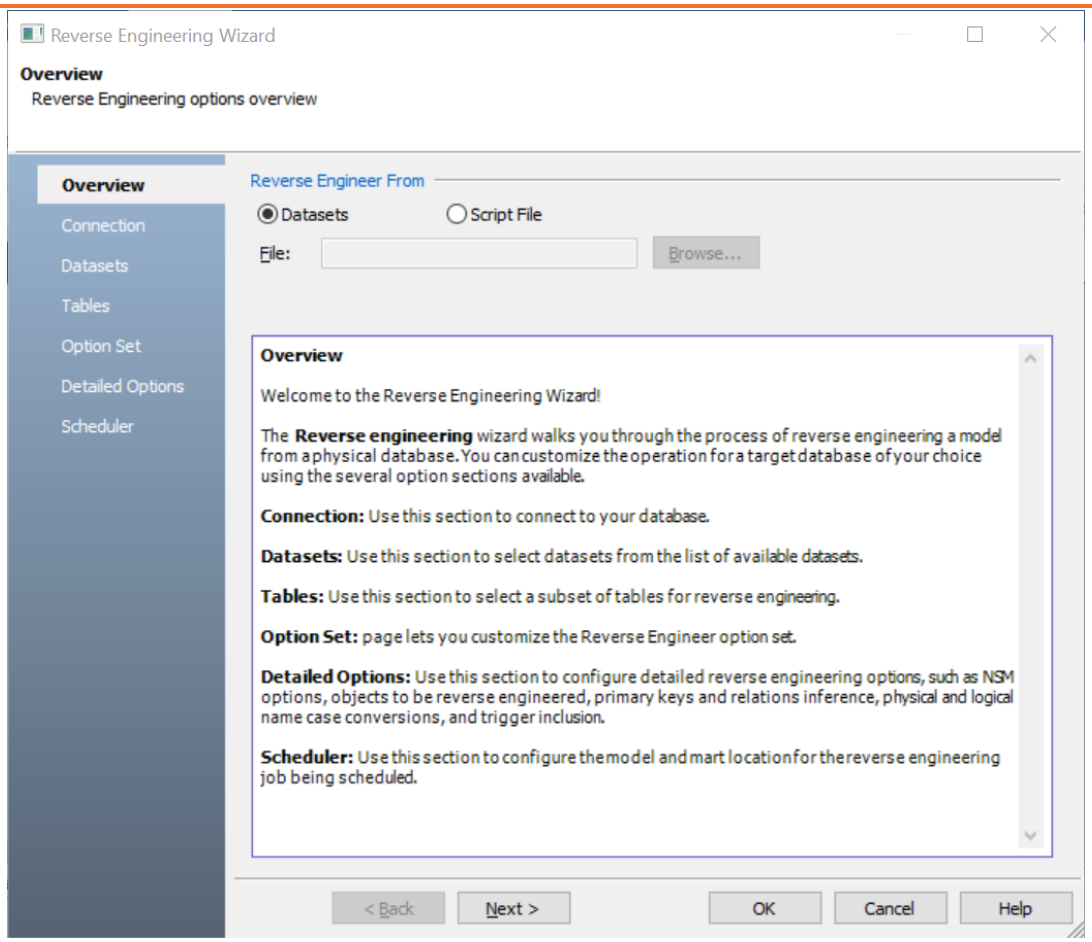
This topic walks you through the steps to reverse engineer a Google BigQuery model. For detailed description of reverse engineering options, refer to the Reverse Engineering Options topic.

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 Google BigQuery.



3. Click **Next**.
   The Reverse Engineering Wizard appears.



4. Click one of the following options:
    - **Datasets**: Use this option to reverse engineer a model from your dataset.

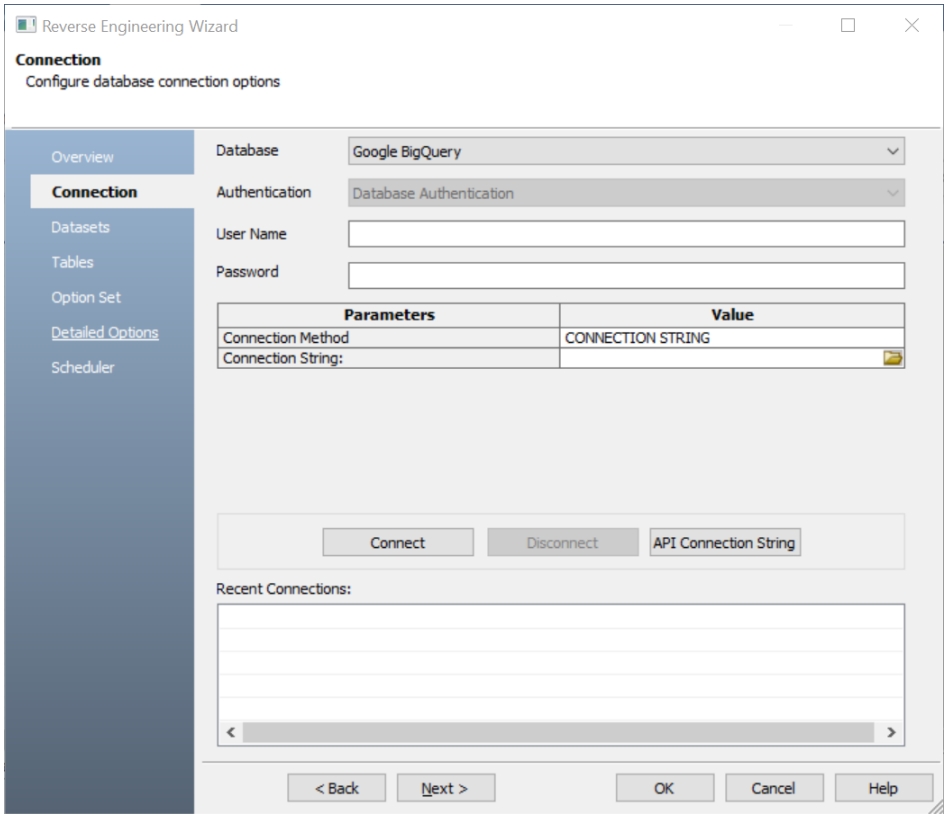
      If you click **Datasets**, continue to step 5.

    - **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.

      If you click **Script File**, see step 13 below.

5. Click **Next**.

The Connection tab appears.



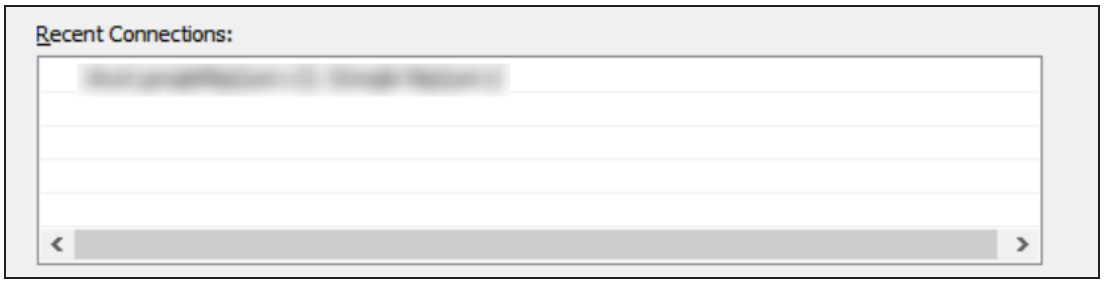
6. Enter your **User Name** and **Password**.

   The following table explains the connection parameters.

Parameter	Description	Additional Information
Connection Method	Specifies the type of connection you want to use. **Connection String** connects to your cluster using a connection string.	
Connection String	Specifies the path to the secure connect JSON file in the following format: *C:\<file name>.json*	This option is available when Connection Method is set to Connection String

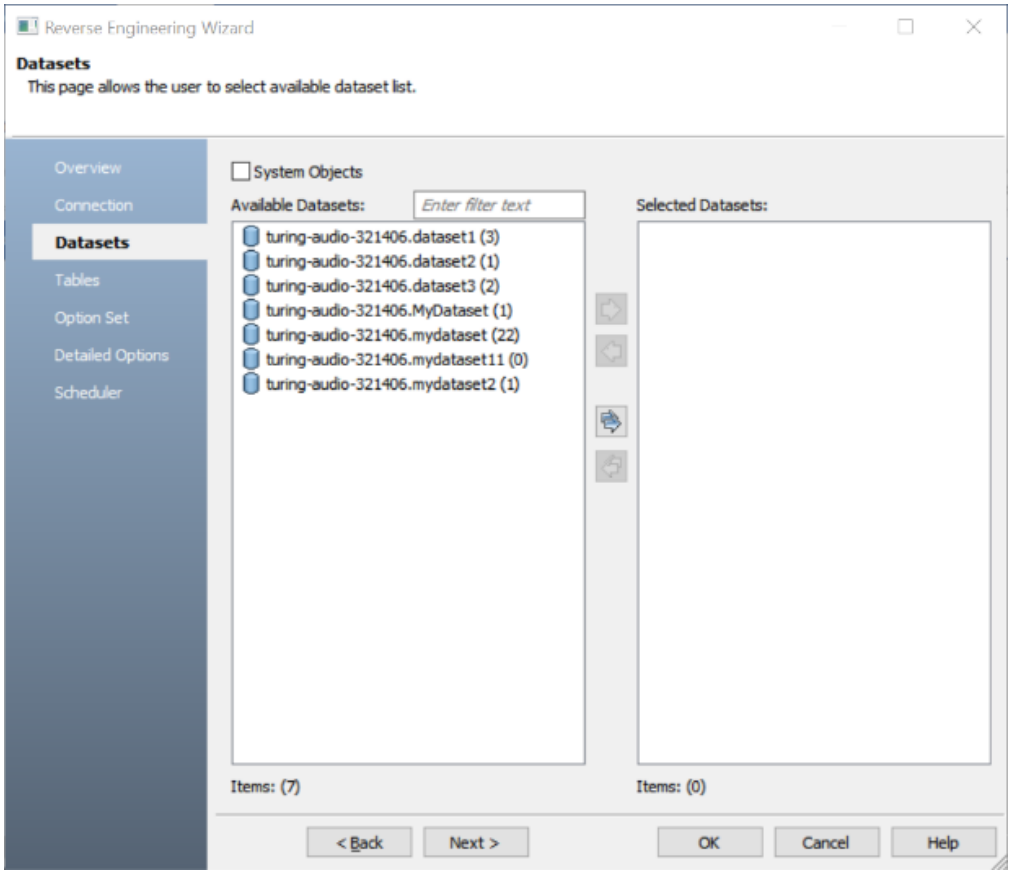
For information on creating JSON file for authentication, In Google BigQuery documentation, refer to the **BigQuery APIs** > **BigQuery API Reference** > **BigQuery API Client Libraries** > **Setting up authentication** section.

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

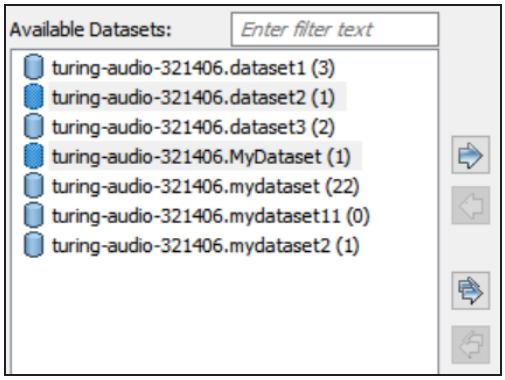


8. Click **Next**.

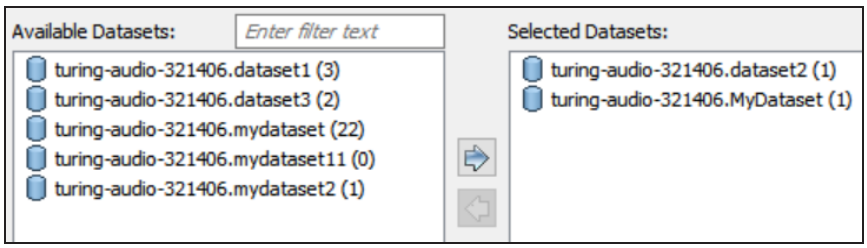
The Datasets tab appears. It displays a list of available datasets.



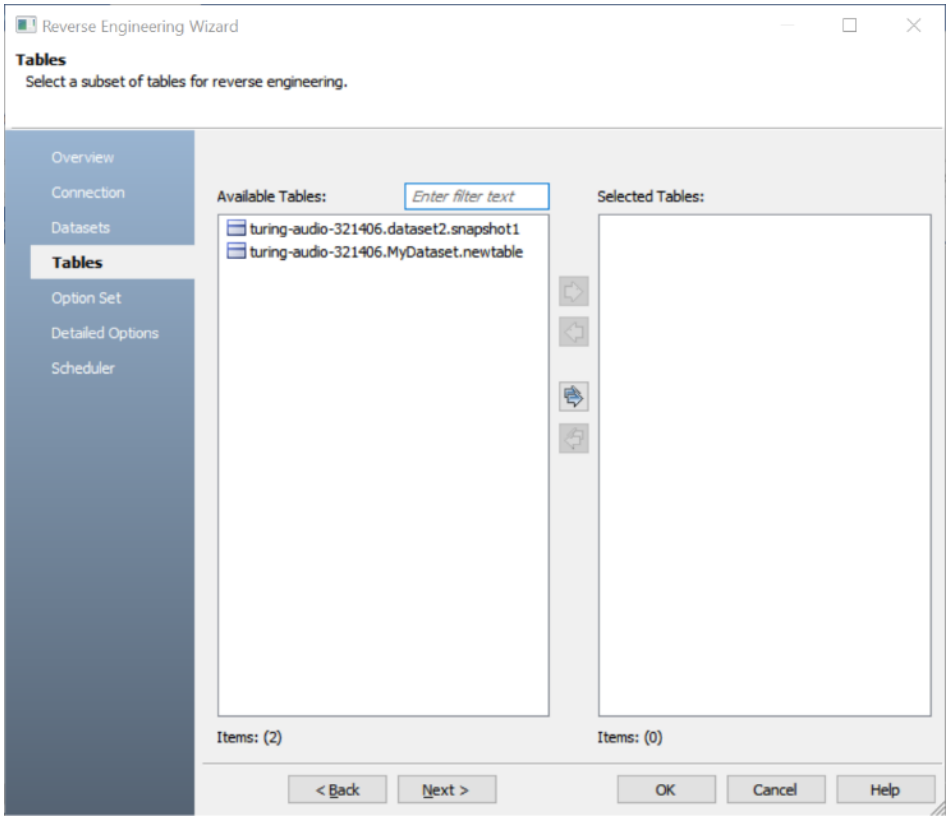
9. Under **Available Datasets**, select the datasets that you want to reverse engineer. Then, click  .



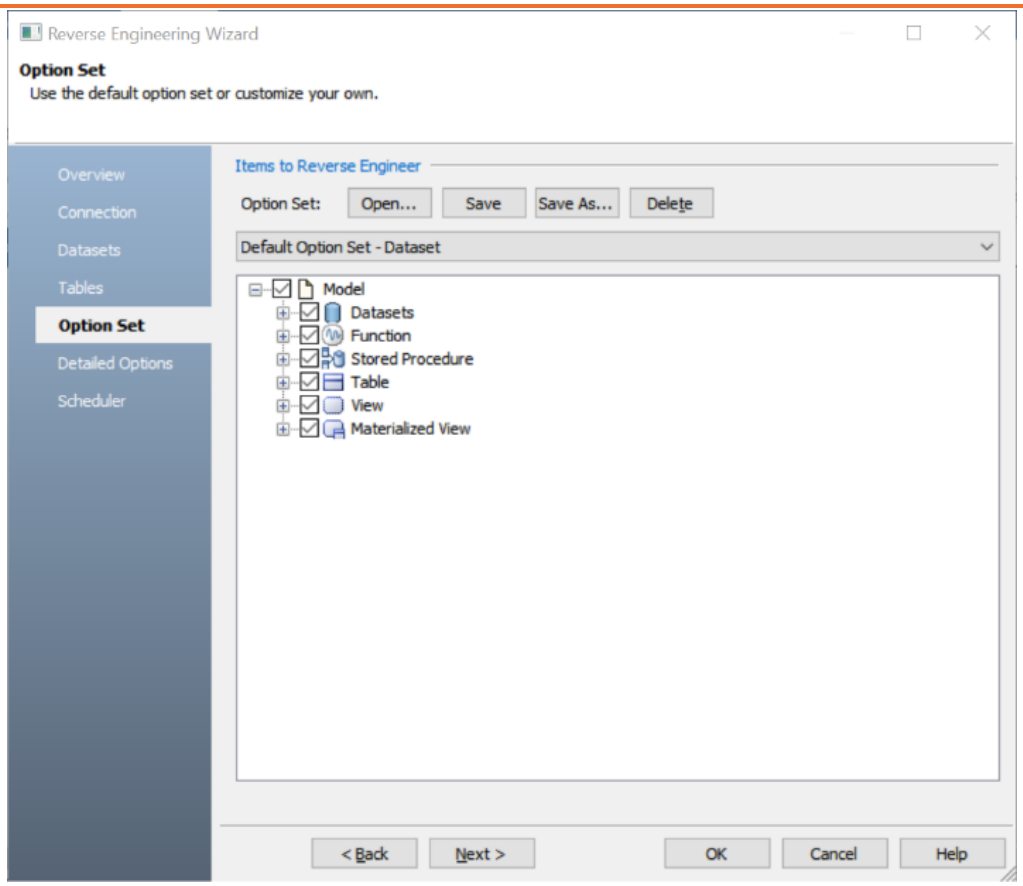
This moves the selected datasets under Selected Datasets.



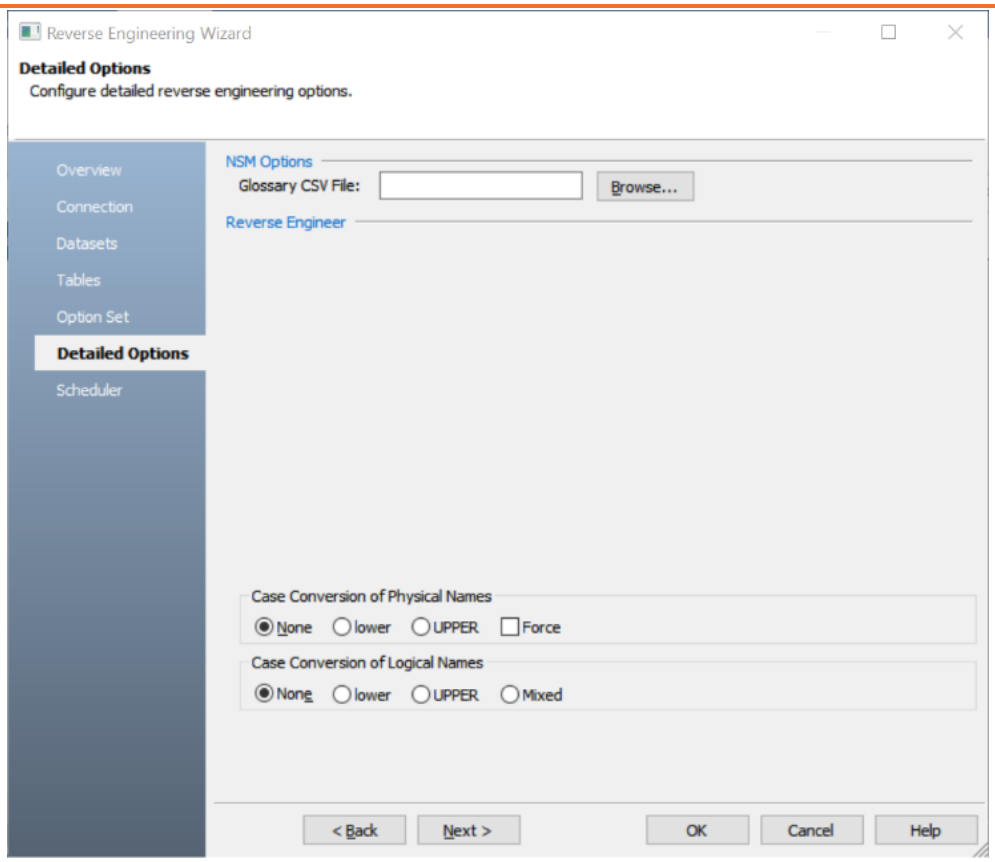
10. Click **Next**.
    The Tables tab appears. It displays a list of available tables in the datasets that you selected in step 8.



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

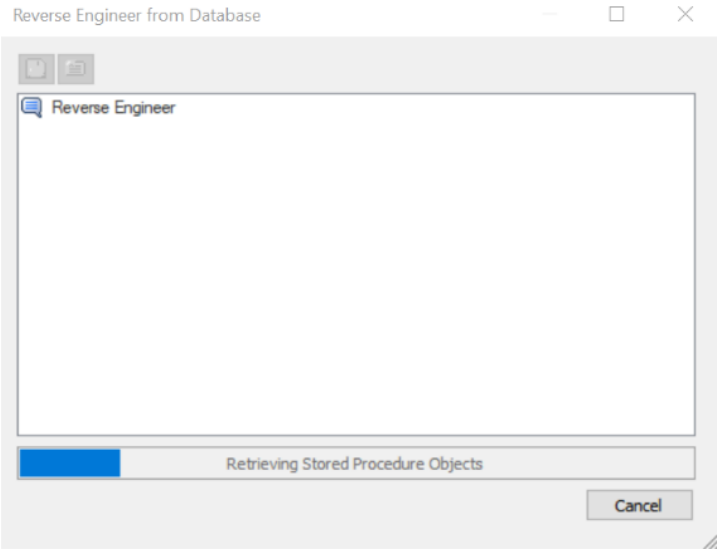


12. Click **Next**.
    The Detail Options tab appears. Set up appropriate options based on your requirement.

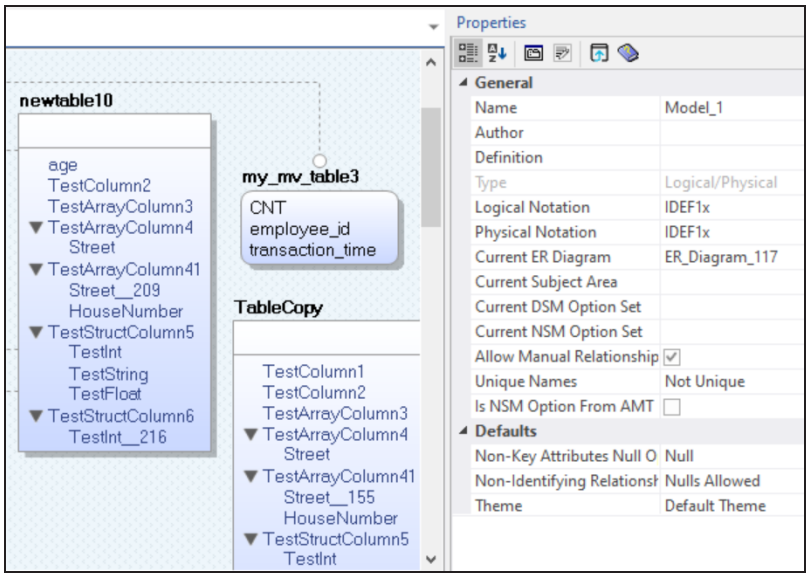

Reverse Engineering Wizard
Detailed Options
Configure detailed reverse engineering options.
Overview
Connection
Datasets
Tables
Option Set
Detailed Options
Scheduler
NSM Options
Glossary CSV File:
Browse...
Reverse Engineer
Case Conversion of Physical Names
None
lower
UPPER
Force
Case Conversion of Logical Names
None
lower
UPPER
Mixed
< Back
Next >
OK
Cancel
Help

13. 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.



Reverse Engineering Options for Google BigQuery

Following are the reverse engineering options for Google BigQuery.

Overview

Parameter	Description	Additional Information
Reverse Engineer From	Specifies whether you want to reverse engineer from a script or dataset	**Datasets**: Indicates that the model is reverse engineered from datasets **Script File**: Indicates that the model is reverse engineered from a script
File	Specifies the script file location	This option is available when Script File is selected.

Connection

Parameter	Description	Additional Information
Connection Method	Specifies the type of connection you want to use. **Connection String** connects to your cluster using a connection string.	
Connection String	Specifies the path to the secure connect JSON file in the following format: *C:\<file name>.json*	This option is available when Connection Method is set to Connection String

For information on creating JSON file for authentication, In Google BigQuery documentation, refer to the **BigQuery APIs** > **BigQuery API Reference** > **BigQuery API Client Libraries** > **Setting up authentication** section.

Datasets

Parameter	Description	Additional Information

System Objects	Specifies whether system objects are included under the Available Datasets	
Available Datasets	Specifies a list of available datasets	
Selected Datasets	Specifies a list of selected datasets for reverse engineering	

Tables

Parameter	Description	Additional Information
Available Tables	Specifies a list of available tables	
Selected Tables	Specifies a list of selected tables for reverse engineering	

Option Sets

Parameter	Description	Additional Information
Option Set	Specifies the option set template for reverse engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save the configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at some external location. **Delete**: Use this option to delete an option set.
<Option Set Name>	Specifies the objects to be reverse engineered according to the selected option set. You can edit this list.	

Detailed Options

Parameter	Description	Additional Information
NSM Options	Specifies the naming standard glossary file in the .CSV format	
Case Conversion of Physical Names	Specifies how the case conversion of physical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Force**: Indicates whether the physical name property of all the logical/physical models is overridden. If this option is enabled, the logical/physical link is broken between the logical and physical name. If this option is not enabled, all logical and physical names are set to the same value after the process completes.
Case Conversion of Logical Names	Specifies how the case conversion of logical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Mixed**: Indicates that the mixed-case logical names are preserved

Scheduler

Parameter	Description	Additional Information
Model	Specifies the location and name of the reverse engineered model	For example: C:\Scheduler\<Model Name>.erwin When you schedule a job on a remote server, ensure the model path is same for remote and local server.
Mart Folder	Specifies the location or library in your mart where the	To use this option, ensure that you are connected to a mart. For more information, refer to

	reverse engineered model is saved	the Connecting to Mart topic.
Complete Compare	Specifies whether the Complete Compare (CC) process should run while reverse engineering	
Output File	Specifies the location of the CC output file generated	
File	Specifies that the target model location is on the local system	
Mart	Specifies that the target model location is in the mart	
Using Latest Version	Specifies whether the target model is the latest version of the model in the mart	This option is available only when Mart is selected.
Save To Mart	Specifies whether the reverse engineered model is saved to the mart	This option is available only when Using Latest Version is selected.
Target Model	Specifies the location of the target model for CC	
Option Set	Specifies the option set that is used for CC	**Advanced Default Option Set**: Indicates that all erwin DM metadata is included. CC works slowest with this option. **Speed Option Set**: Indicates that only the essential metadata is included. CC works the fastest with this option set. **Standard Default Option Set**: Indicates that standard metadata is included. CC works fast with this option set compared to the Advanced option set.

Forward Engineering Models

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

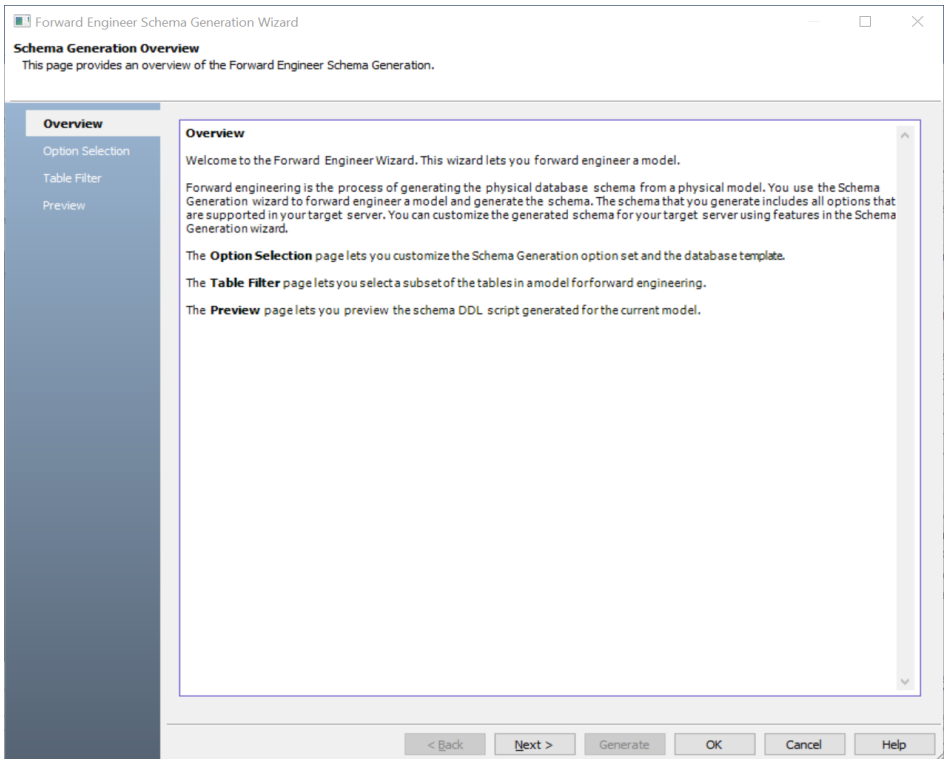
This topic walks you through the steps to forward engineer a Google BigQuery model. For detailed description of forward engineering options, refer to the Forward Engineering Options topic.

To forward engineer a Google BigQuery model:

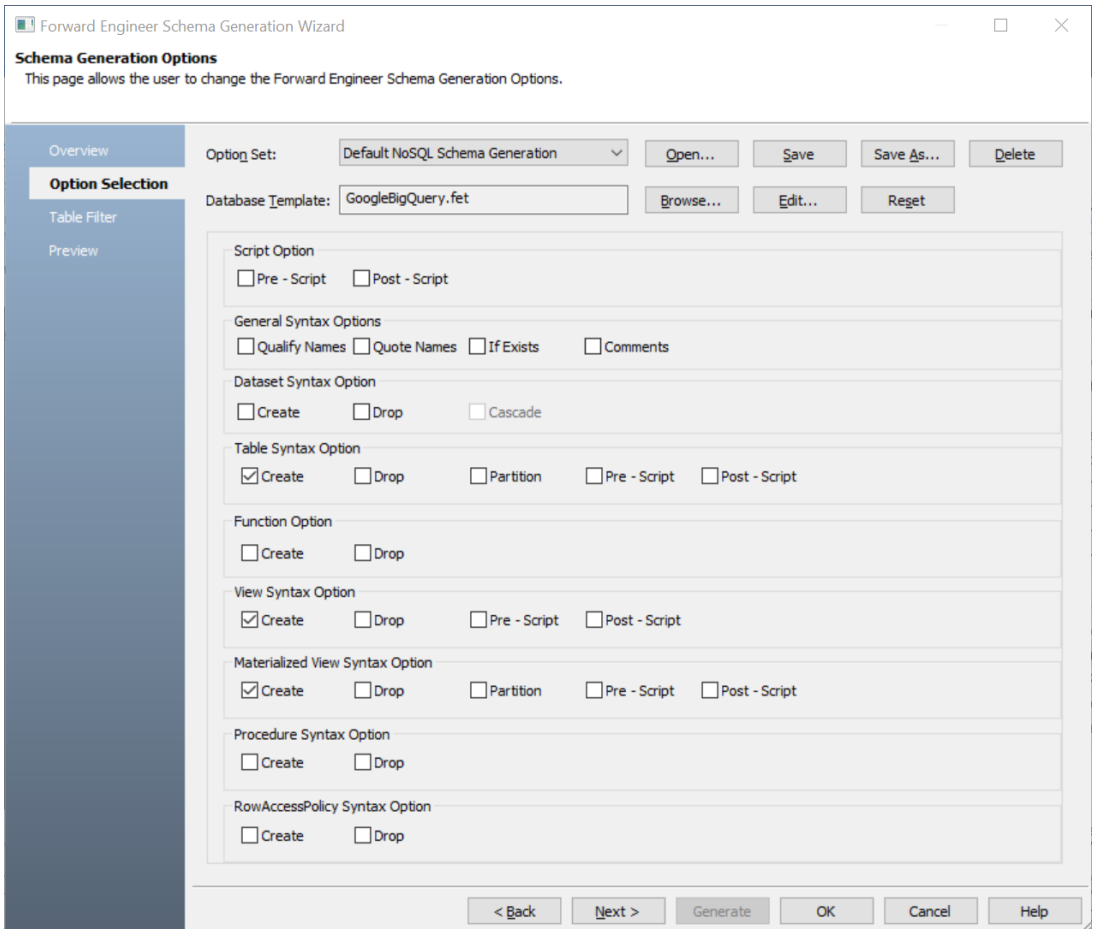
1. Open your Google BigQuery model in erwin Data Modeler (DM).

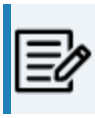
   Ensure that you are in the Physical mode.

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



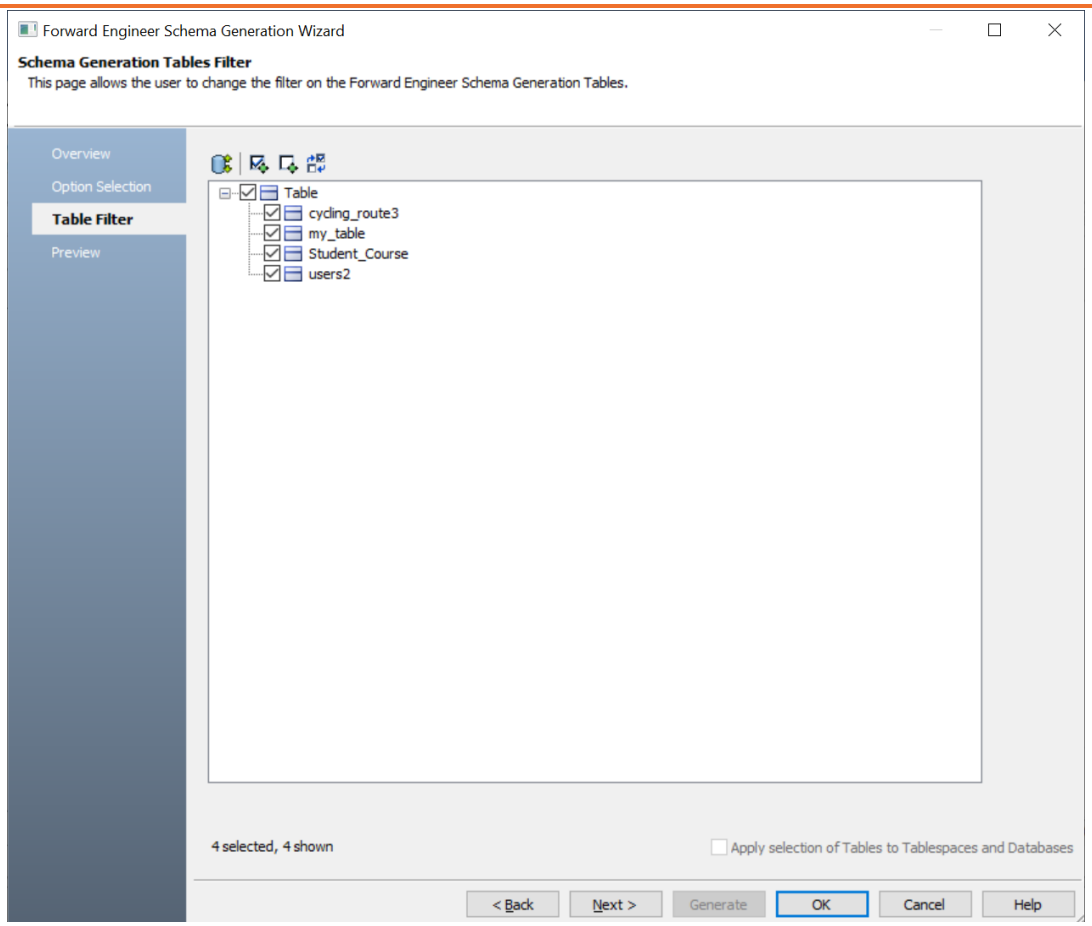
3. Click **Option Selection**.
   The Option Selection tab displays the default option set. Clear the **Drop** check boxes and select other syntax check boxes as required.





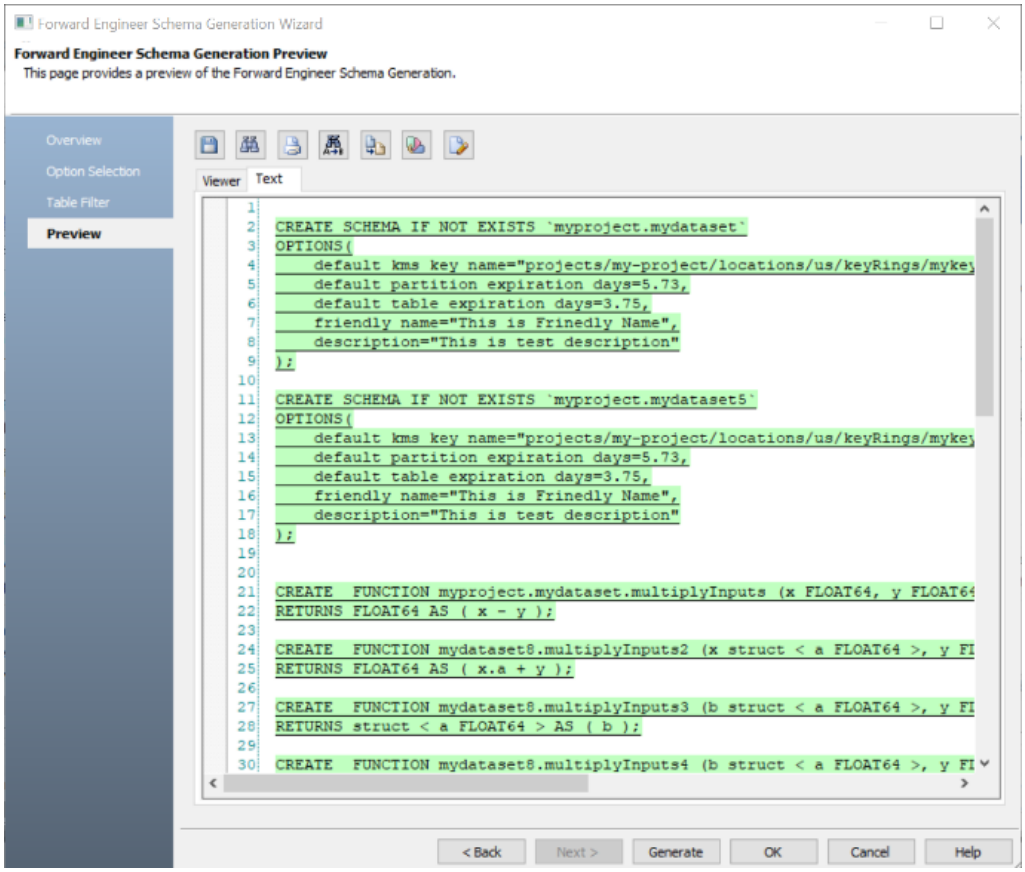
Ensure that you keep a backup of your original table before making any changes.

4. Click **Next**.
   The Table Filter tab appears. It displays a list of tables available in your model.



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

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



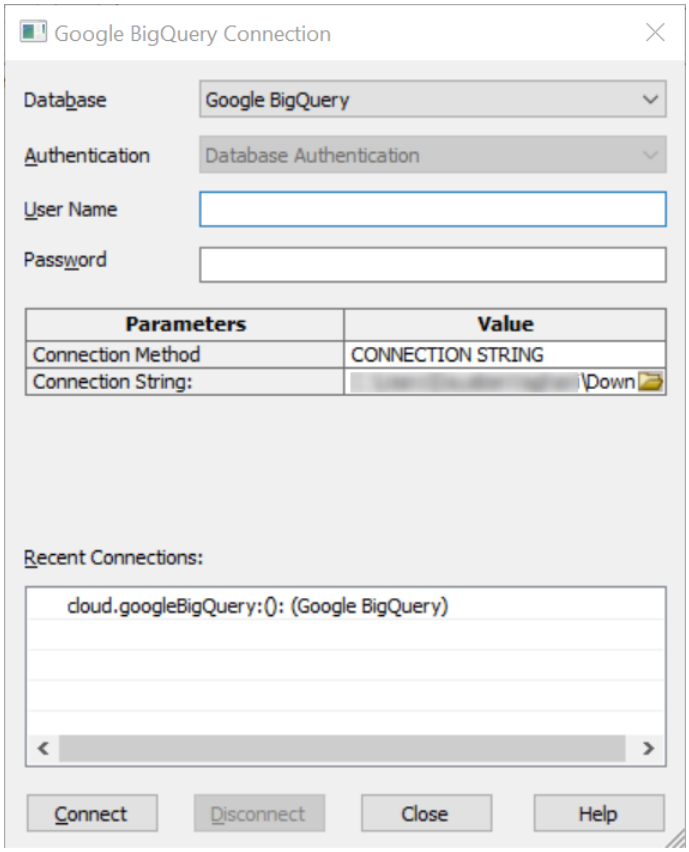
Use the following options:

- **Copy** ( ): Use this option to copy the script.
- **Save** ( ): Use this option to save the generated script in the ERS, SQL, or DDL format.
- **Search** ( ): Use this option to search through the generated schema.
- **Print** ( ): Use this option to print the generated schema.
- **Replace** ( ): Use this option to find and replace in the generated schema.

- **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.
- **Error Check** (): Use this option to run an error check. Based on the results, you can correct the generated script.

7. Click **Generate**.

   The Google BigQuery Connection screen appears.

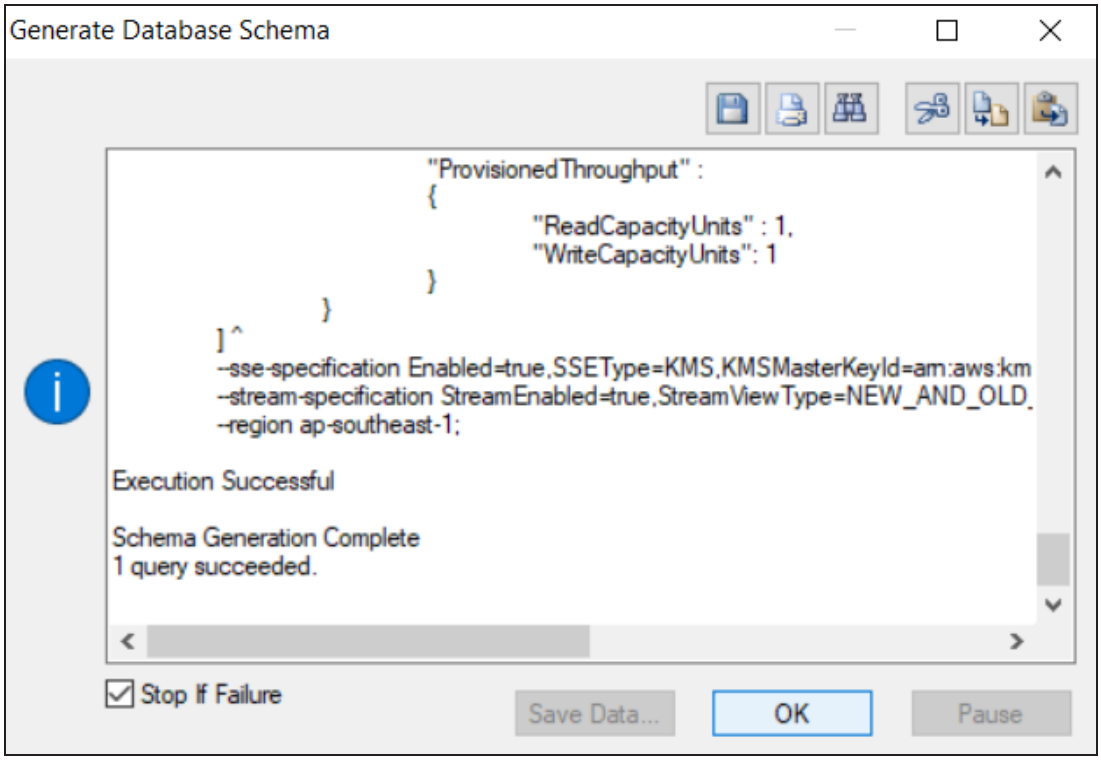


8. Enter username, password, and appropriate connection parameters to connect the required database. Then, click **Connect**.

> Objects in your model move to the database mentioned on the Google BigQuery Connection screen irrespective of the databases defined on

the object editor screens. If you want to retain objects in their respective databases as defined on the object editor screens, keep the database parameter blank.

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



Forward Engineering Options for Google BigQuery

Following are the forward engineering options for Google BigQuery.

Option Selection

Parameter	Description	Additional Information
Option Set	Specifies the option set template for forward engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save a configured option set.

		Save As: Use this option to save an option set either in the model or in the XML format at an external location. **Delete**: Use this option to delete an option set.
Database Template	Specifies the database template for controlling schema generation	**Browse**: Use this option to browse and select a database template. **Edit**: Use this option to edit a template in the Template Editor. **Reset**: Use this option to reset the Database Template option.
Script Option	Specifies the script option for schema generation	**Pre-Script**: Indicates whether pre-scripts attached to the schema are executed **Post-Script**: Indicates whether the post-scripts attached to the schema are executed
General Syntax Options	Specifies the general syntax options for schema generation	**Qualify Names**: Indicates whether the Qualify names syntax for general properties is executed **Quote Names**: Indicates whether the Quote names syntax for general properties is executed **If Exists**: Indicates whether the If exists syntax for general properties is executed **Comments**: Indicates whether comments are included in the schema
Dataset Syntax Option	Specifies the dataset syntax options for schema generation	**Create**: Indicates whether the Create syntax for datasets is executed **Drop**: Indicates whether the Drop syntax for datasets is executed **Cascade**: Indicates whether the Cascade syn-

		tax for datasets is executed
Table Syntax Option	Specifies the table syntax options for schema generation	**Create**: Indicates whether the Create syntax for tables is executed **Drop**: Indicates whether the Drop syntax for tables is executed **Partition**: Indicates whether the Partition syntax for tables is executed **Pre-Script**: Indicates whether pre-scripts attached to the tables are executed **Post-Script**: Indicates whether the post-scripts attached to the tables are executed
Function Option	Specifies the function syntax options for schema generation	**Create**: Indicates whether the Create syntax for functions is executed **Drop**: Indicates whether the Drop syntax for functions is executed
View Syntax Option	Specifies the view syntax options for schema generation	**Create**: Indicates whether the Create syntax for views is executed **Drop**: Indicates whether the Drop syntax for views is executed **Pre-Script**: Indicates whether pre-scripts attached to the views are executed **Post-Script**: Indicates whether the post-scripts attached to the views are executed
Materialized View Syntax Option	Specifies the materialized view syntax options for schema generation	**Create**: Indicates whether the Create syntax for materialize views is executed **Drop**: Indicates whether the Drop syntax for materialized views is executed **Partition**: Indicates whether the Partition syntax for materialize views is executed **Pre-Script**: Indicates whether pre-scripts

		attached to the materialize views are executed **Post-Script**: Indicates whether the post-scripts attached to the materialize views are executed
Procedure Syntax Option	Specifies the procedure syntax options for schema generation	**Create**: Indicates whether the Create syntax for procedures is executed **Drop**: Indicates whether the Drop syntax for procedures is executed
RowAccesPolicy Syntax Option	Specifies the row access policy syntax options for schema generation	**Create**: Indicates whether the Create syntax for row access policies is executed **Drop**: Indicates whether the Drop syntax for row access policies is executed

Table Filter

Parameter	Description	Additional Information
Tables	Specifies the selected tables for schema generation	
Display either Logical Names or Physical Names		**Logical Names**: Indicates that only logical names of the records are included in the generated schema **Physical Names**: Indicates that only physical names of the records are included in the generated schema **Physical Names, show owner**: Indicates that physical names and owners of the records are included in the generated schema **Physical Names, show owner using User**: Indicates that the physical names and owners of the records are included in the generated schema.

		Owners of the records are displayed using User.
Select all of the items in the list	Use this option to select all the records in the list.	
Unselect all of the items in the list	Use this option to clear all the records.	
Select all unselected items, and unselect all selected items	Use this option to select all the unselected records and clear all the previously selected records.	

Ensure that you keep a backup of your original table before making any changes.

Preview

Parameter	Description	Additional Information
Text	Displays the schema in the text editor	**Save**: Use this option to save the generated schema. **Search**: Use this option to search through the generated schema. **Print**: Use this option to print the generated schema. **Replace**: Use this option to find and replace text in the generated schema. **Copy**: Use this option to copy the selected text in the schema. **Text Options**: Use this option to edit window settings, fonts, syntax color. **Git**: Use this option to commit the FE script to a Git repository.

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.

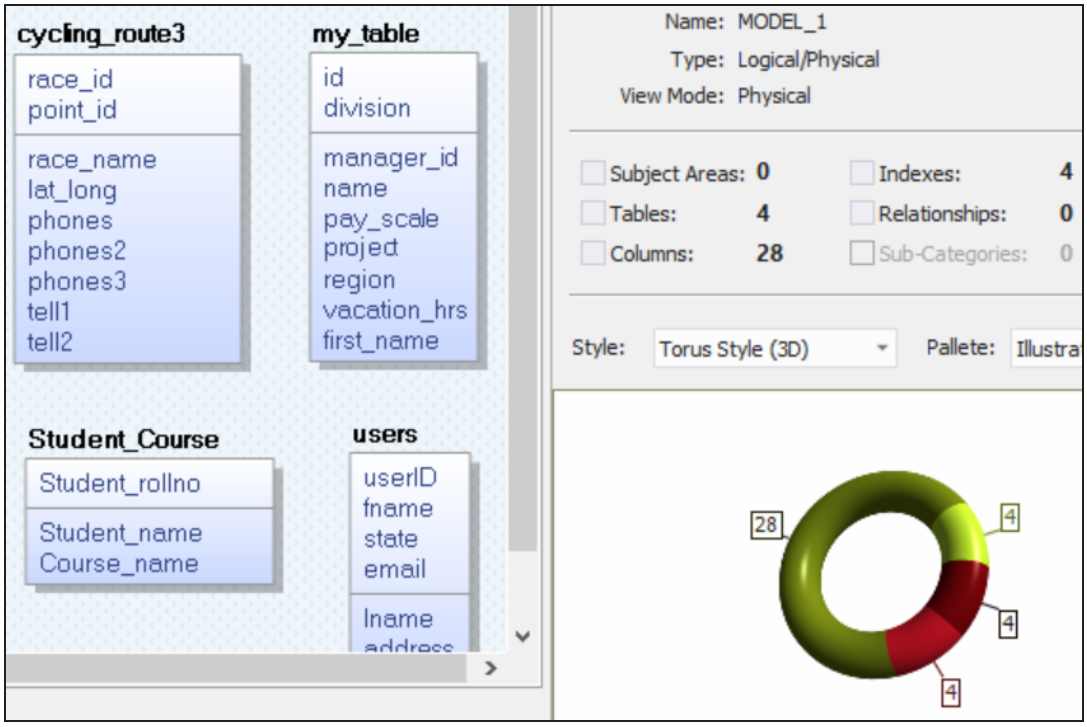
This topic walks you through the steps to compare a Google BigQuery model with database.

To compare models with database:

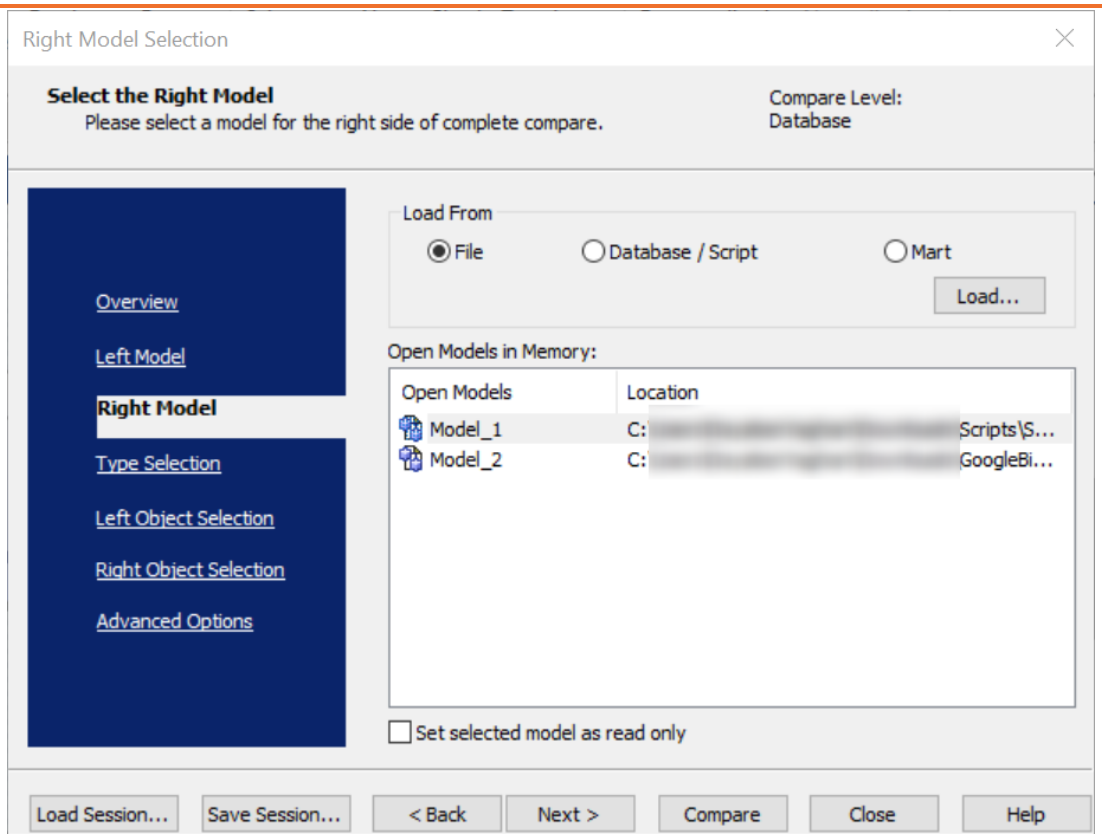
1. Open your Google BigQuery model.

   Ensure that you are in the Physical mode.

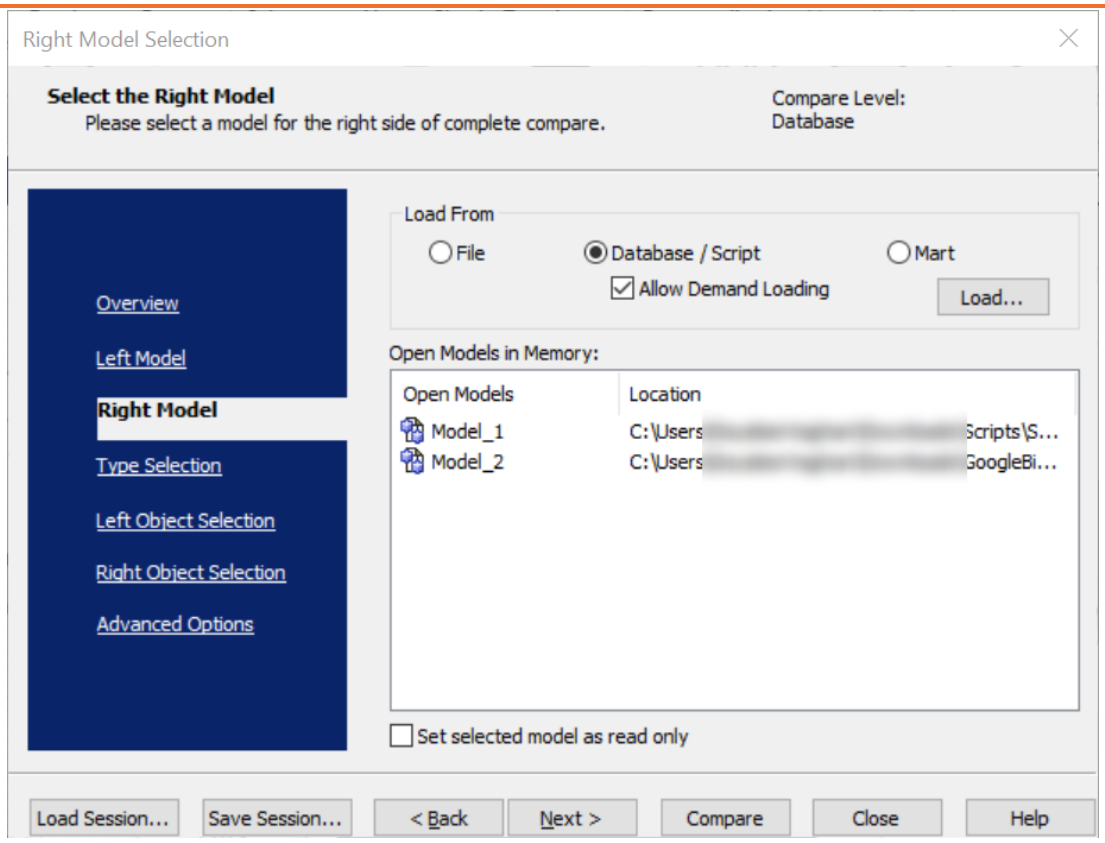
   For example, the following image uses a Google BigQuery model with 4 tables.



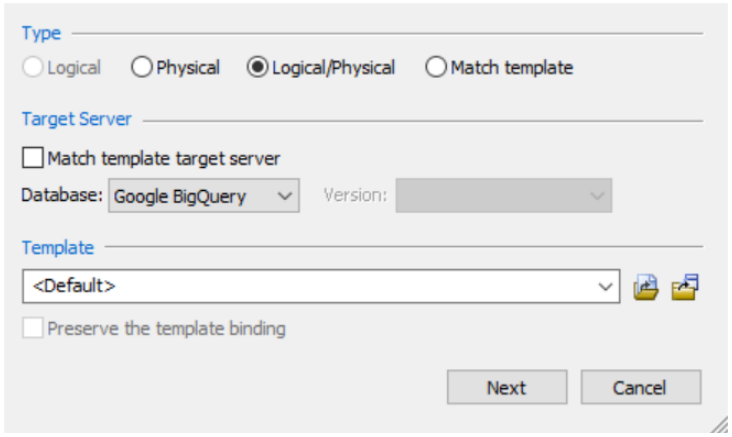
2. Click **Actions** > **Complete Compare**.
   By default, the Complete Compare wizard assigns the open model as the Left Model. Hence, the Right Model tab appears.



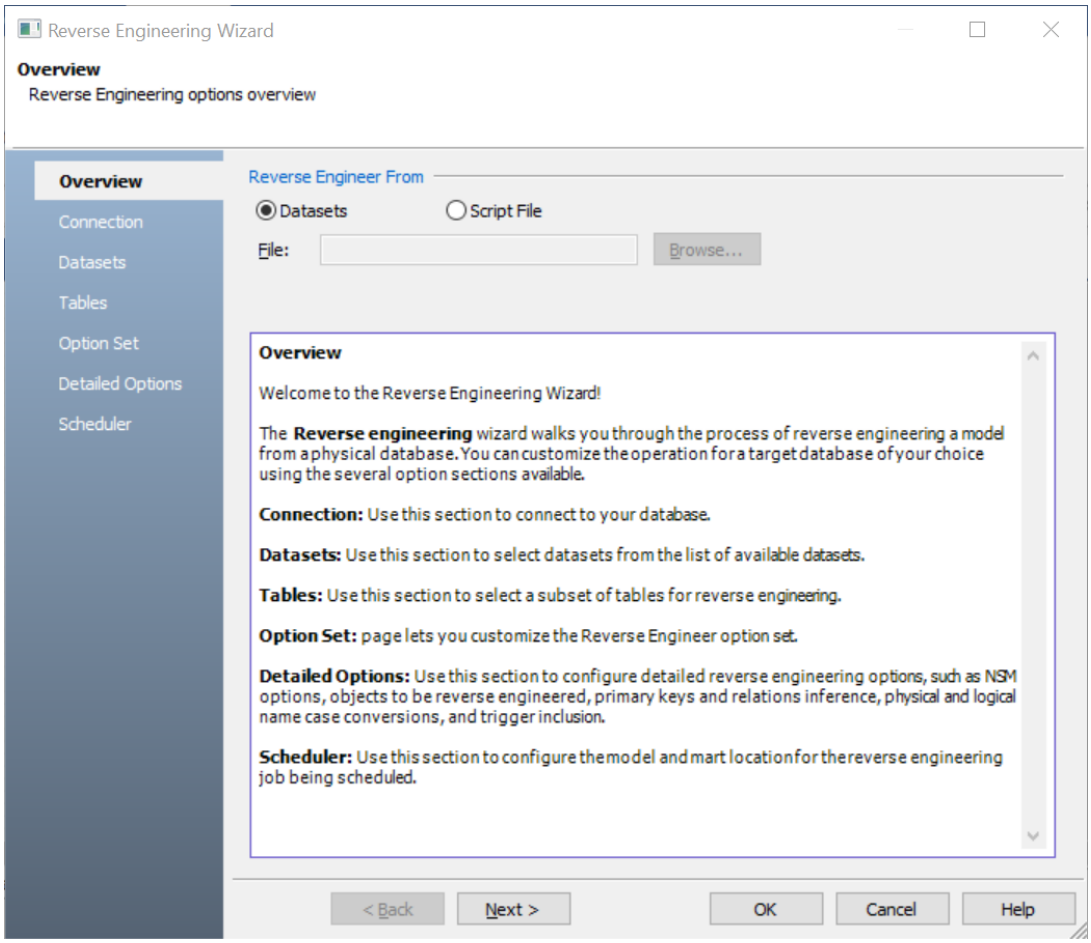
3. Click **Database/Script**.
   By default, the Allow Demand Loading option is selected.



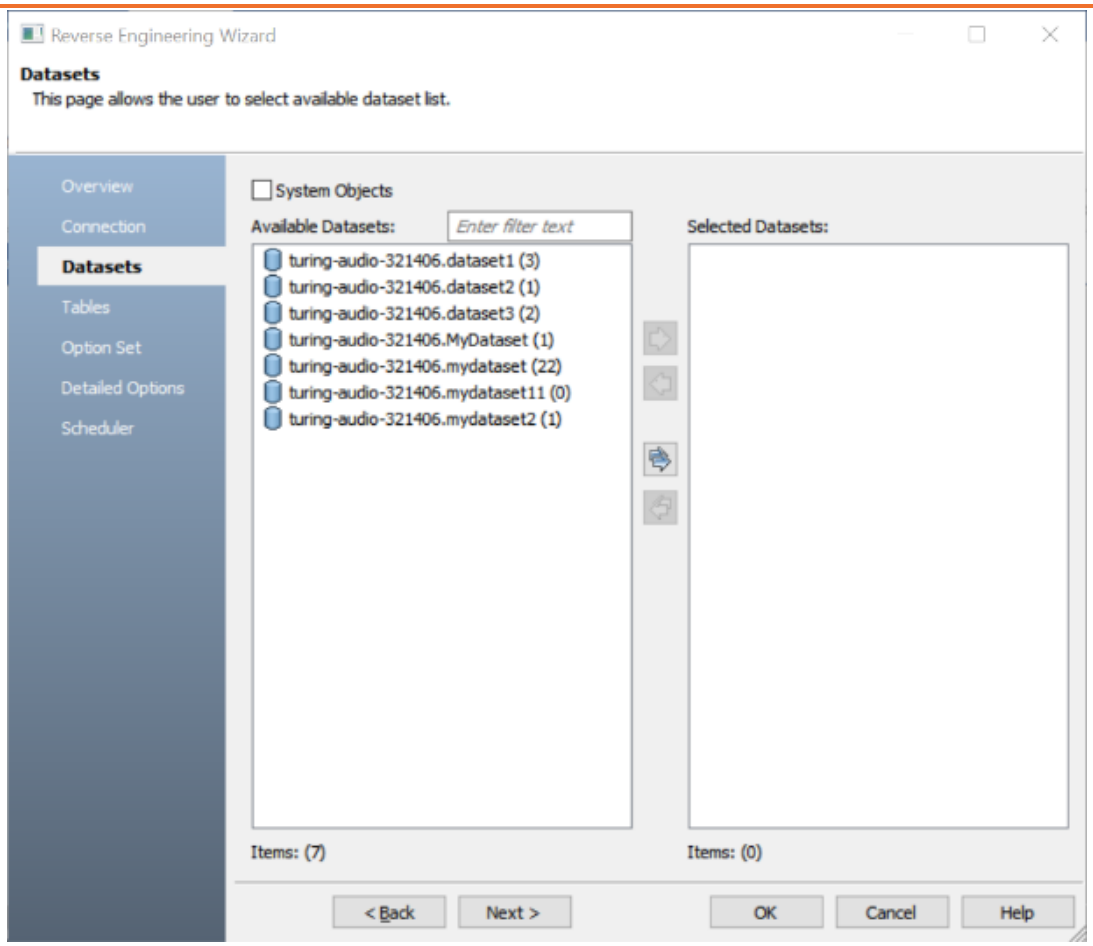
4. Click **Load**.
   The New Model dialog box appears. This starts the reverse engineering process to pull a model from the database to compare.



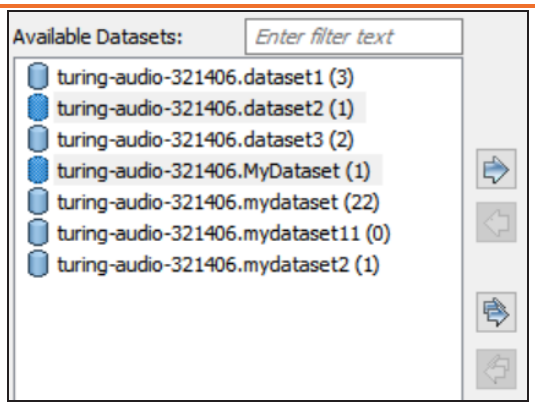
5. Ensure that the Database is set to Google BigQuery. Then, click **Next**.
   The Reverse Engineering Wizard appears.



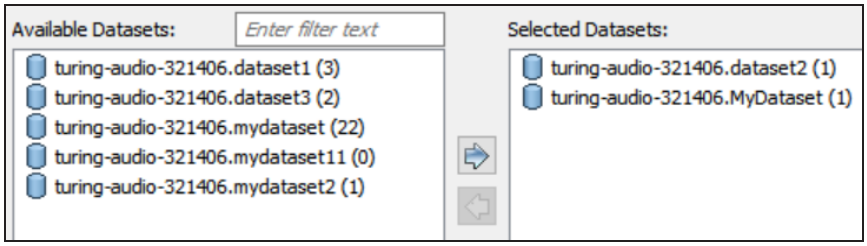
6. Click **Datasets**. Then, click **Next**.
   The Connection tab appears. Use this section to connect to the database from which you want to reverse engineer the model.
7. After connection is established, click **Next**.
   The Datasets tab appears. It displays a list of available datasets.



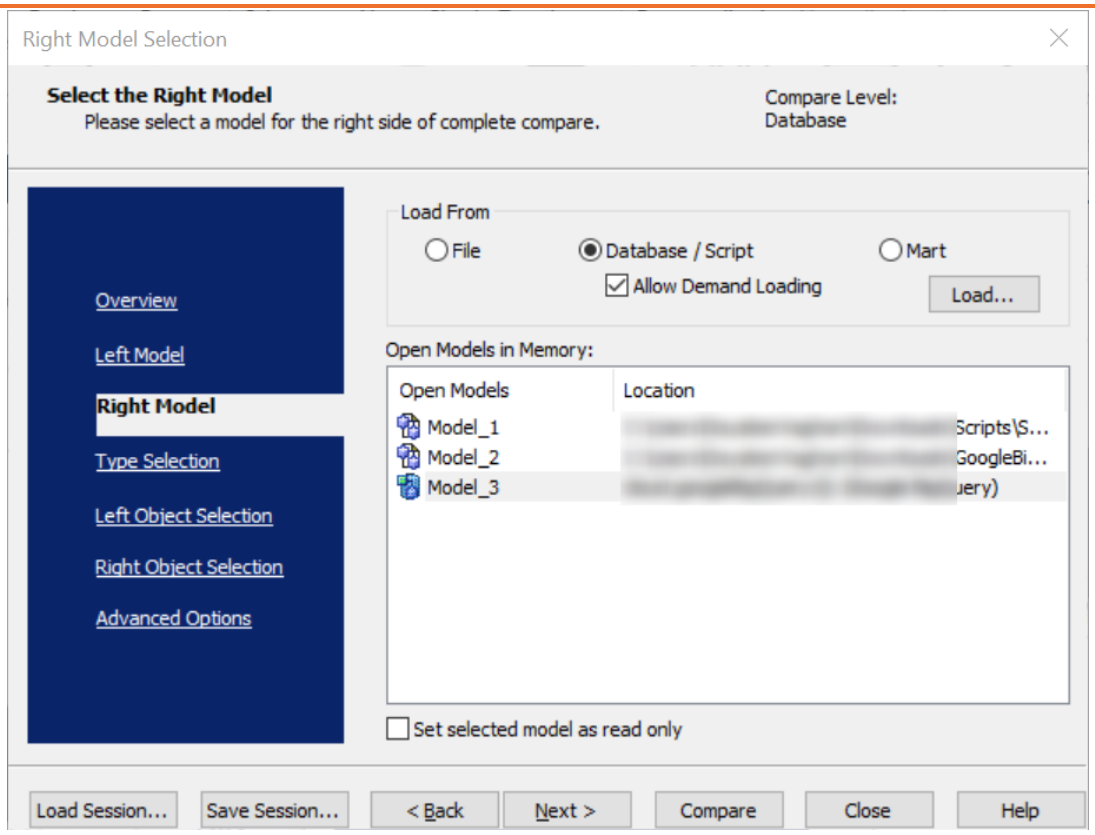
8. Under **Available Datasets**, select the datasets that you want to reverse engineer. Then, click  .
   This moves the selected datasets under Selected Datasets .



9. Click **Next** and on the Tables tab , click  .
   This selects all the available tables.

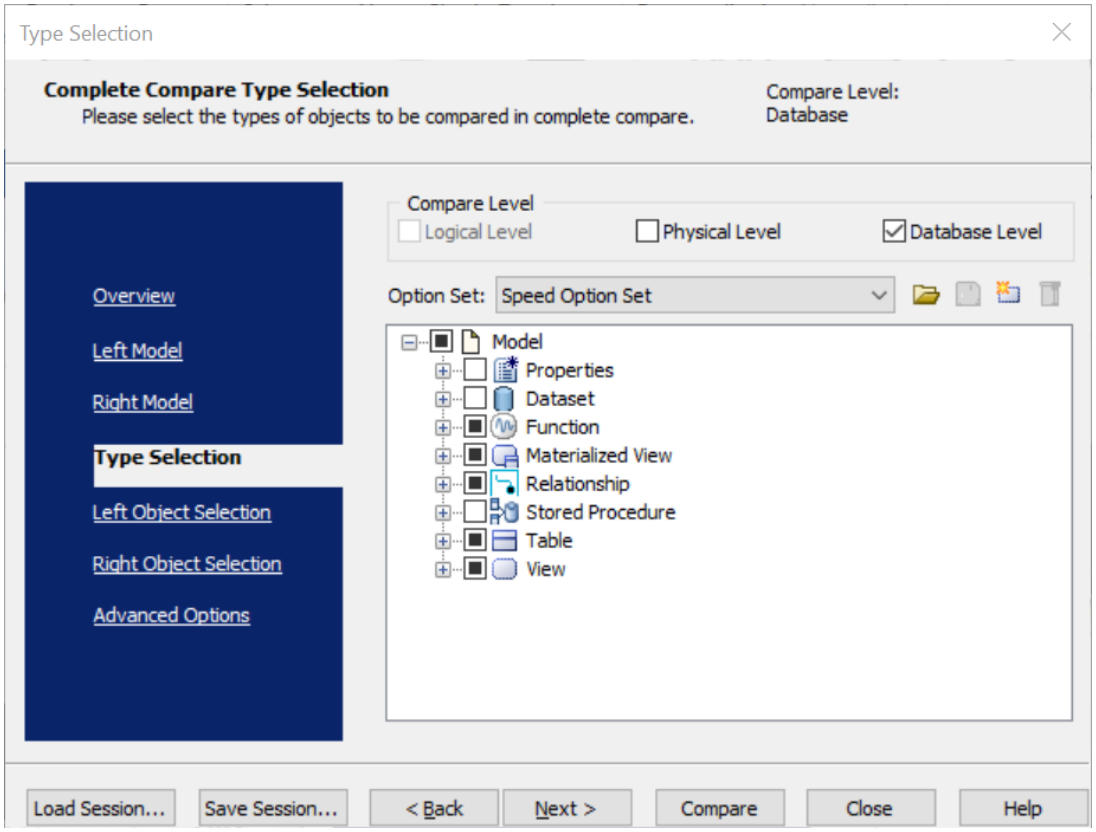


10. Click **Next** and on the Option Set tab, keep the default configuration.
11. Click **Next** and on the Detail Options tab, keep the default configuration.
12. Click **OK**.
    The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse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



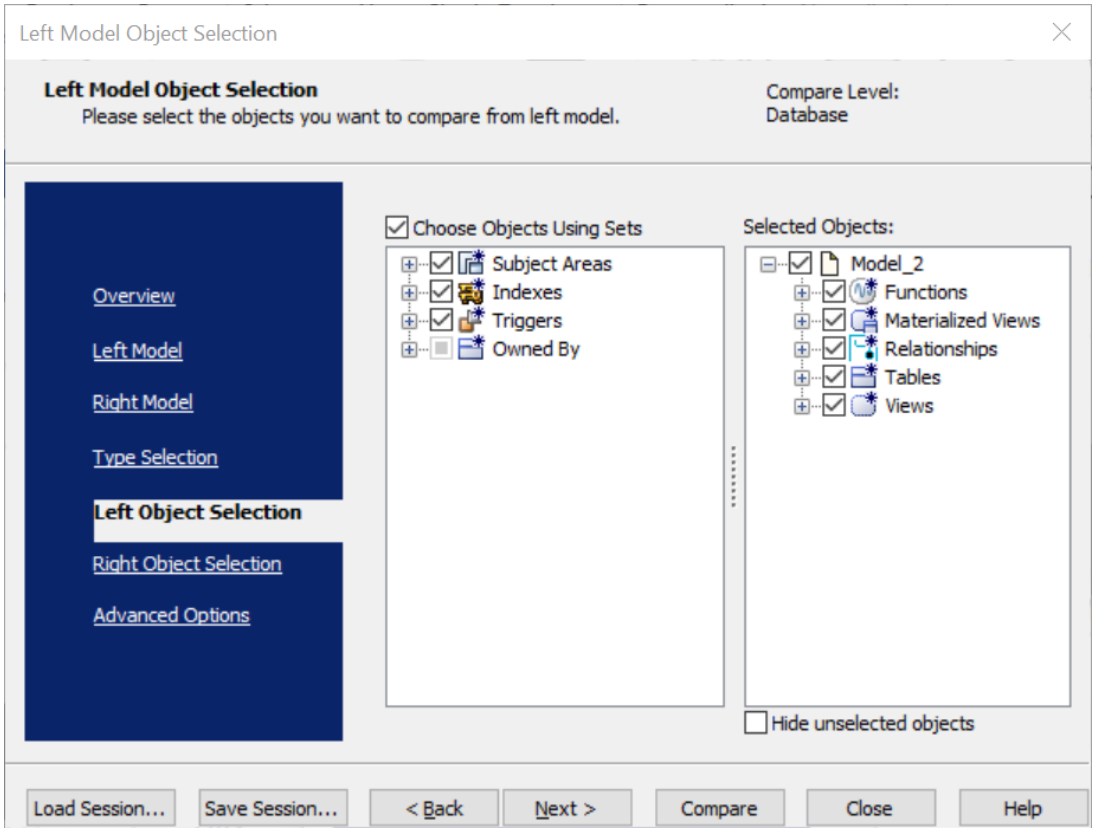
13. Click **Next** and on the Type Selection tab, select the appropriate options.

For example, the following image shows the default options.



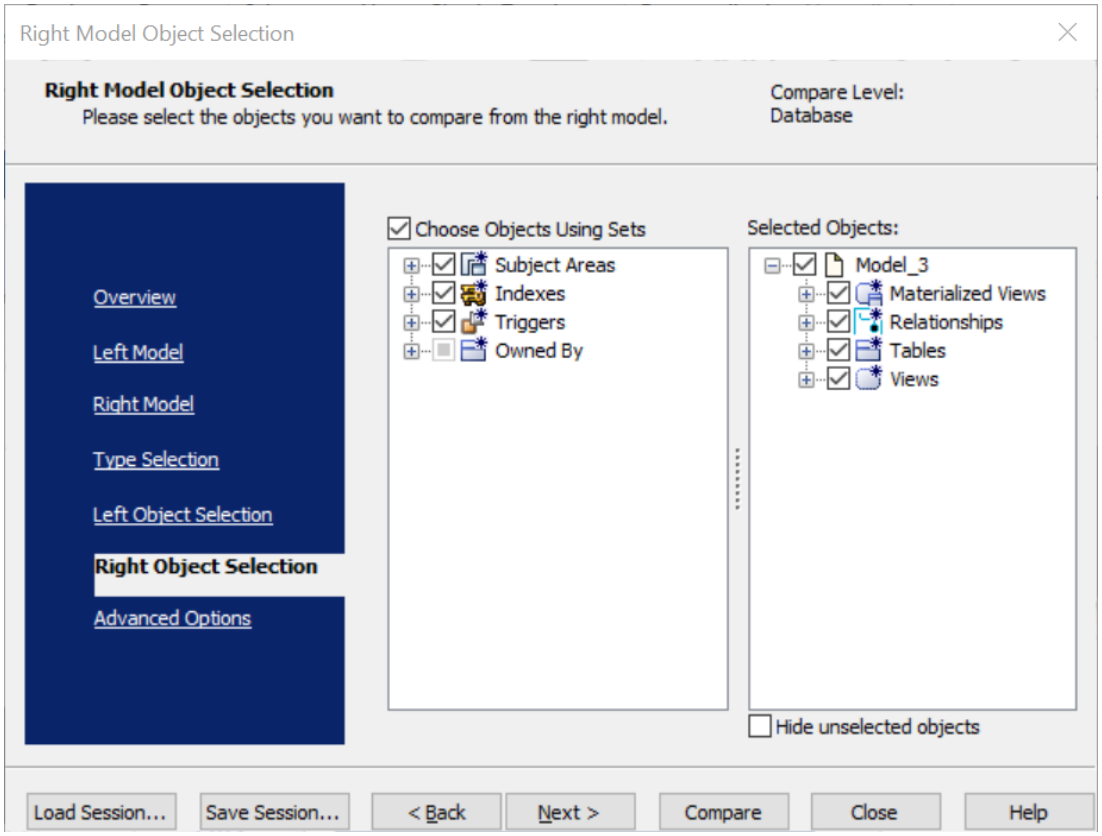
14. Click **Next** and in the Left Object Selection tab, select the appropriate options.

For example, the following image shows the default options.

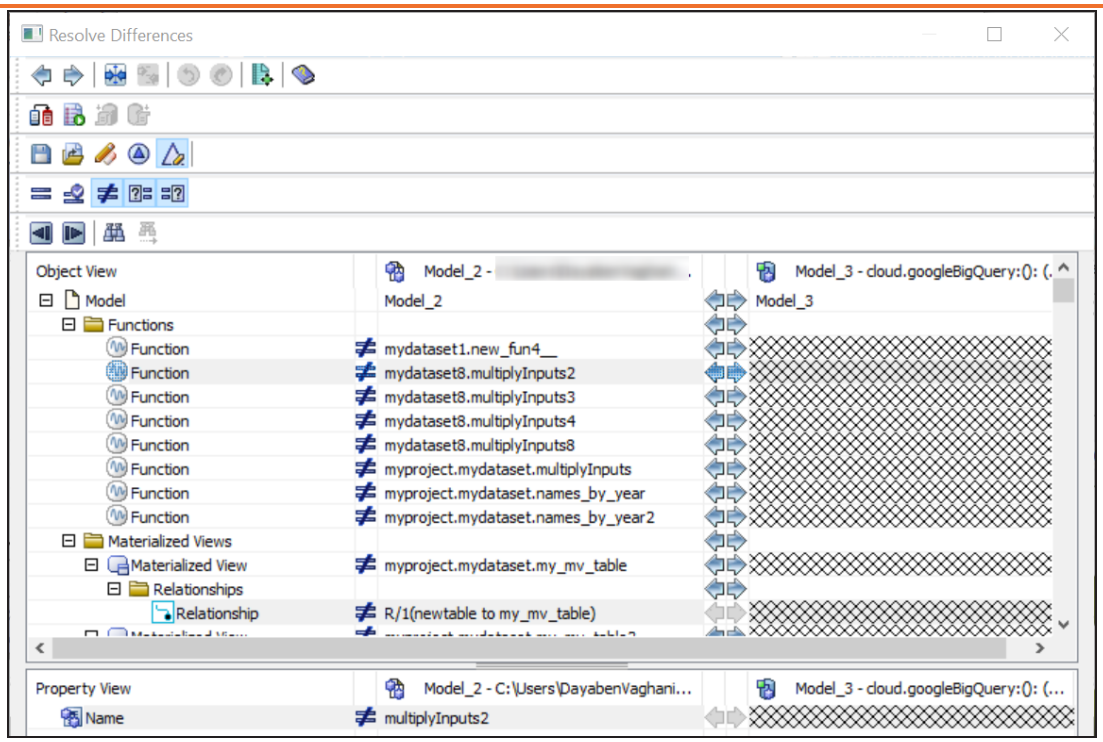


15. Click **Next** and on the Right Object Selection tab, select the appropriate options.

For example, the following image shows the default options.



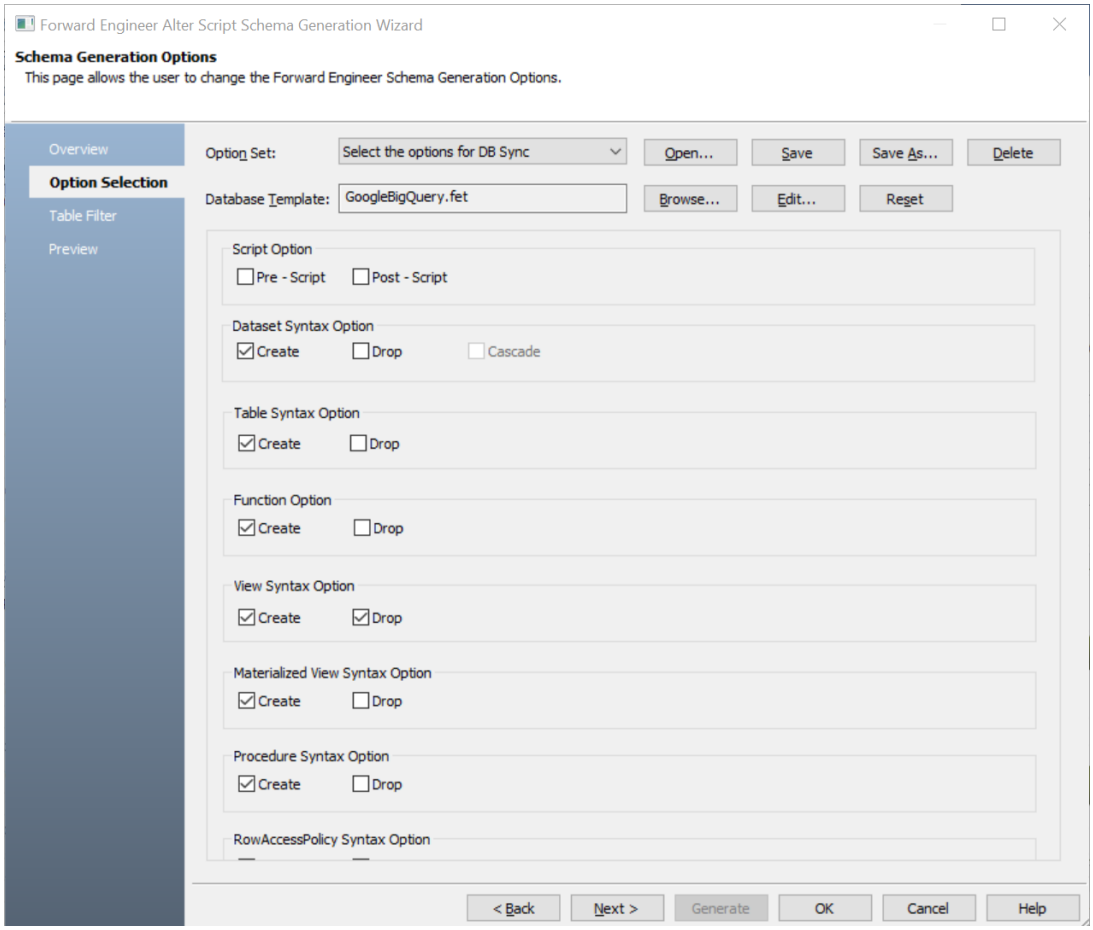
16. Click **Compare**.
    The comparison process runs, and the Resolve Differences dialog box appears. It displays the differences between your model and database.
    For example, the following image shows that the mydataset8.multiplyInputs3 function is available in your model but not in the database.



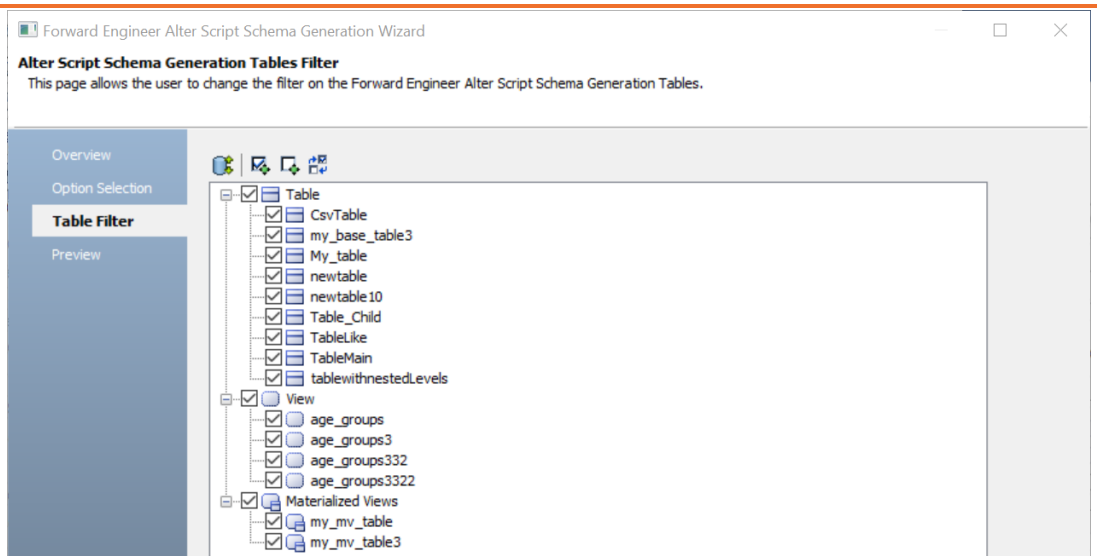
Select the Function and click . This will move the mydataset8.multiplyInputs3 table to the right model (from the database). Similarly, resolve other differences.

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

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



19. Click **Table Filter** and select or verify the tables to be included in the forward engineering script.



20. Click **Preview** to view and verify the alter script.
21. Click **Generate** and connect to your Google BigQuery 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**.

Migrating Relational Models to Google BigQuery Models

You can convert and migrate your relational models to Google BigQuery 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 Google BigQuery model.

Ensure that you keep a backup of your original models.

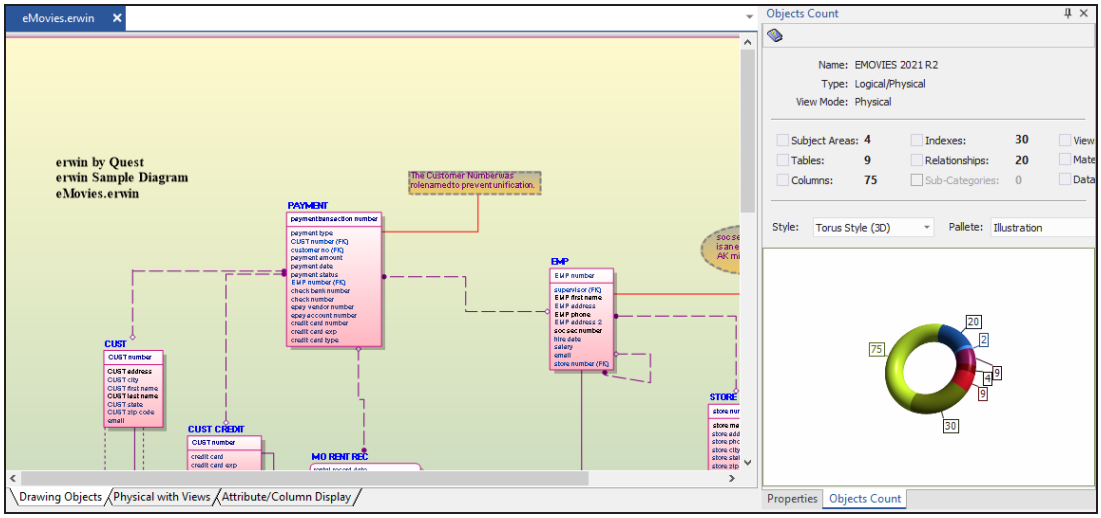
Migration by Changing the Target Database

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

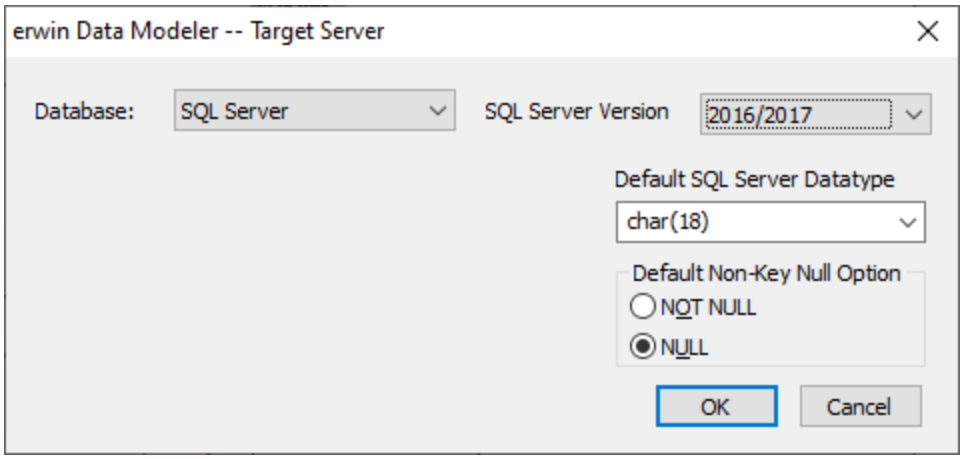
1. Open your relational model.

   Ensure that you are in the Physical mode.

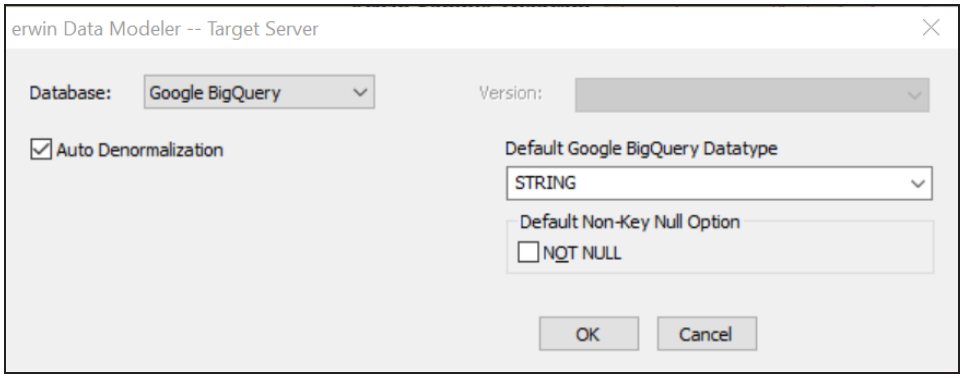
   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.



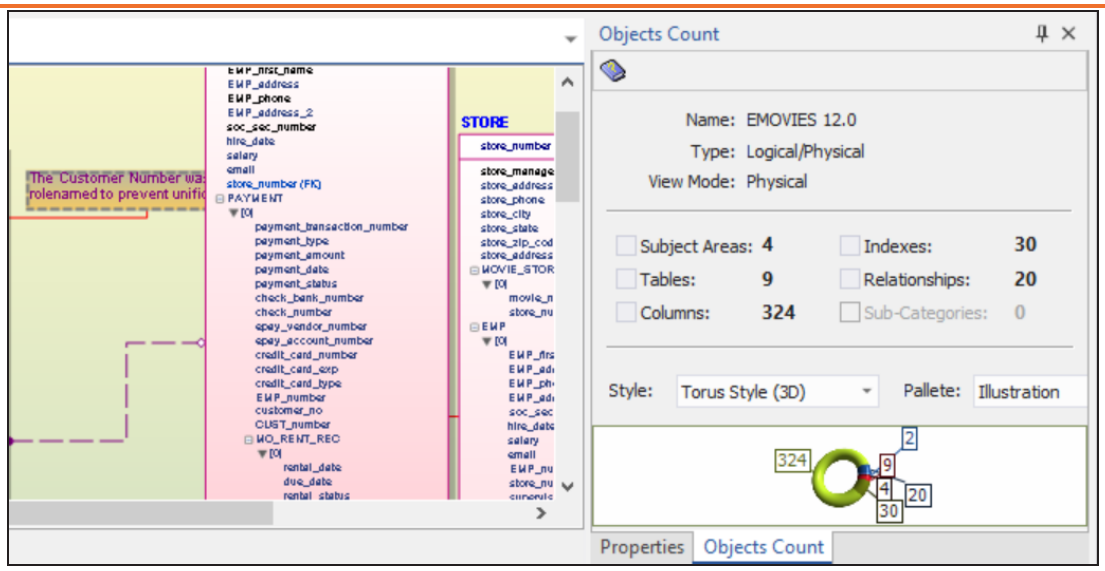
2. On the ribbon, click **Actions** > **Target Database** or on the status bar, click the database name.
   The erwin Data Modeler -- Target Server screen appears.



3. In the **Database** drop-down list, select Google BigQuery.



4. Click **OK**.
   Once the conversion is complete, the existing model is migrated to a Google BigQuery model.



In the **Objects Count** pane, note that number of columns has increased.

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

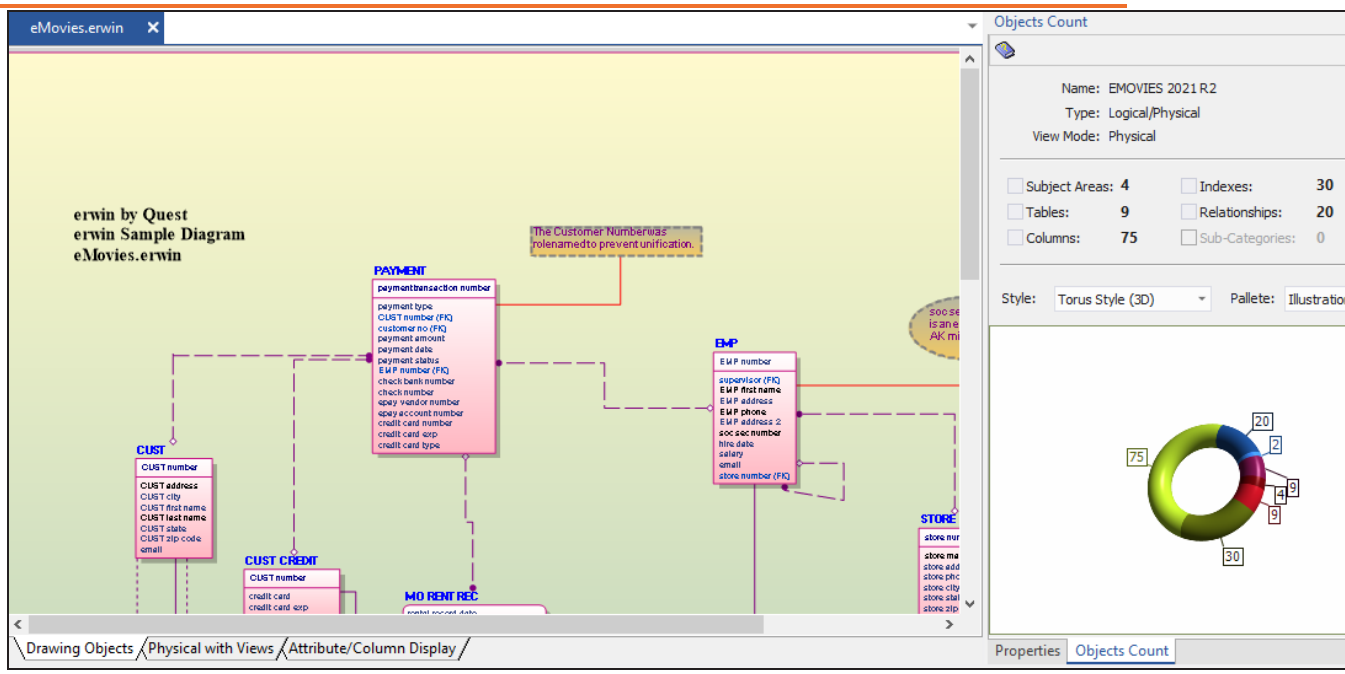
Migration by Deriving a Model

To migrate by deriving a model, follow these steps:

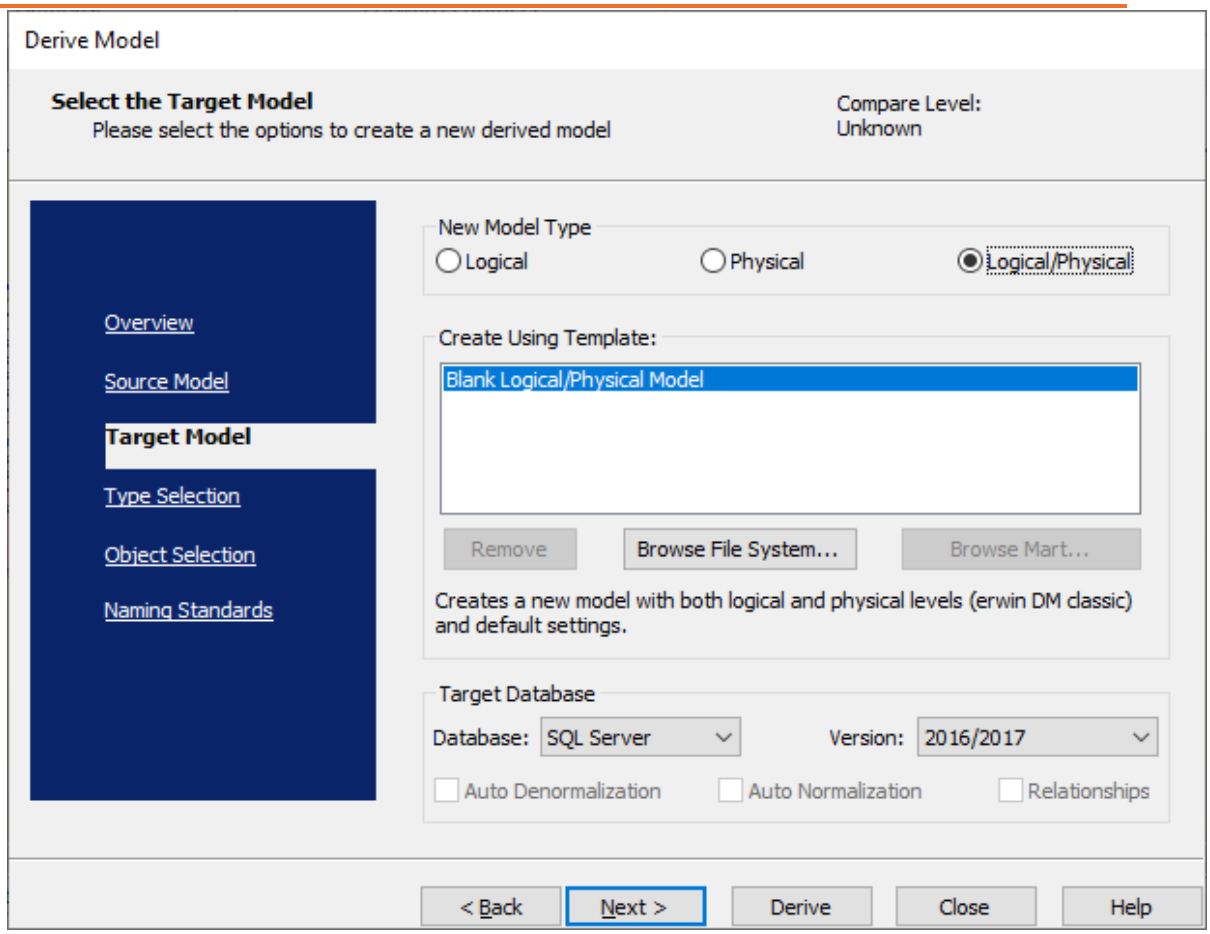
1. Open your relational model.

   > Ensure that you are in the Physical mode.

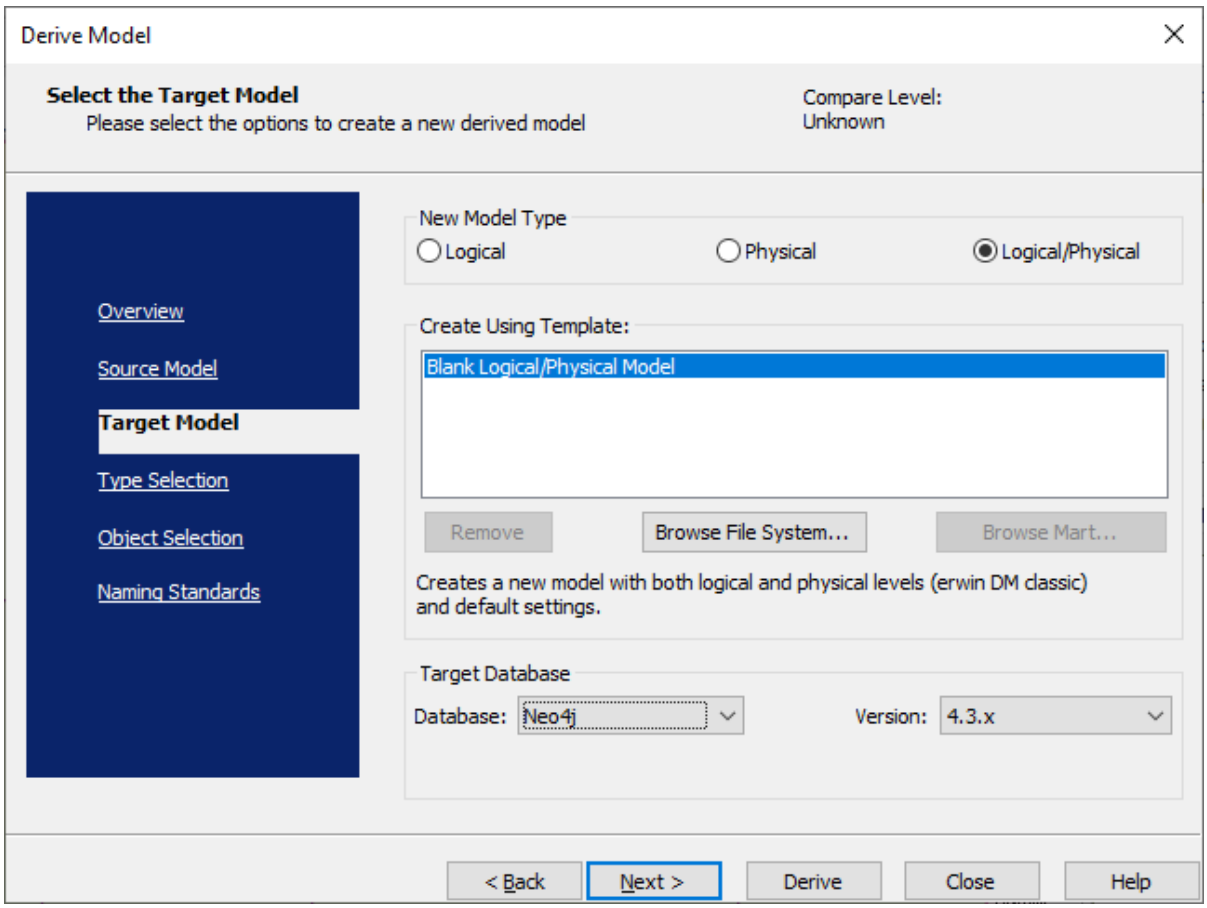
   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.



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.


Derive Model
Select the Target Model
Please select the options to create a new derived model
Compare Level:
Unknown
Overview
Source Model
Target Model
Type Selection
Object Selection
Naming Standards
New Model Type
Logical
Physical
Logical/Physical
Create Using Template:
Blank Logical/Physical Model
Remove
Browse File System...
Browse Mart...
Creates a new model with both logical and physical levels (erwin DM classic) and default settings.
Target Database
Database:
SQL Server
Version:
2016/2017
Auto Denormalization
Auto Normalization
Relationships
< Back
Next >
Derive
Close
Help

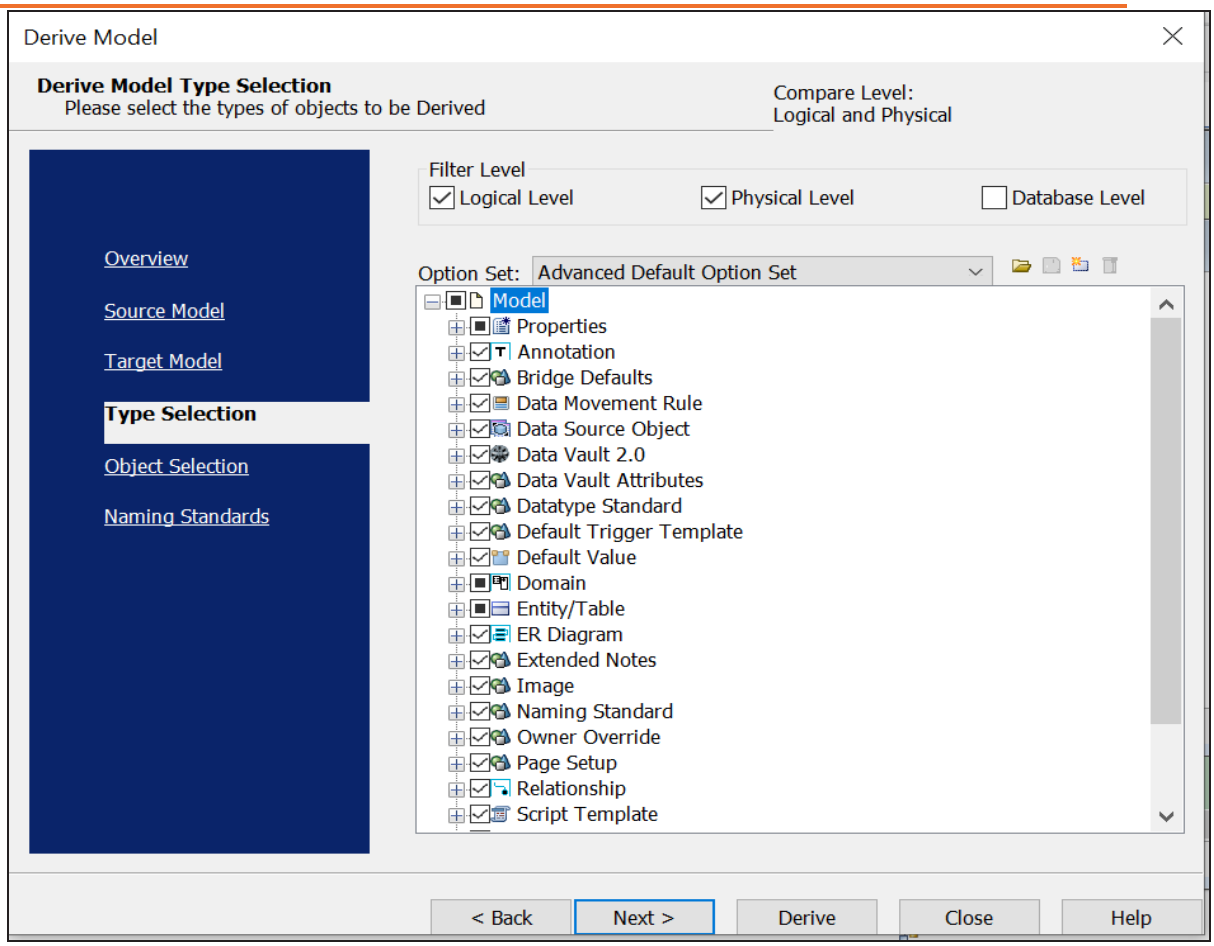
3. In the **Database** drop-down list, select **Google BigQuery**.



4. Click **Next**.

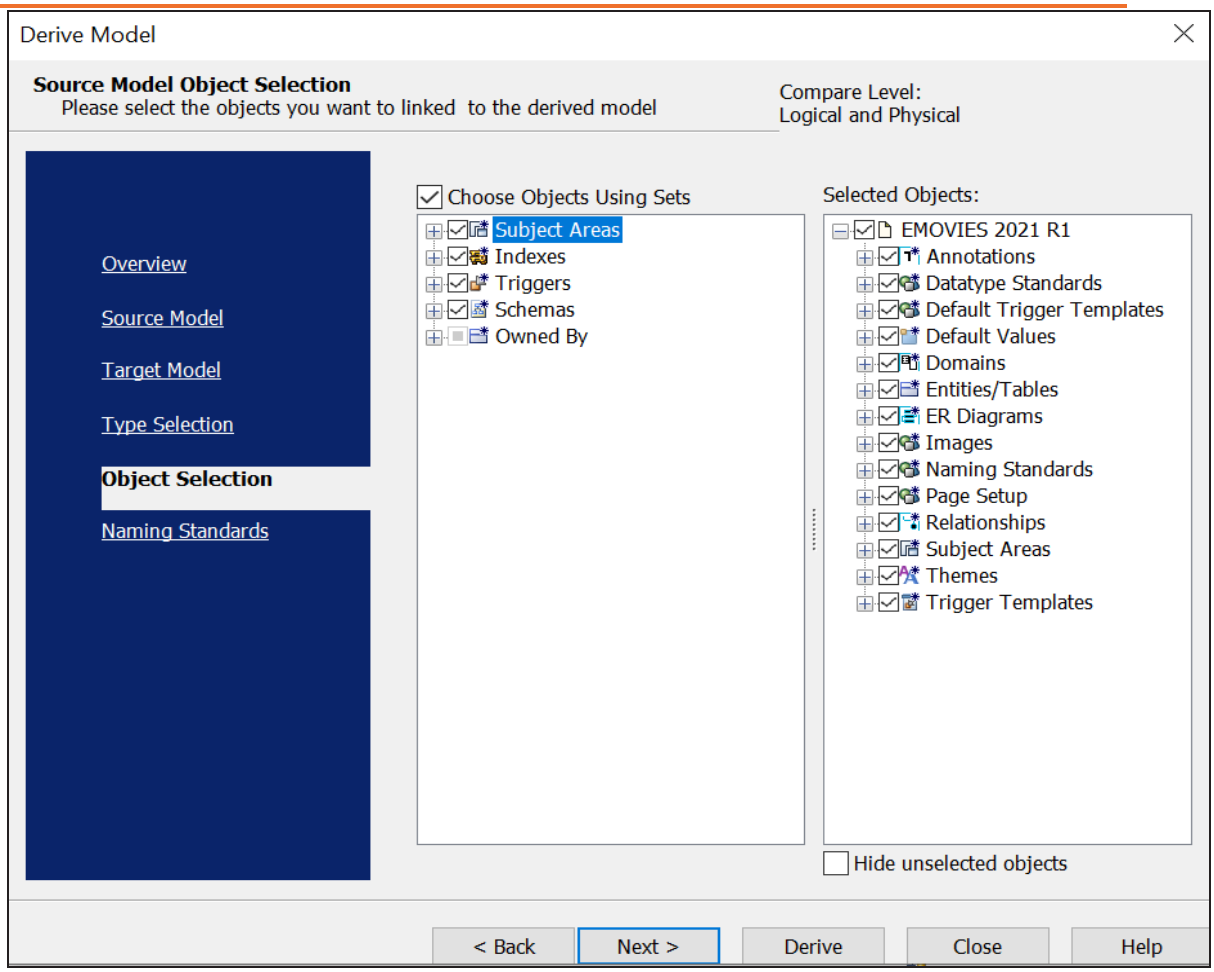
> If the Type Resolution screen appears, click **Finish**.

The Type Selection section appears.

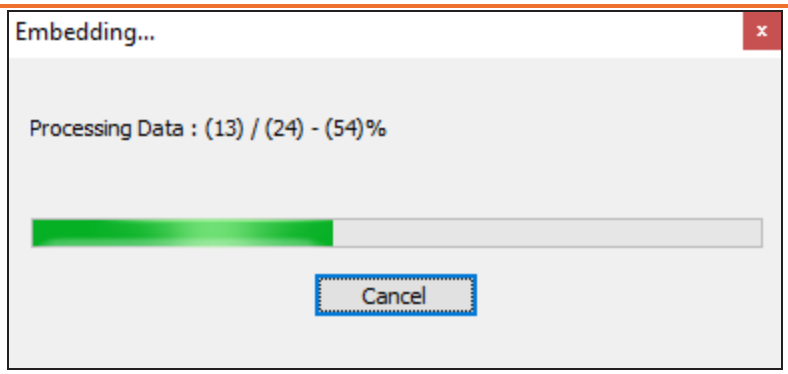


5. Select the types of objects that you want to derive into the target Google BigQuery model.

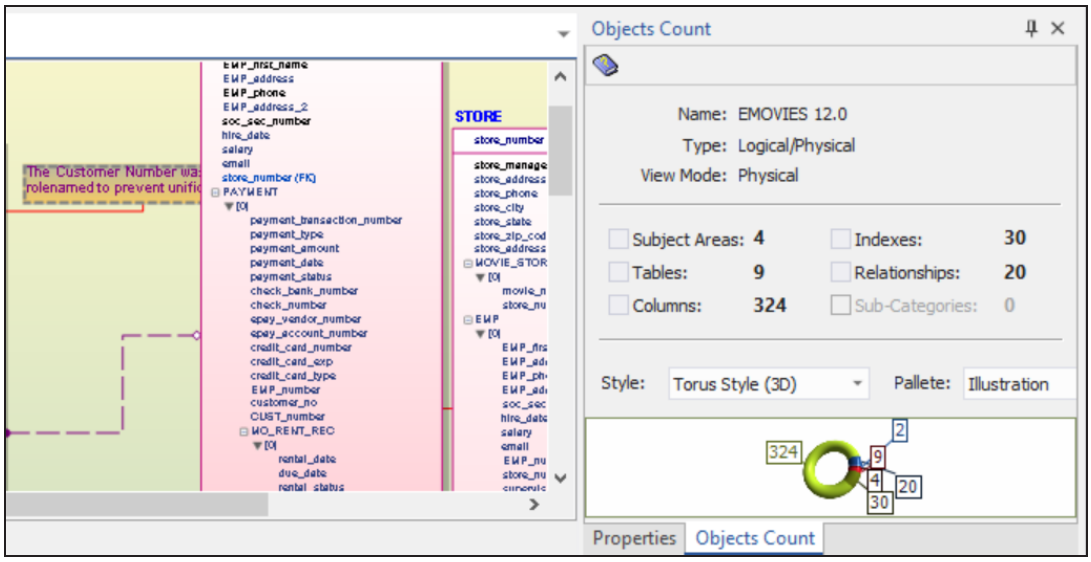
6. Click **Next**.
   The Object Selection section appears. Based on the object types you selected in step 5, it displays a list of objects.



7. Select the objects that you want to derive into the target Google BigQuery model.
8. Click **Derive**.
   The model derivation process starts.



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



In the **Objects Count** pane, note that number of columns has increased. The migration process converts tables, columns, and relationships to the NoSQL format according to the database that you select.

Databricks Support

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

- CTAS
  - CTAS Column

- Database
- Function
- Group
- Table
  - Table Column
  - Table Partition
- User Id
- View
  - View Column

The following is the list of supported data types:

- Array
- Boolean
- Double
- Integer
- Null
- Object
- String

Databricks implementation supports all erwin DM features and functions. The following sections walk you through these features:

- Reverse engineering models from database and script
- Forward engineering models to database
- Comparing changes using Complete Compare

Reverse Engineering Models

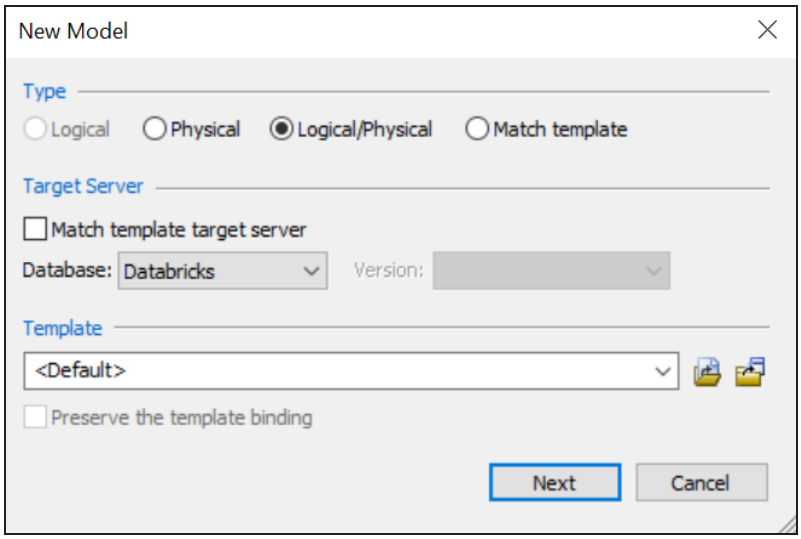
You can create a data model from a database or a script using the Reverse Engineering process. This topic walks you through the steps to reverse engineer a Databricks model. While

reverse engineering erwin Data Modeler focuses on schema generation rather than data or information.

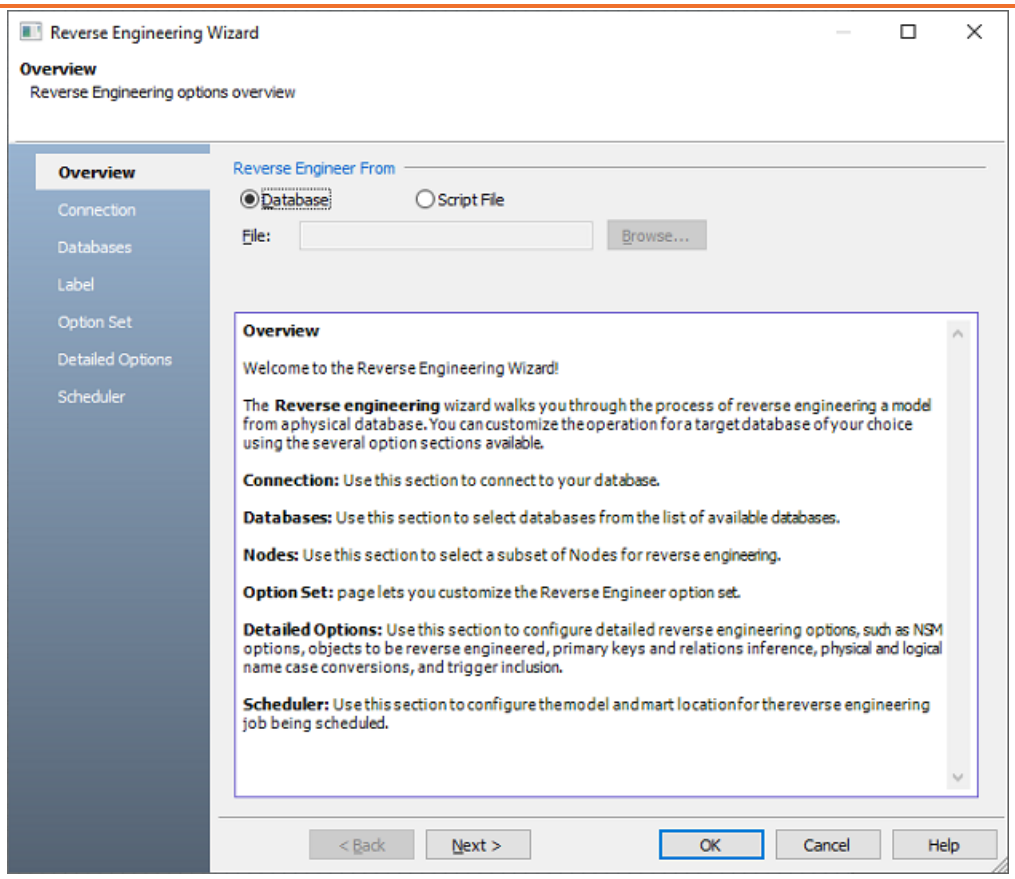
For detailed description of reverse engineering options, refer to the Reverse Engineering Options topic.

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 Databricks.



3. Click **Next**.
   The Reverse Engineering Wizard appears.



4. Click one of the following options:
    - **Database**: Use this option to reverse engineer a model from your database.

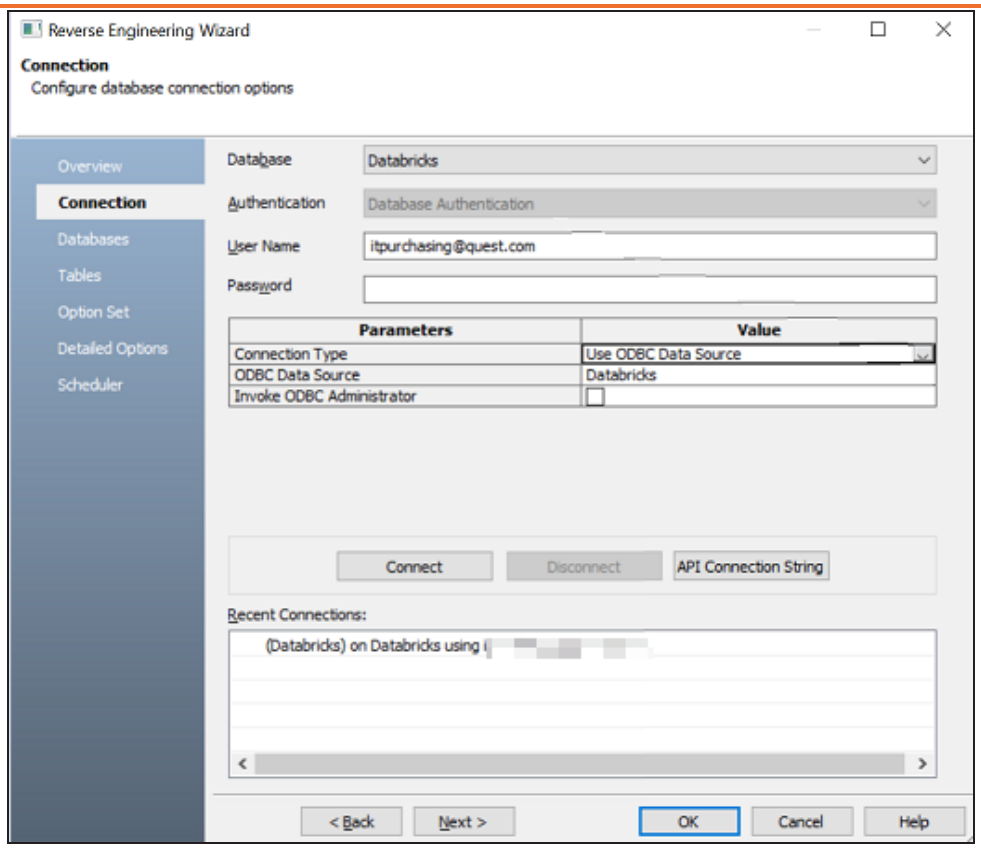
      If you click **Database**, continue to step 5.

    - **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.

      If you click **Script File**, see step 13 below.

5. Click **Next**.

   The Connection tab appears.



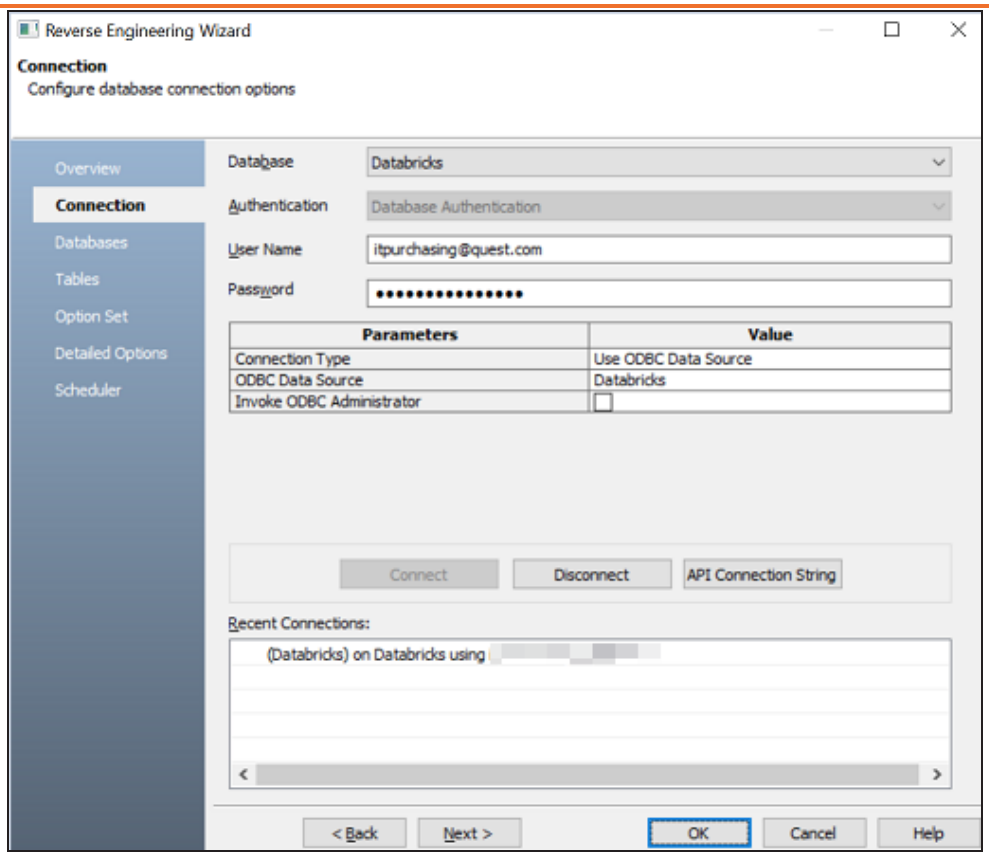
6. Enter your **User Name** and **Password**.

   The following table explains the connection parameters:

Parameter	Description	Additional Information
Connection Type	Specifies the type of connection you want to use. Select **Use ODBC Data Source** to connect using the ODBC data source that you have defined. Select **Use JDBC Connection** to connect using JDBC.	

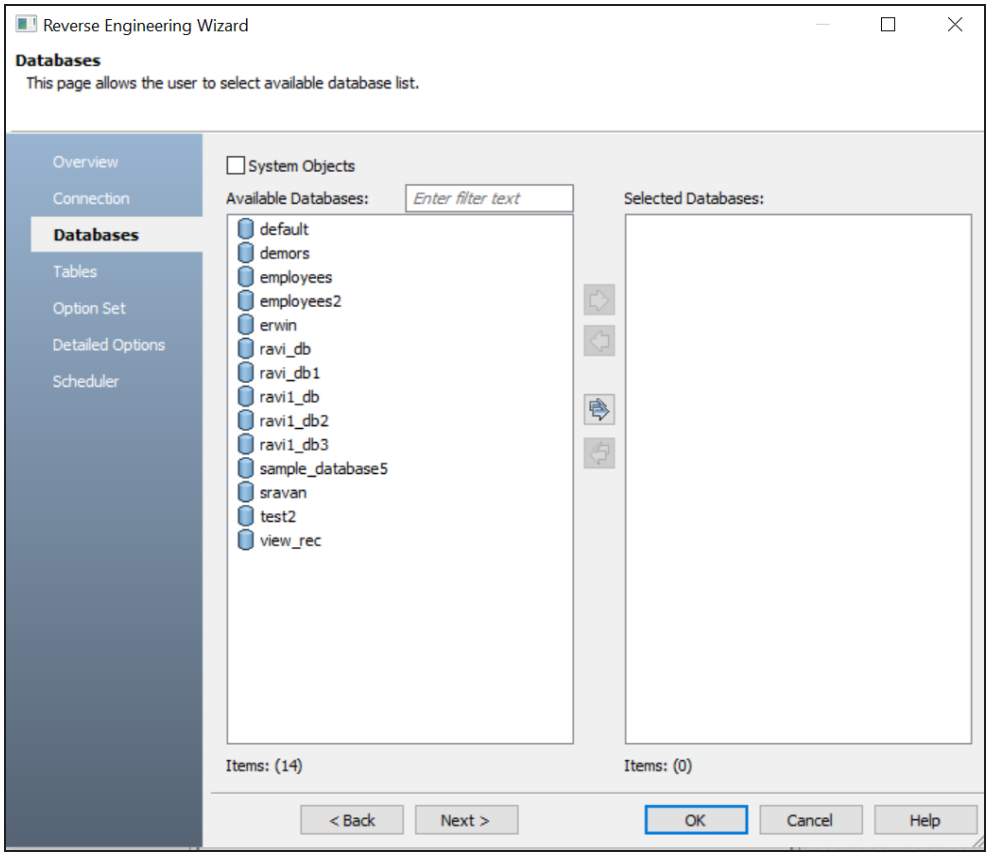
ODBC Data Source	Specifies the data source to which you want to connect. The drop-down list displays the data sources that are defined on your computer.	This option is available only when Connection Type is set to Use ODBC Data Source.
Invoke ODBC Administrator	Specifies whether you want to start the ODBC Administrator software and display the Select Data Source dialog. You can then select a previously defined data source, or create a data source.	This option is available only when Connection Type is set to Use ODBC Data Source.
Connection String	Specifies the connection string based on your JDBC instance in the following format: *jdbc:spark://<server-hostname>:443/default;transportMode=http;ssl=1;httpPath=<http-path>*	This option is available only when Connection Type is set to Use JDBC Connection. For example, jdbc:spark://dbc-64e36c82-9e5d.cloud.databricks.com:443/default;transportMode=http;ssl=1;httpPath=sql/protocolv1/o/2132616201277612/1108-064928-9gy4v7gf

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

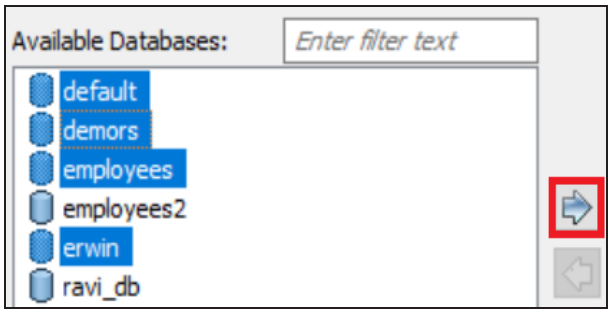

Reverse Engineering Wizard
Connection
Configure database connection options
Overview
Connection
Databases
Tables
Option Set
Detailed Options
Scheduler
Database
Databricks
Authentication
Database Authentication
User Name
itpurchasing@quest.com
Password
Parameters
Value
Connection Type
Use ODBC Data Source
ODBC Data Source
Databricks
Invoke ODBC Administrator
Connect
Disconnect
API Connection String
Recent Connections:
(Databricks) on Databricks using
< Back
Next >
OK
Cancel
Help

8. Click **Next**.

The Databases tab appears. It displays a list of available databases.

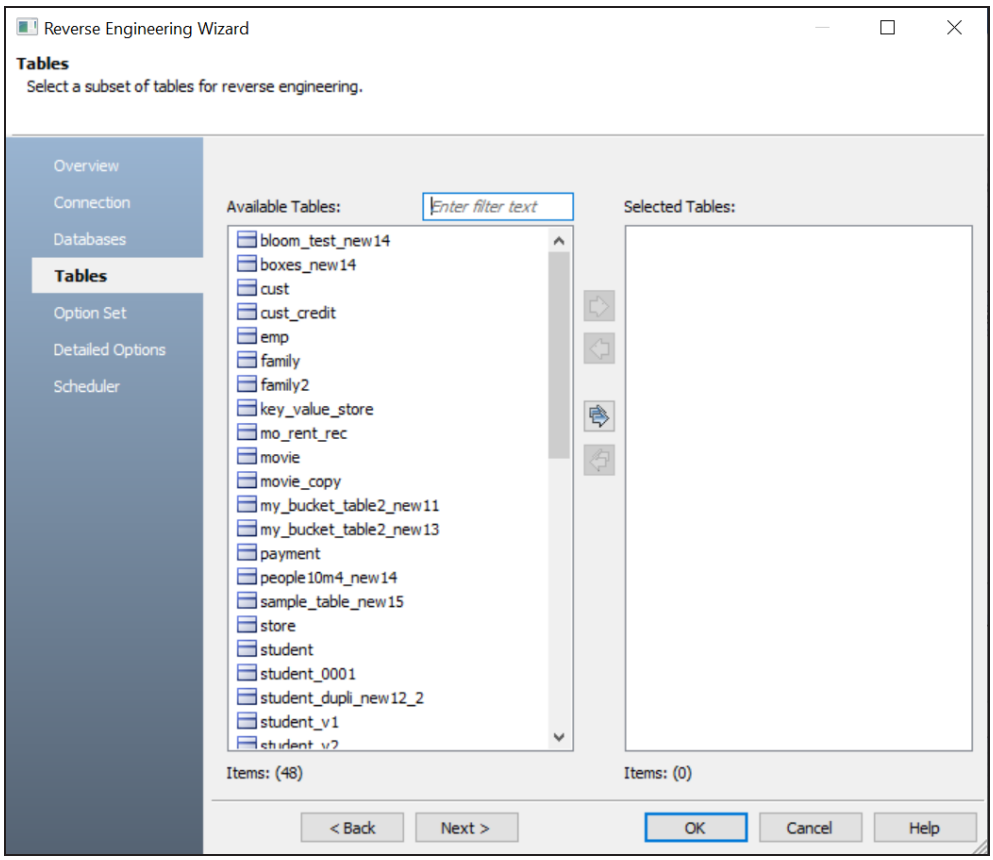


9. Under **Available Databases**, select the databases that you want to reverse engineer. Then, click  .

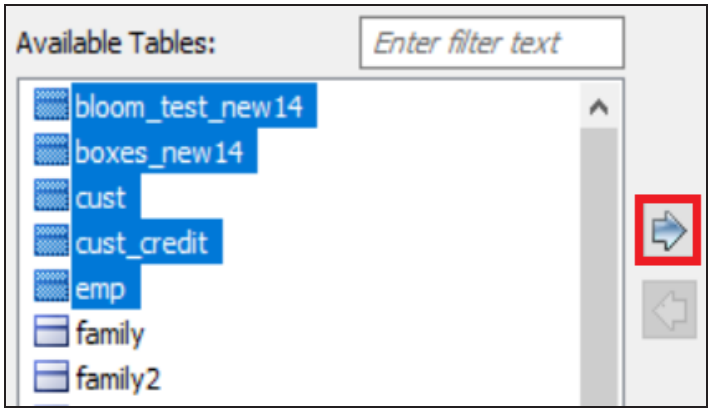


10. Click **Next**.
    The Tables tab appears. It displays a list of available tables in the databases that you

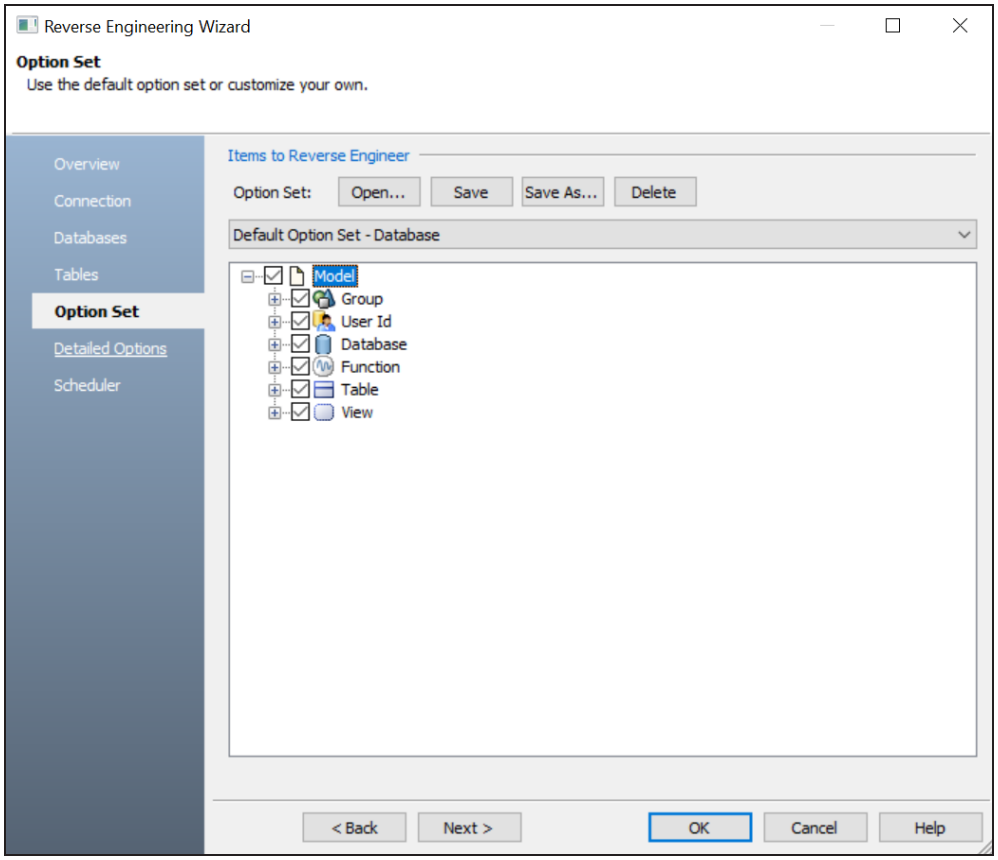
selected in step 9.



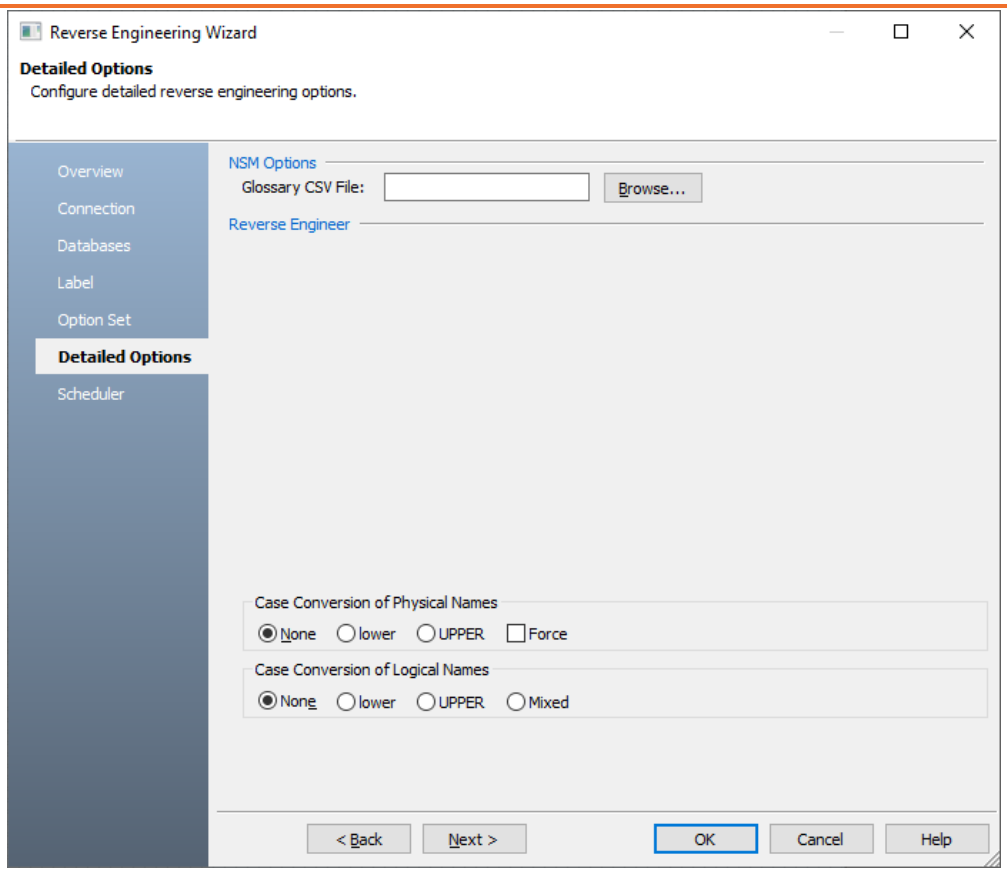
11. Under **Available tables**, select the tables that you want to reverse engineer. Then, click  .



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

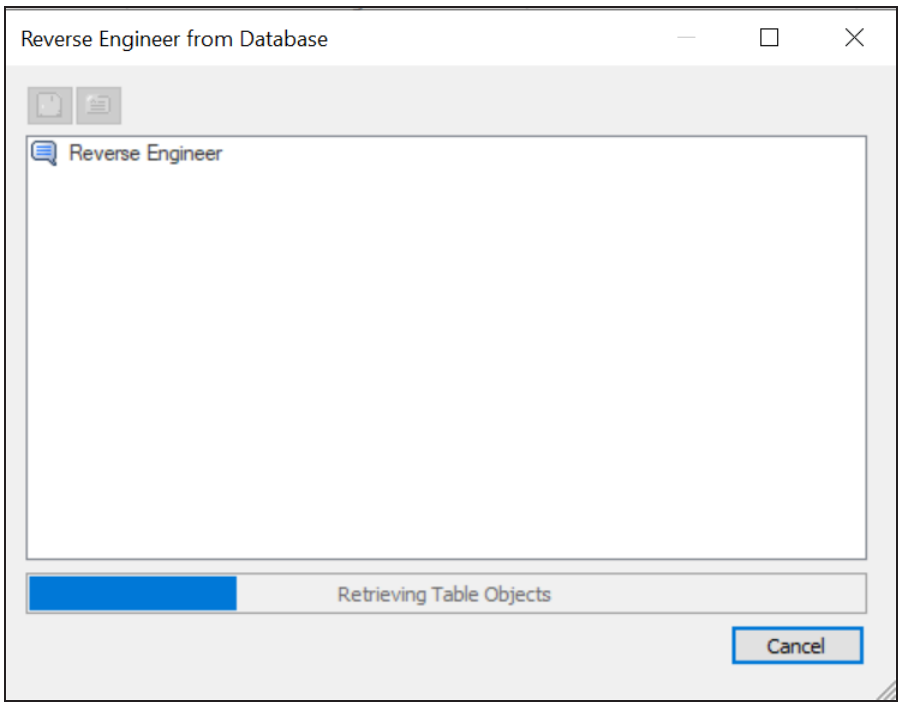


13. Click **Next**.
    The Detailed Options tab appears. Set up appropriate options based on your requirement.

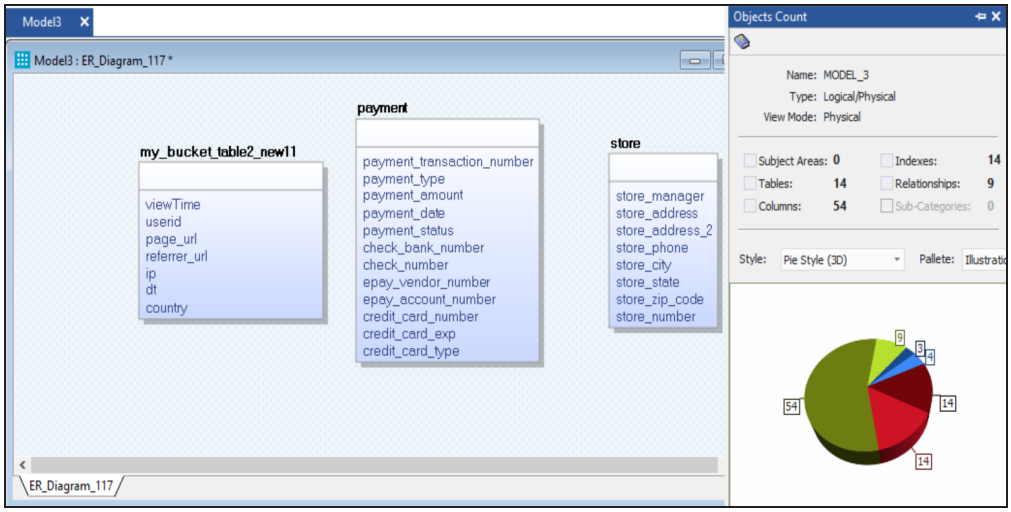

Reverse Engineering Wizard
Detailed Options
Configure detailed reverse engineering options.
Overview
Connection
Databases
Label
Option Set
Detailed Options
Scheduler
NSM Options
Glossary CSV File:
Browse...
Reverse Engineer
Case Conversion of Physical Names
None
lower
UPPER
Force
Case Conversion of Logical Names
None
lower
UPPER
Mixed
< Back
Next >
OK
Cancel
Help

14. Click **OK**.

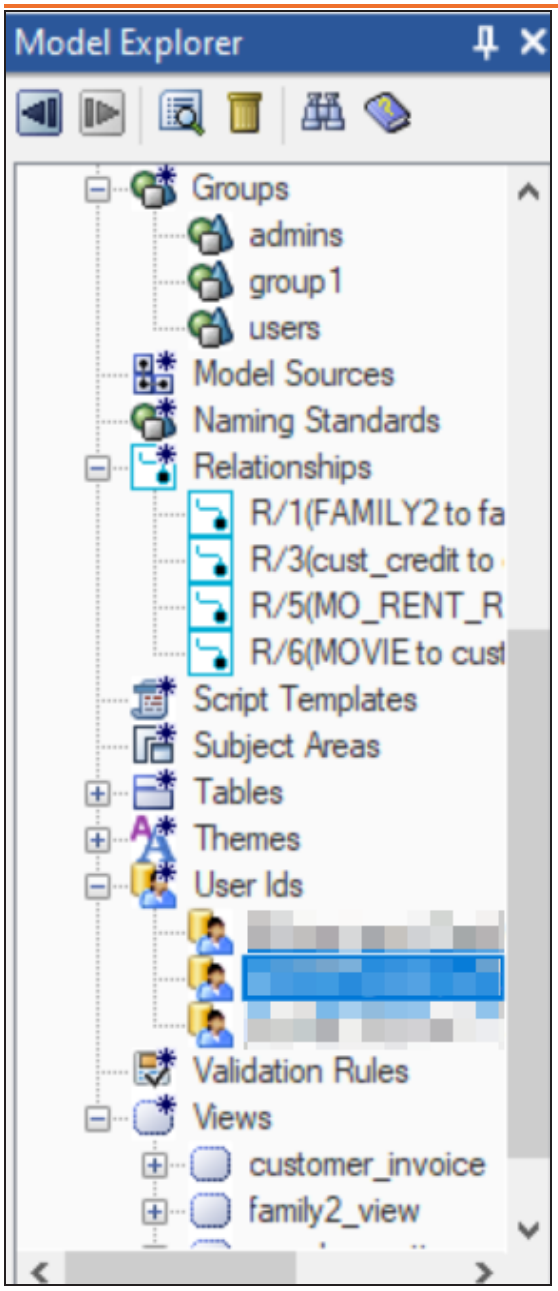
The reverse engineering process starts.



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

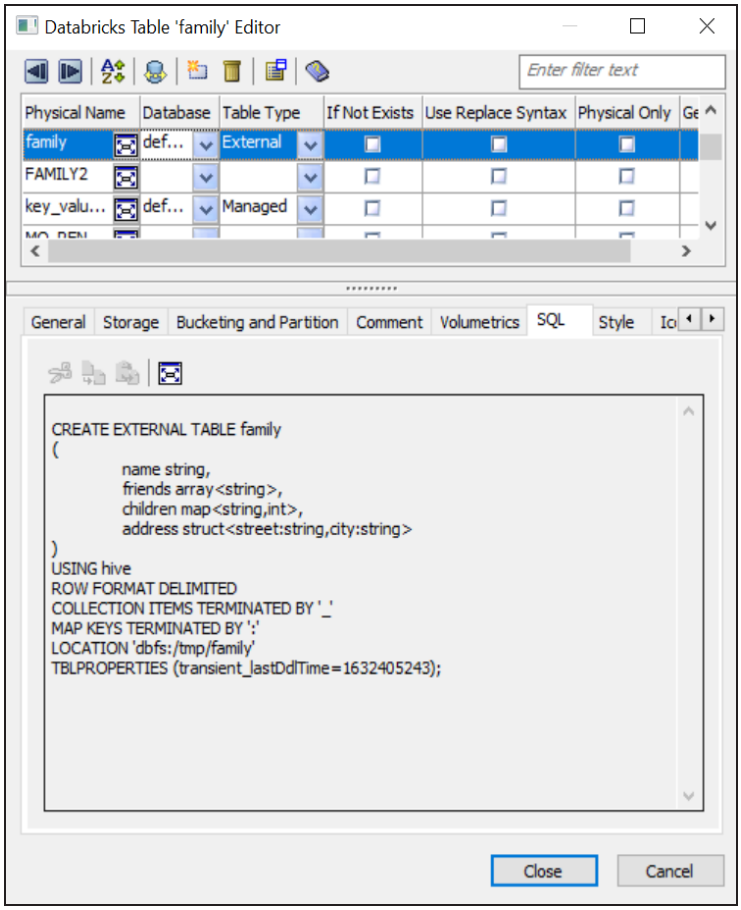


Along with Databases and Tables, other objects, such as Groups, Relationships, User IDs, and Views are retrieved.



You can view these objects via the model diagram or view their properties via the Model Explorer. Right-click an object and then, click the required Properties option. For example, on the model diagram, right click a table and then, click **Table Properties**. The Databricks

Table Editor appears. You can view the table's CREATE statement on the SQL tab. As seen, the table, family has four columns, address, children, friends, and name.



Reverse Engineering Options for Databricks

Following are the reverse engineering options for Databricks.

Overview

Parameter	Description	Additional Information
Reverse Engineer From	Specifies whether you want to reverse engineer from a script or database	**Database**: Indicates that the model is reverse engineered from database **Script File**: Indicates that the model is reverse engineered from a script

File	Specifies the script file location	This option is available when Script File is selected.

While reverse engineering from script, for the LIKE TABLE syntax, ensure that you use the USING clause.

Connection

Parameter	Description	Additional Information
Connection Type	Specifies the type of connection you want to use. Select **Use ODBC Data Source** to connect using the ODBC data source that you have defined. Select **Use JDBC Connection** to connect using JDBC.	
ODBC Data Source	Specifies the data source to which you want to connect. The drop-down list displays the data sources that are defined on your computer.	This option is available only when Connection Type is set to Use ODBC Data Source.
Invoke ODBC Administrator	Specifies whether you want to start the ODBC Administrator software and display the Select Data Source dialog. You can then select a previously defined data source, or create a data source.	This option is available only when Connection Type is set to Use ODBC Data Source.
Connection	Specifies the connection string based on your JDBC instance in the following	This option is available only when Connection Type is set to Use JDBC Connection.

String	format: *jdbc:spark://<server-host-name>:443/de-fault;trans-portMode-e=http;ssl=1;httpPath=<http-path>*	For example, jdbc:spark://dbc-64e36c82-9e5d.cloud.dat-abrick-s.com:443/de-fault;trans-portMode-=http;ss-l=1;h-ttpPath=sql/protocolv1/o/2132616201277612/1108-064928-9gy4v7gf

Databases

Parameter	Description	Additional Information
Available Data-bases	Specifies a list of available databases	
Selected Data-bases	Specifies a list of selected databases for reverse engin-eering	
System Objects	Specifies whether system databases are included under the Available Databases	

Tables

Parameter	Description	Additional Inform-ation
Available Tables	Specifies a list of available tables	
Selected Tables	Specifies a list of selected tables for reverse engin-eering	

Option Sets

Parameter	Description	Additional Information
Option Set	Specifies the option set template for	**Open**: Use this option to open a saved

	reverse engineering	XML option set file. **Save**: Use this option to save the configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at some external location. **Delete**: Use this option to delete an option set.
<Option Set Name>	Specifies the objects to be reverse engineered according to the selected option set. You can edit this list.	

Detailed Options

Parameter	Description	Additional Information
NSM Options	Specifies the naming standard glossary file in the .CSV format	
Case Conversion of Physical Names	Specifies how the case conversion of physical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Force**: Indicates whether the physical name property of all the logical/physical models is overridden. If this option is enabled, the logical/physical link is broken between the logical and physical name. If this option is not enabled, all logical and physical names are set to the same value after the process completes.
Case Conversion of Logical	Specifies how the case con-	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case

Names	version of logical names is handled	**UPPER**: Indicates that the names are converted to upper case **Mixed**: Indicates that the mixed-case logical names are preserved

Scheduler

Parameter	Description	Additional Information
Model	Specifies the location and name of the reverse engineered model	For example: C:\Scheduler\<Model Name>.erwin When you schedule a job on a remote server, ensure the model path is same for remote and local server.
Mart Folder	Specifies the location or library in your mart where the reverse engineered model is saved	To use this option, ensure that you are connected to a mart. For more information, refer to the Connecting to Mart topic.
Complete Compare	Specifies whether the Complete Compare (CC) process should run while reverse engineering	
Output File	Specifies the location of the CC output file generated	
File	Specifies that the target model location is on the local system	
Mart	Specifies that the target model location is in the mart	
Using Latest Version	Specifies whether the target model is the latest version of the model in the mart	This option is available only when Mart is selected.
Save To Mart	Specifies whether the reverse engineered model is saved to the mart	This option is available only when Using Latest Version is selected.

Target Model	Specifies the location of the target model for CC	
Option Set	Specifies the option set that is used for CC	**Advanced Default Option Set**: Indicates that all erwin DM metadata is included. CC works slowest with this option. **Speed Option Set**: Indicates that only the essential metadata is included. CC works the fastest with this option set. **Standard Default Option Set**: Indicates that standard metadata is included. CC works fast with this option set compared to the Advanced option set.

Forward Engineering Models

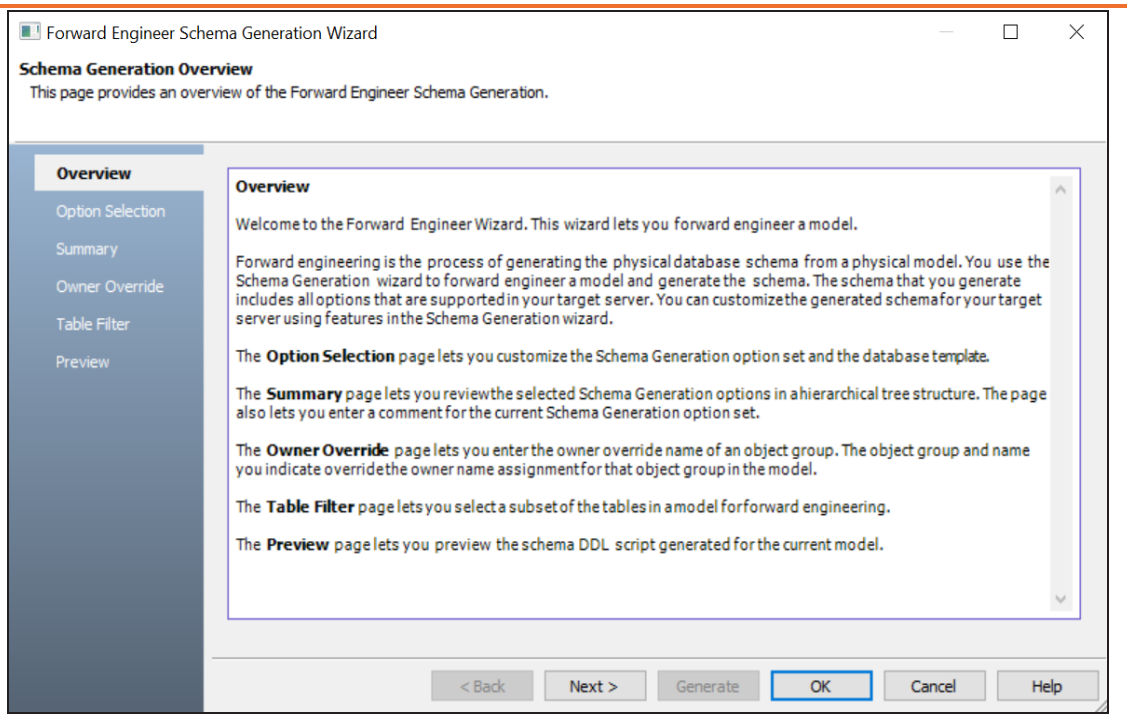
You can generate a physical database schema from a physical model using the Forward Engineering process. This topic walks you through the steps to forward engineer a Databricks model. For detailed description of forward engineering options, refer to the Forward Engineering Options topic.

To forward engineer a model:

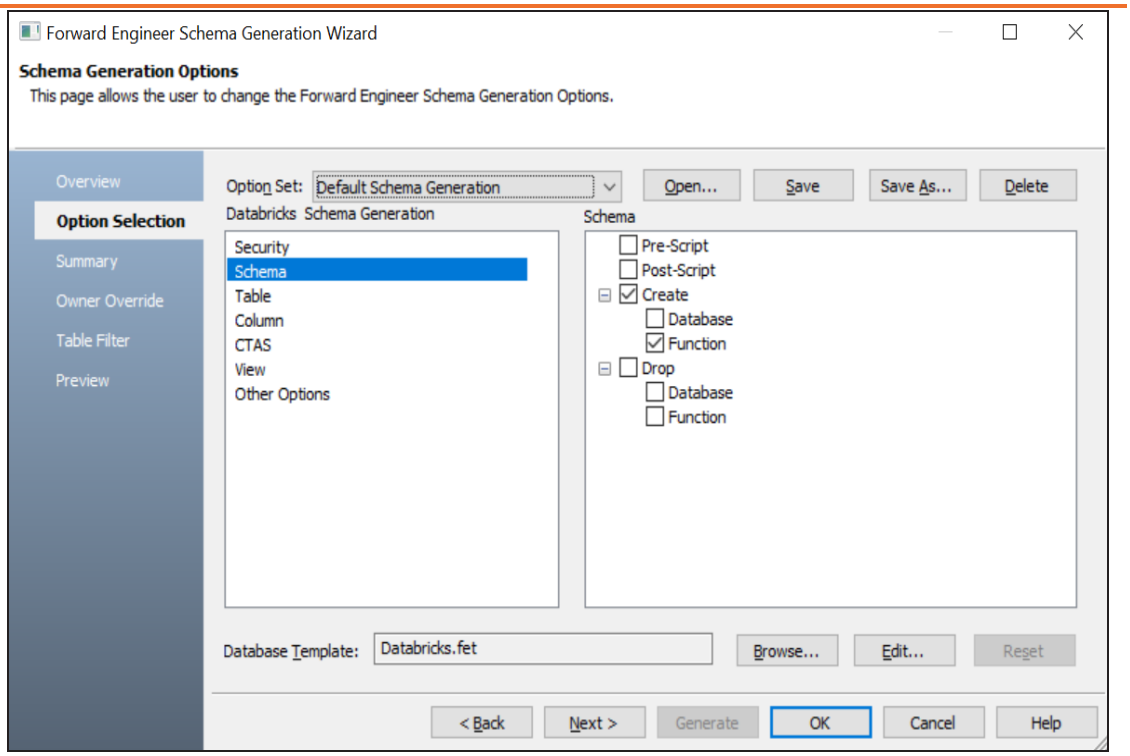
1. Open your Databricks model.

   Ensure that you are in the Physical mode.

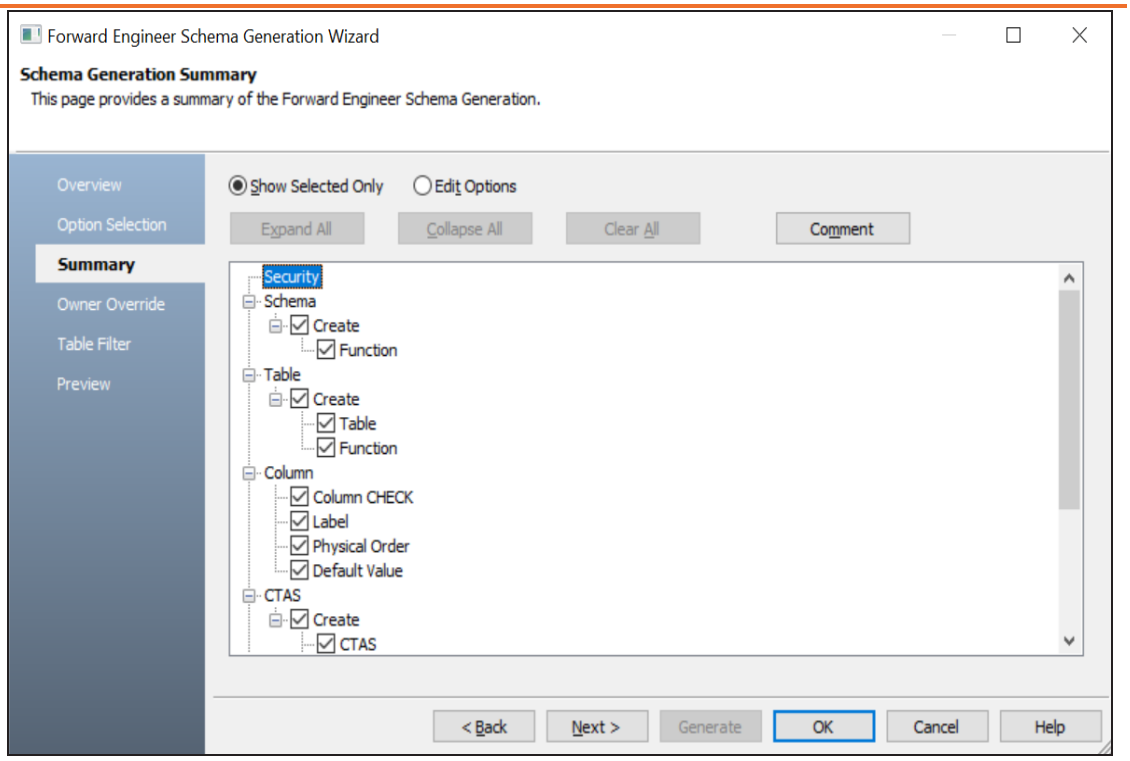
2. Click **Actions** > **Schema**.
   The Forward Engineer Schema Generation Wizard appears.



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

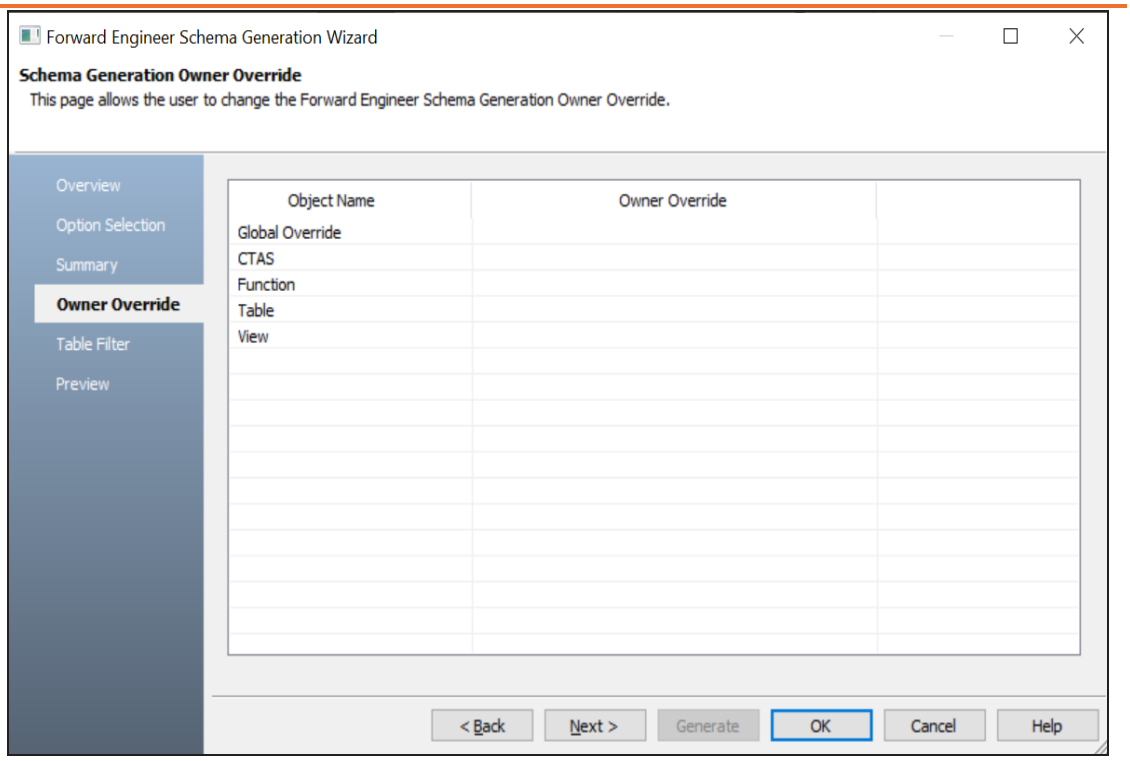


4. Click **Next**.
   The Summary tab appears. It displays a list of selected options for the schema generation. Use Edit Options to update selected options.



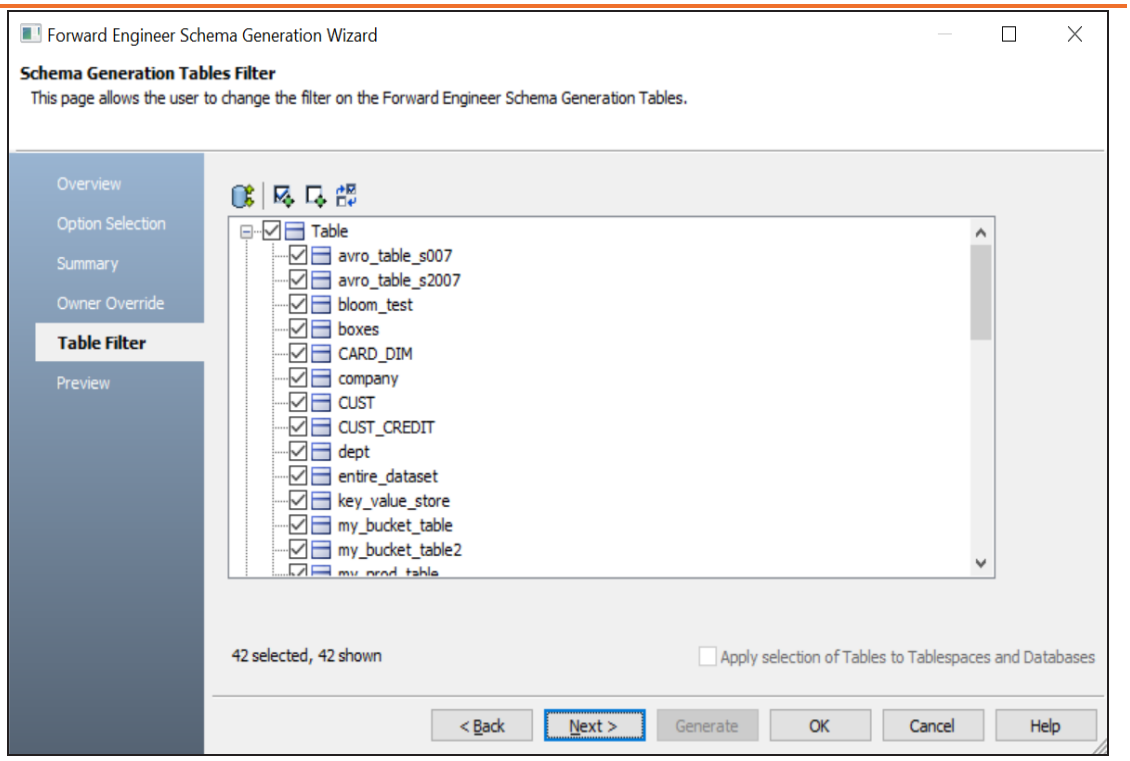
5. Click **Next**.

   The Owner Override tab appears. It displays a list of objects.

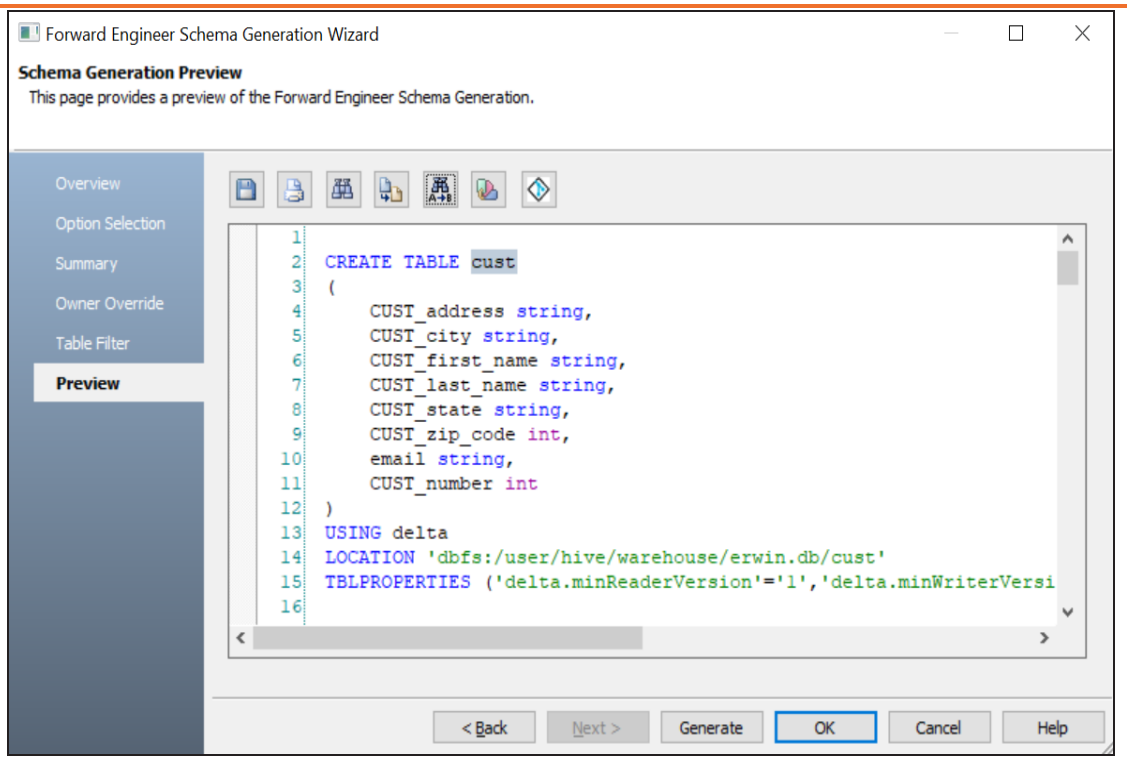


6. Enter database names as owner of the objects.
7. Click **Next**.

   The Table Filter tab appears. It displays a list of tables available in your model.



8. Click **Preview** to view the schema and its script.

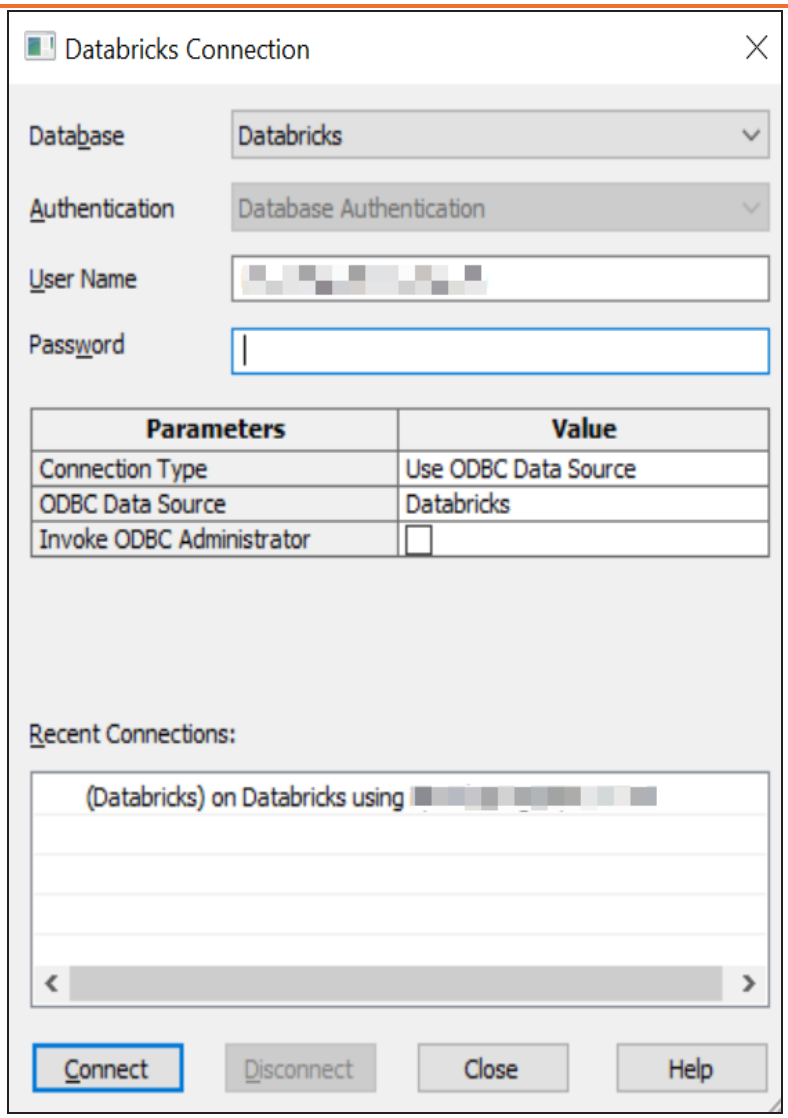


Use the following options:

- **Save** ( ): Use this option to save the generated script.
- **Print** ( ): Use this option to print the generated schema.
- **Search** ( ): Use this option to search through the generated schema.
- **Copy** ( ): Use this option to copy the selected text in the schema.
- **Replace** ( ) : Use this option to find and replace text in the generated schema.
- **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.

9. Click **Generate**.

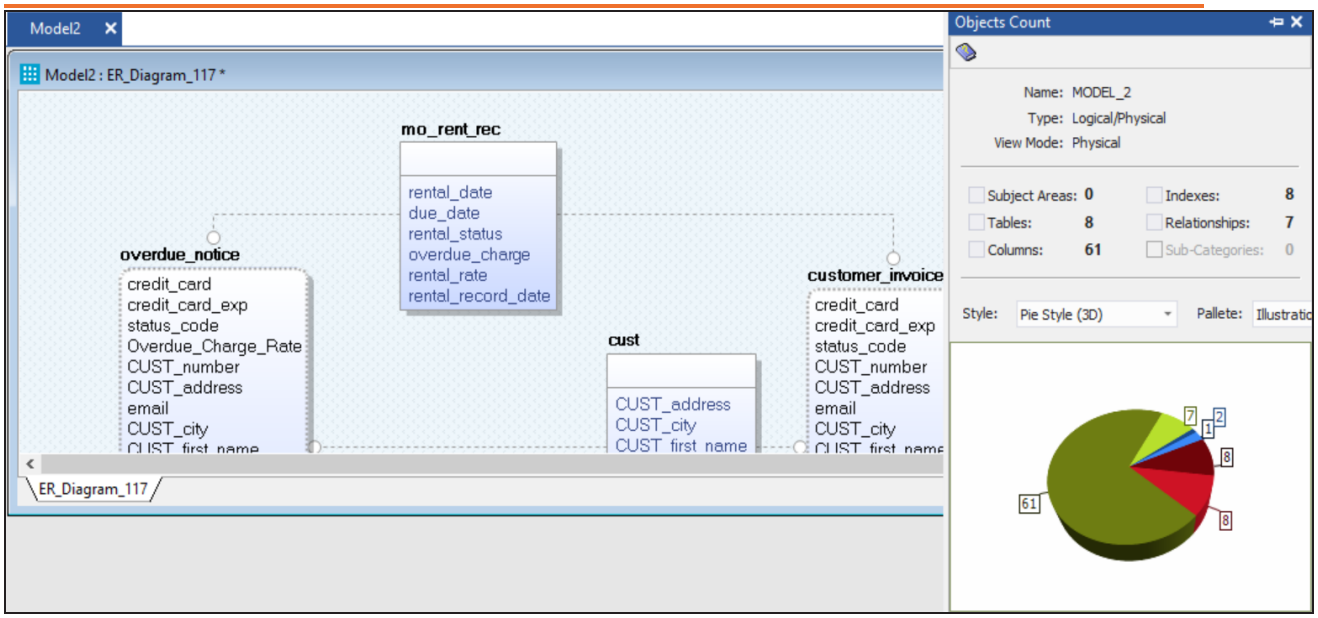
   The Databricks Connection screen appears.



10. Enter User Name, Password, and appropriate connection parameters to connect the required database. Then, click **Connect**. For more information on connection parameters, refer to the connection parameters topic.

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

For example, the following model has one database, eight tables with 61 columns and seven relationships. Apart from this, it has eight indexes.

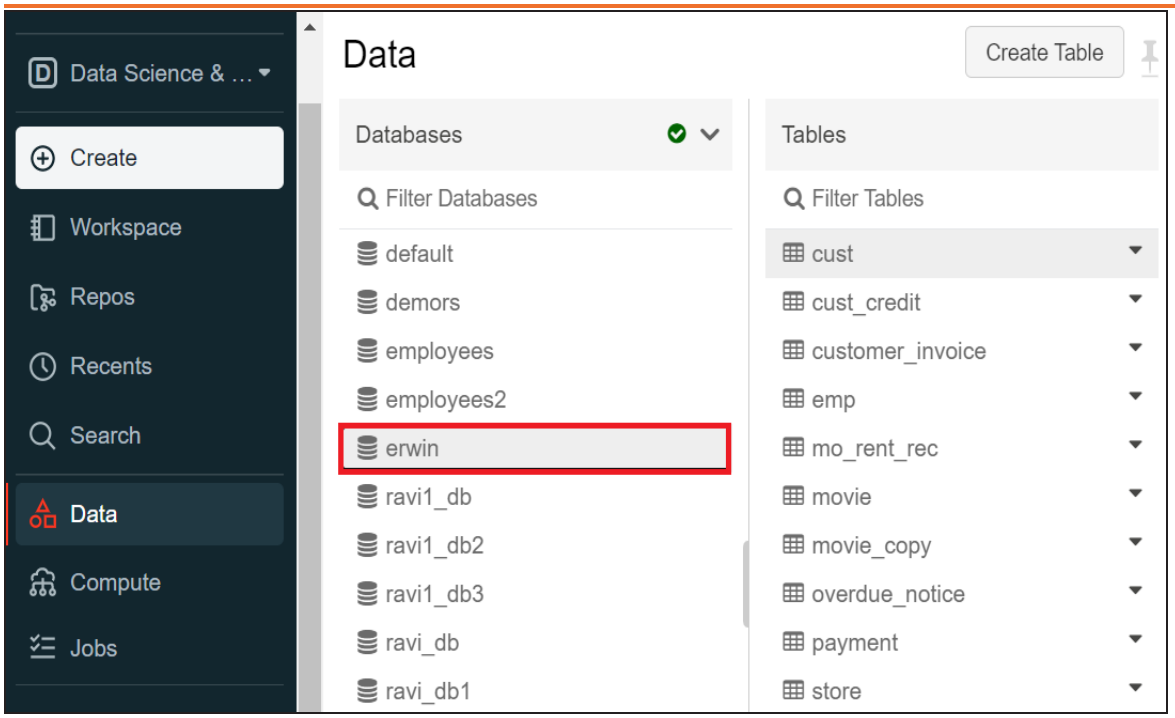


On forward engineering, the following script was generated:

```
CREATE DATABASE erwin
LOCATION 'dbfs:/user/hive/warehouse/erwin.db'
WITH DBPROPERTIES ();

CREATE TABLE cust
(
    CUST_address string,
    CUST_city string,
    CUST_first_name string,
    CUST_last_name string,
    CUST_state string,
    CUST_zip_code int,
    email string,
    CUST_number int
)
```

Based on the generated schema, the erwin database has eight tables with 61 columns.



Forward Engineering Options for Databricks

Following are the forward engineering options for Databricks.

Option Selection

Parameter	Description	Additional Information
Option Set	Specifies the option set template for forward engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save a configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at some external location. **Delete**: Use this option to delete an option set.
Schema	Specifies the schema options for the schema generation	**Pre-Script**: Indicates whether the pre-scripts attached to the schema are executed **Post-Script**: Indicates whether the post-scripts attached to the schema are executed

		Create: Indicates whether the Create syntax for the selected objects are executed. Selecting an object under the Create node executes the create syntax for that object. **Drop**: Indicates whether the Create syntax for the selected objects are executed. Selecting an object under the Drop node executes the drop syntax for that object.
Table	Specifies the table options for the schema generation	**Pre-Script**: Indicates whether the pre-scripts attached to tables are executed **Post-Script**: Indicates whether the post-scripts attached to tables are executed **Create**: Indicates whether the Create syntax for the selected objects are executed. Selecting an object under the Create node executes the create syntax for that object. **Drop**: Indicates whether the Create syntax for the selected objects are executed. Selecting an object under the Create node executes the create syntax for that object.
Column	Specifies the column options for the schema generation	**Column CHECK**: Indicates whether validation rules attached to columns is included in the schema **Label**: Indicates whether labels attached to columns are included in the schema **Physical Order**: Indicates whether physical order attached to columns are included in the schema **Default Value**: Indicates whether default values of the columns are included in the schema
CTAS	Specifies the CTAS options for the schema generation	**Pre-Script**: Indicates whether the pre-scripts attached to CTAS are executed **Post-Script**: Indicates whether the post-scripts attached to CTAS are executed **Create**: Indicates whether the Create syntax for the selected objects are executed. Selecting an object under the

		Create node executes the create syntax for that object. **Drop**: Indicates whether the Create syntax for the selected objects are executed. Selecting an object under the Create node executes the create syntax for that object.
View	Specifies the View options for the schema generation	**Pre-Script**: Indicates whether the pre-scripts attached to Views are executed **Post-Script**: Indicates whether the post-scripts attached to Views are executed **Create**: Indicates whether the Create syntax for the selected objects are executed. Selecting an object under the Create node executes the create syntax for that object. **Drop**: Indicates whether the Create syntax for the selected objects are executed. Selecting an object under the Create node executes the create syntax for that object.
Other Options	Specifies the other options for the schema generation	Selecting an object includes it in the schema

Summary

Parameter	Description	Additional Information
Summary	Specifies the summary of the options selected for the schema generation	**Show Selected Only**: Use this option to show the selected options only. **Edit Options**: Use this option to edit the selected options for the schema generation.
Comment	Use this option to add comments about the schema.	

Owner Override

Parameter	Description	Additional Inform-

		ation
Object Name	Specifies the object names	
Owner Override	Specifies database as the owner of the object	

Table Filter

Parameter	Description	Additional Information
Table	Specifies the selected tables for the schema generation	
Display either Logical Names or Physical Names		**Logical Names**: Indicates that only logical names of the tables are included in the generated schema **Physical Names**: Indicates that only physical names of the tables are included in the generated schema **Physical Names, show owner**: Indicates that physical names and owners of the tables are included in the generated schema **Physical Names, show owner using User**: Indicates that the physical names and owners of the tables are included in the generated schema. Owners of the tables are displayed using User.
Select all of the items in the list	Use this option to select all the tables in the list.	
Unselect all of the items in the list	Use this option to unselect all the tables.	
Select all unselected items, and unselect all selected	Use this option to select all the unselected tables and unselect all the previously selected tables.	

items		

Preview

Description	Additional Information
Displays the schema in the text editor	**Save**: Use this option to save the generated schema. **Search**: Use this option to search through the generated schema. **Print**: Use this option to print the generated schema. **Replace**: Use this option to find and replace text in the generated schema. **Copy**: Use this option to copy the selected text in the schema. **Text Options**: Use this option to edit window settings, fonts, and syntax color. **Git**: Use this option to commit the FE script to a Git repository.

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.

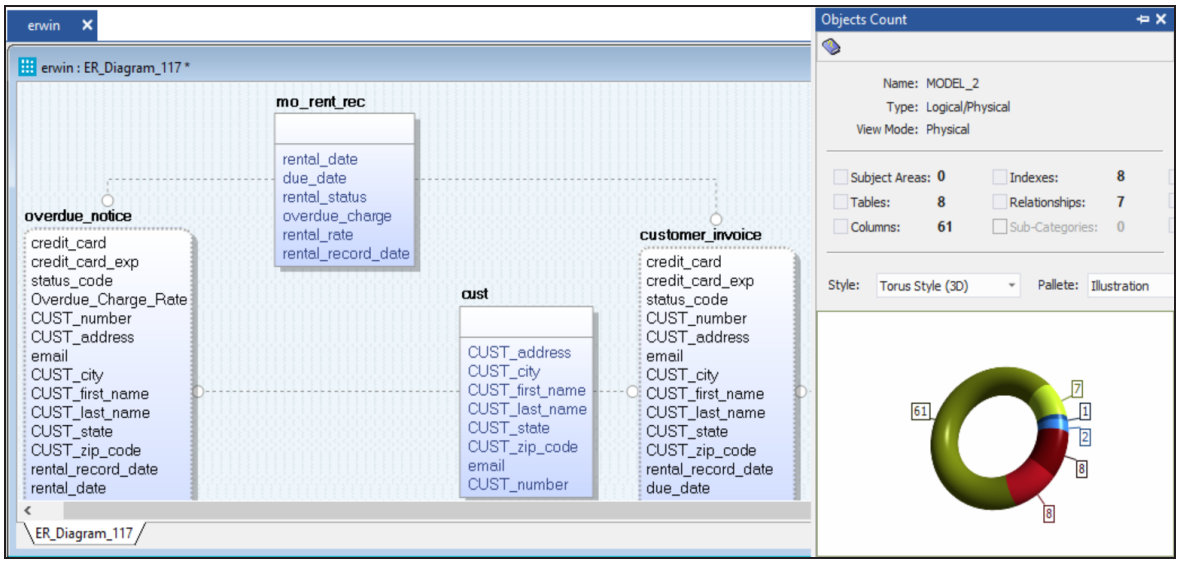
This topic walks you through the steps to compare a Databricks model with database.

To compare models with database:

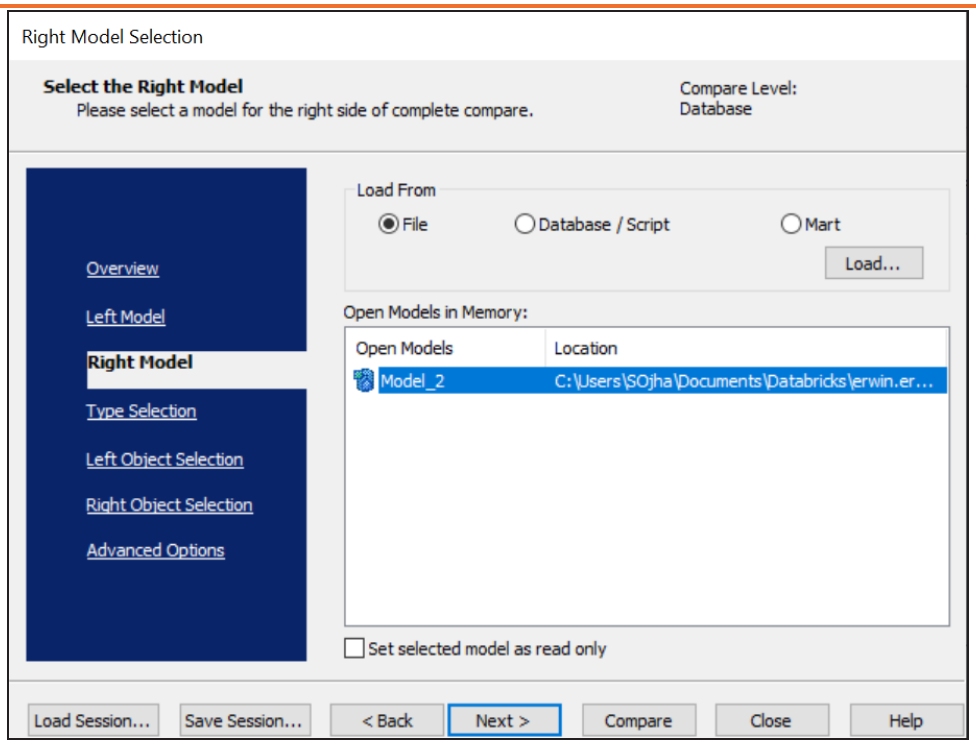
1. Open your Databricks model.


Ensure that you are in the Physical mode.

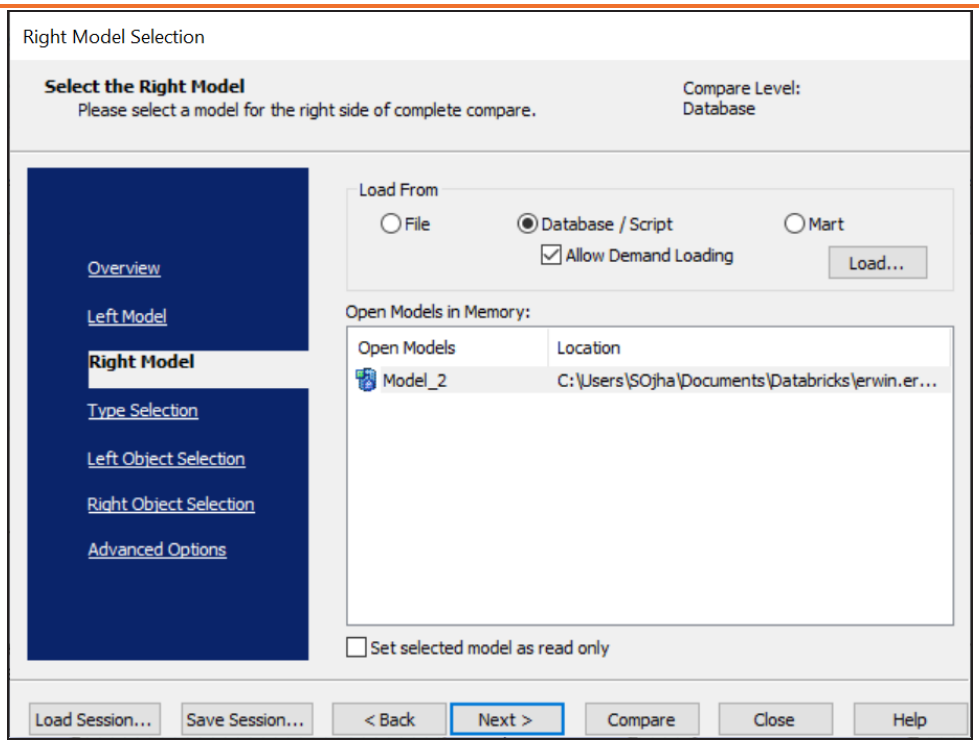
For example, the following image uses a Databricks model with eight tables, 61 columns, and seven relationships.



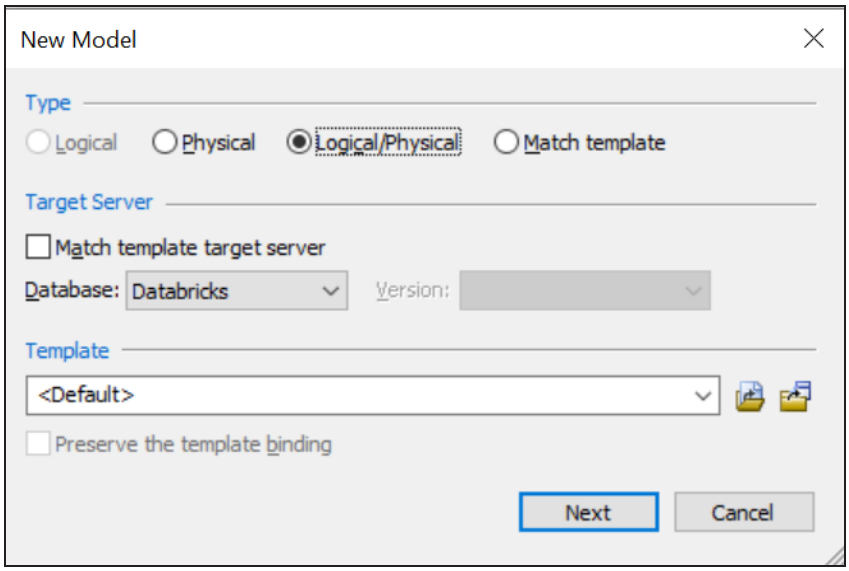
2. Click **Actions** > **Complete Compare**.
   By default, the Complete Compare wizard assigns the open model as the Left Model. Hence, the Right Model tab appears.



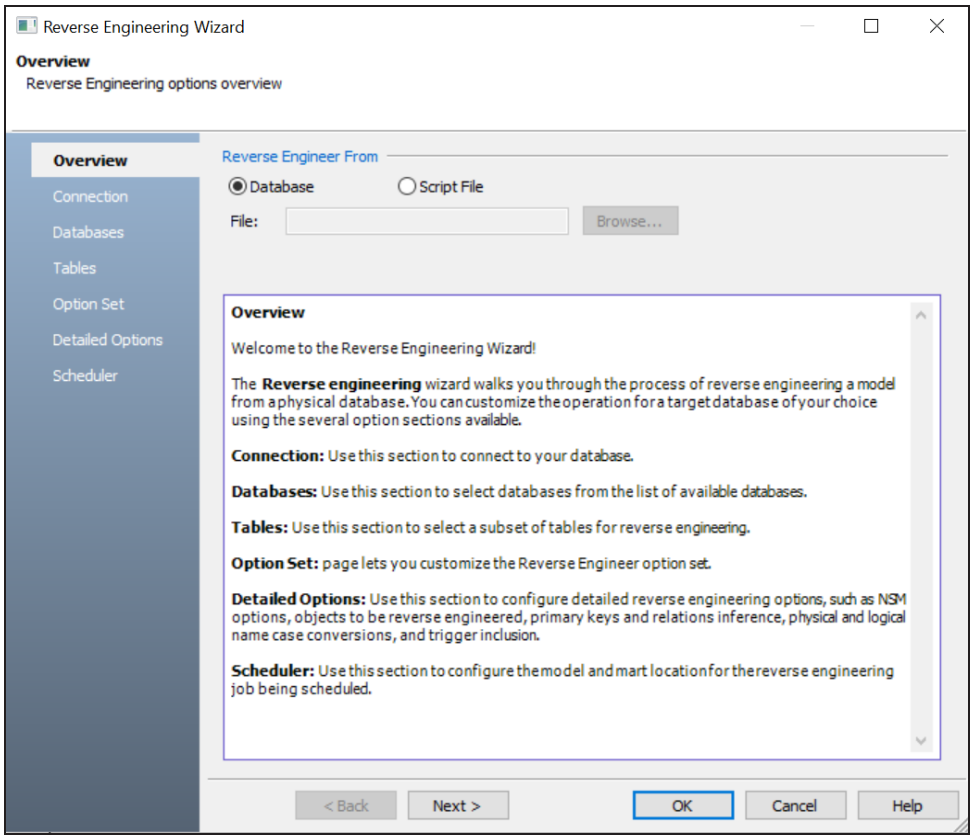
3. Click **Database/Script**.
   By default, the Allow Demand Loading option is selected.



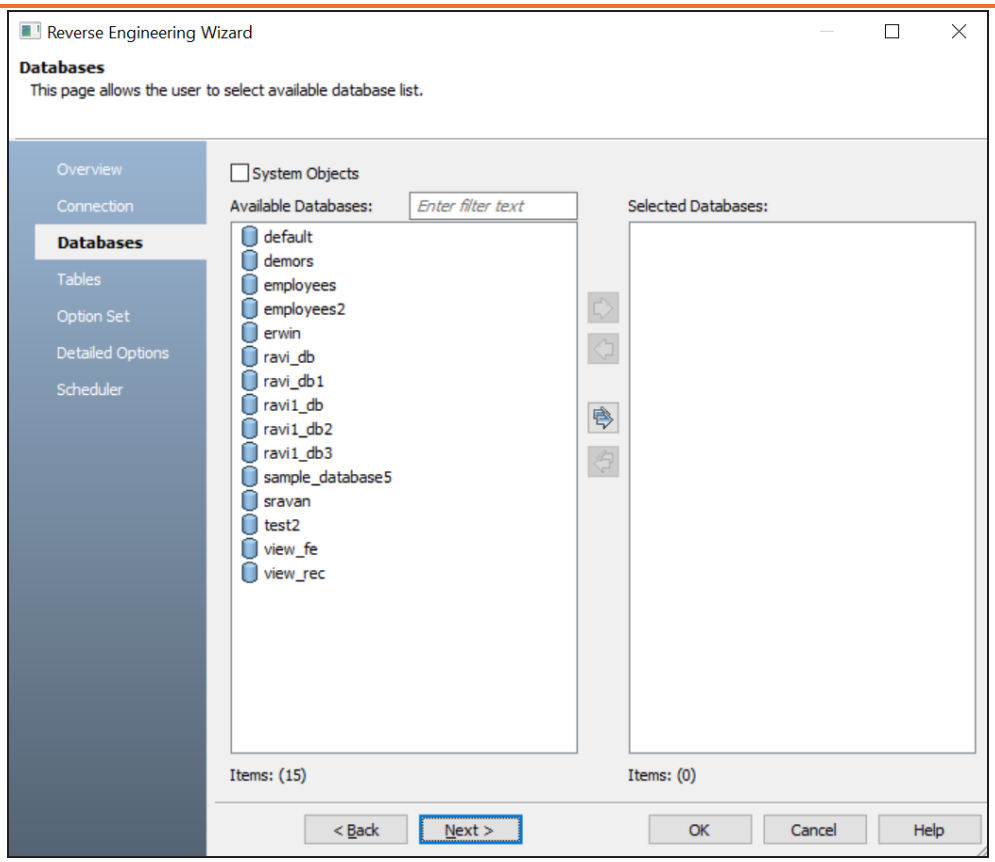
4. Click **Load**.
   The New Model dialog box appears. This starts the reverse engineering process to pull a model from the database to compare.



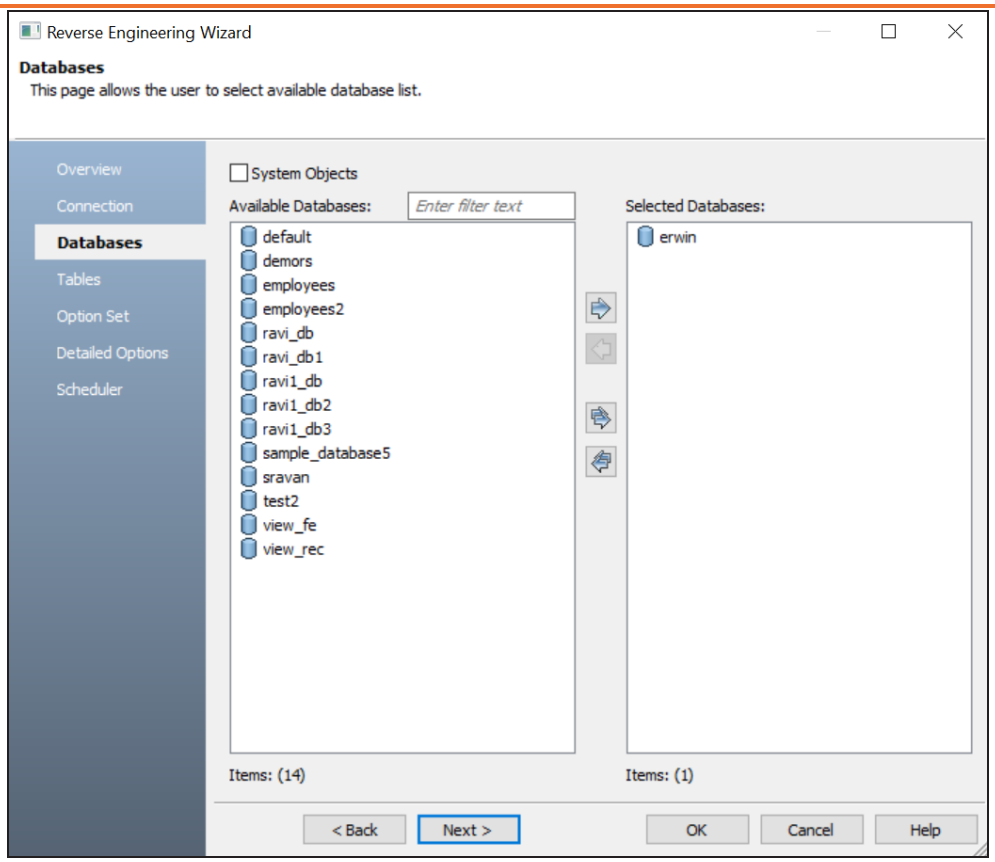
5. Ensure that the Database is set to Databricks. Then, click **Next**.
   The Reverse Engineer Wizard appears.



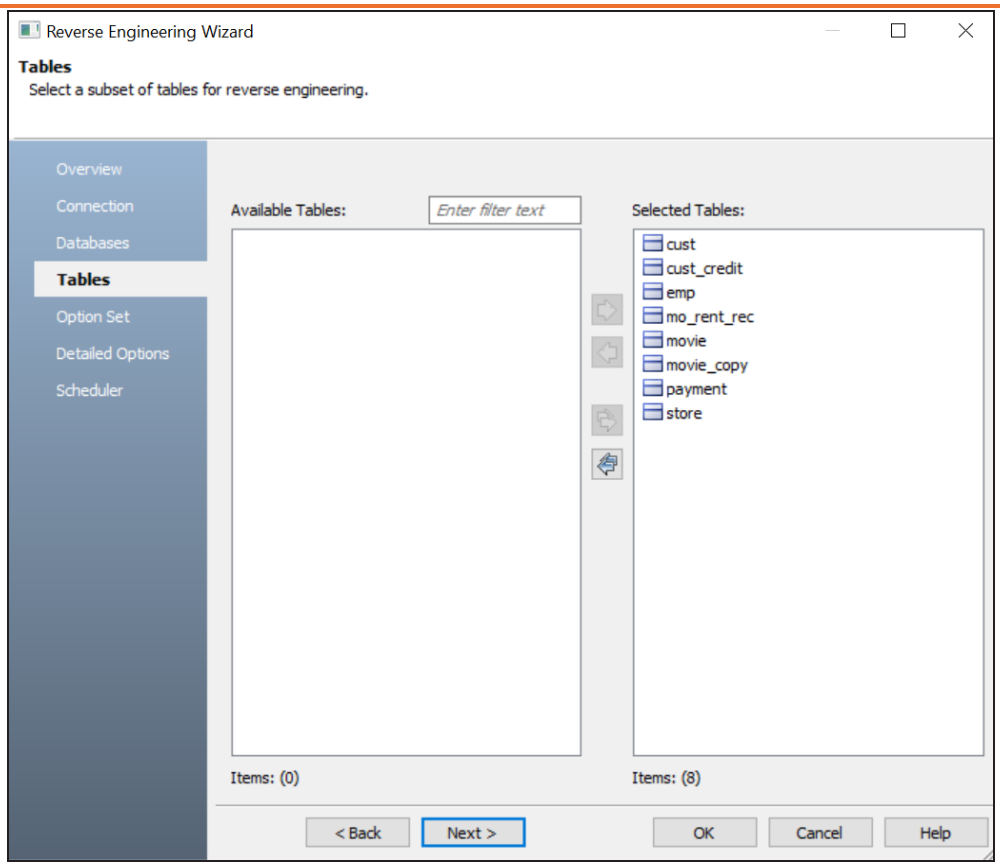
6. Click **Database**. Then, click **Next**.
   The Connection tab appears. Use this tab to connect to the database from which you want to reverse engineer the model.
7. After connection is established, click **Next**.
   The Databases tab appears. It displays a list of available databases.



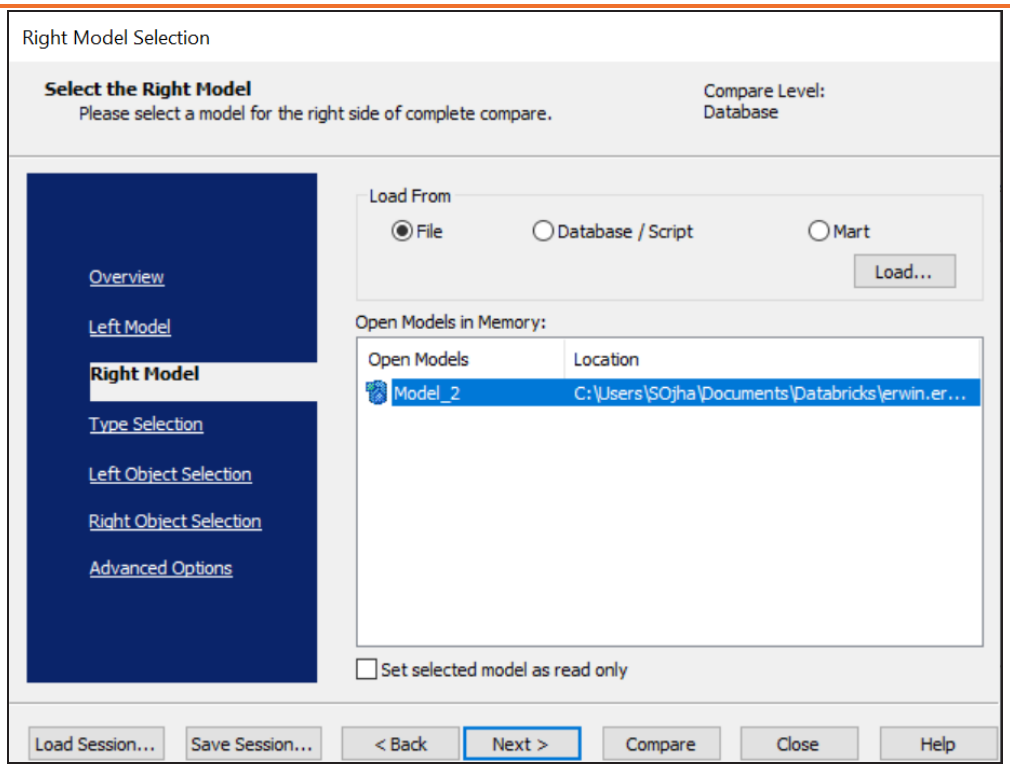
8. Under **Available Databases**, select the databases that you want to reverse engineer. Then, click  .
   This moves the selected databases under Selected Databases.



9. Click **Next** and on the Tables tab, click  .
   This selects all the available tables.

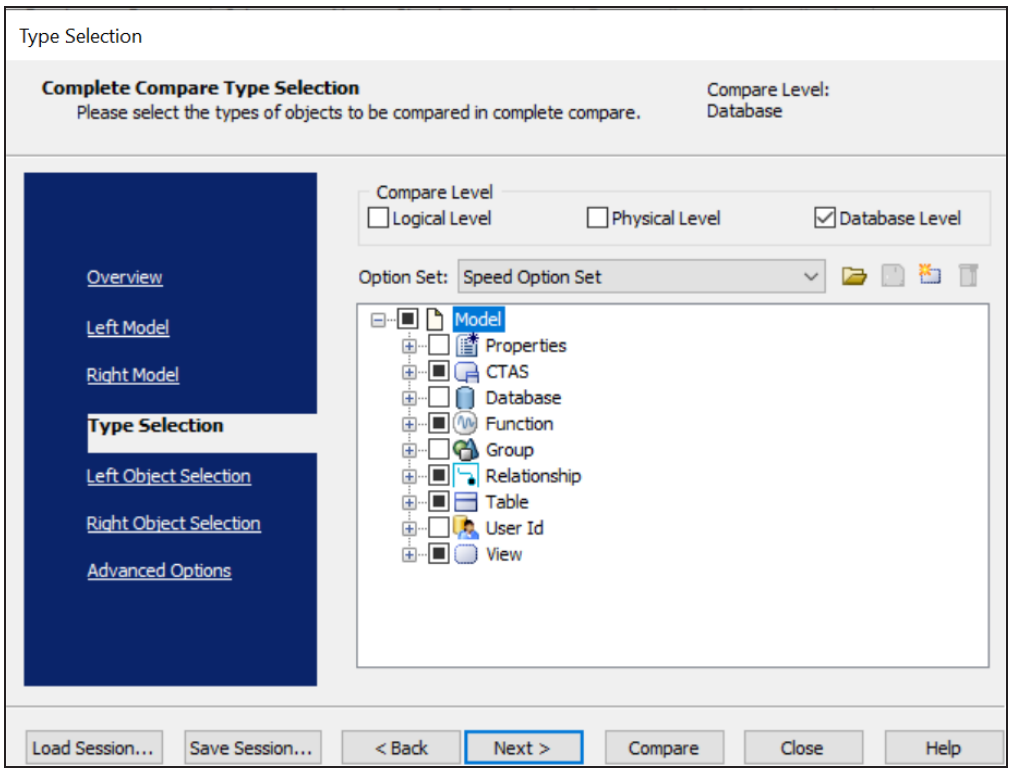


10. Click **Next** and on the Option Set tab, keep the default configuration.
11. Click **Next** and on the Detailed Options tab, keep the default configuration.
12. Click **OK**.
    The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse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



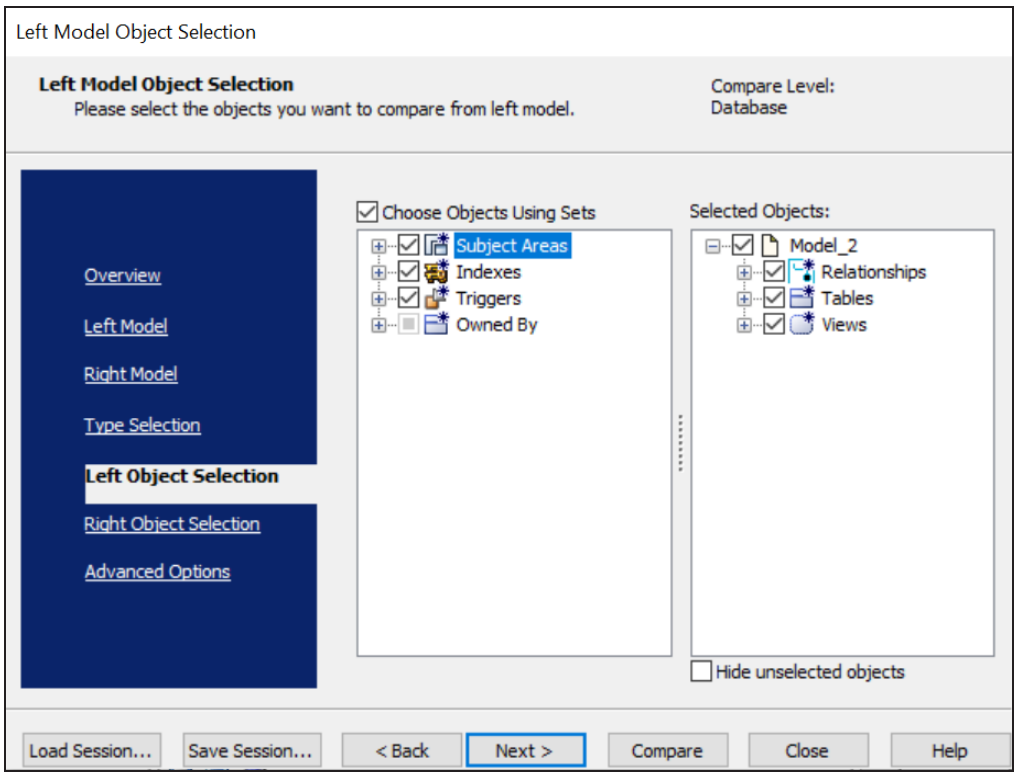
13. Click **Next** and on the Type Selection tab, select the appropriate options.

For example, the following image shows the default options.



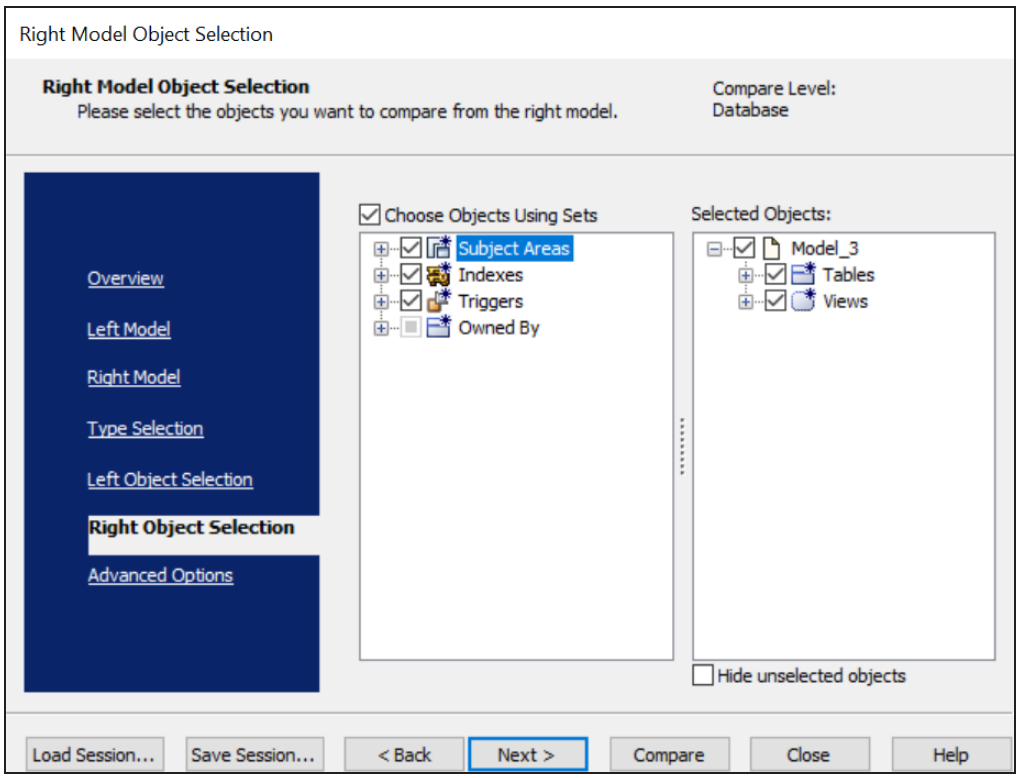
14. Click **Next** and on the Left Object Selection tab, select the appropriate options.

For example, the following image shows the default options.

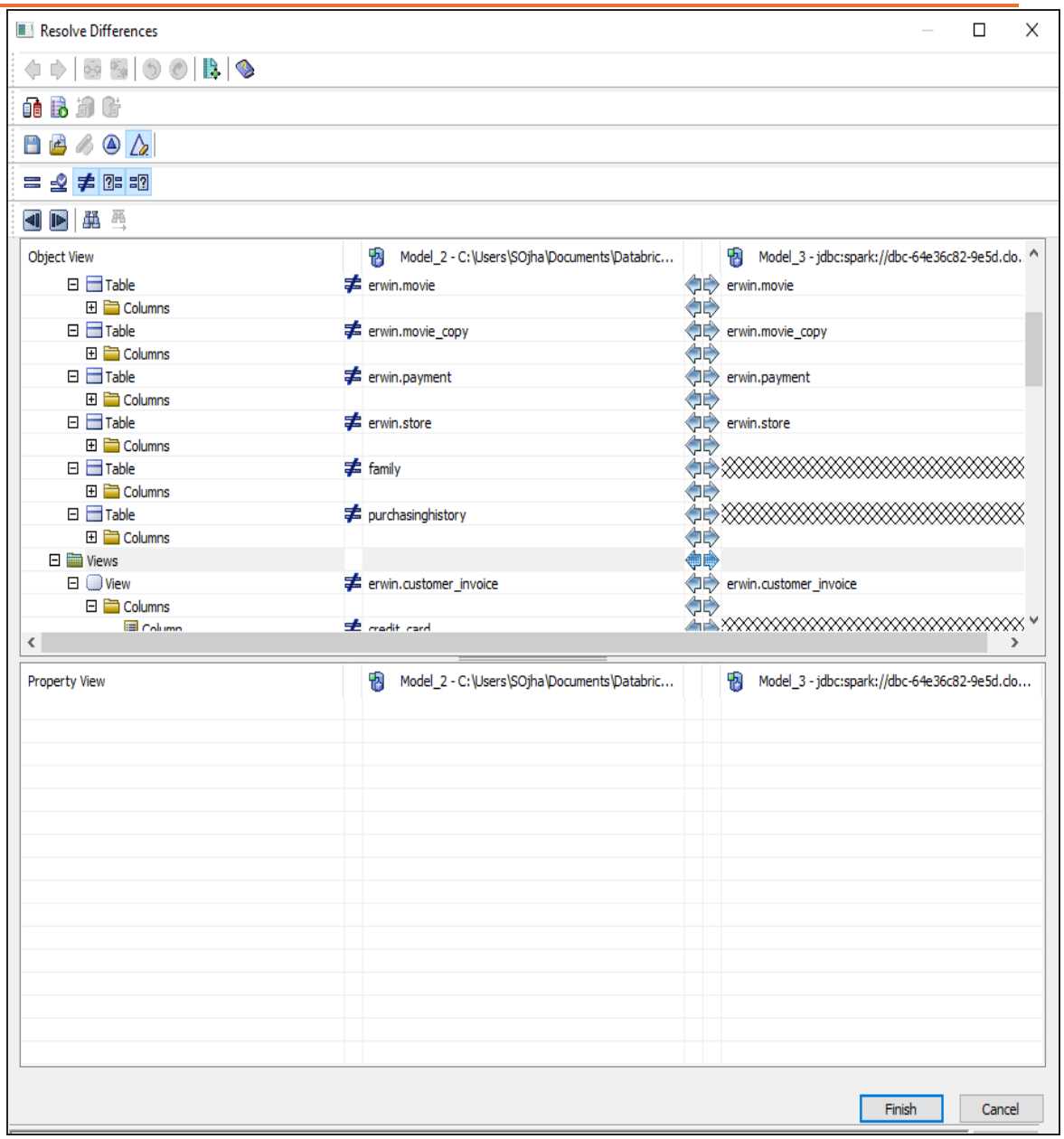


15. Click **Next** and on the Right Object Selection tab, select the appropriate options.

For example, the following image shows the default options.



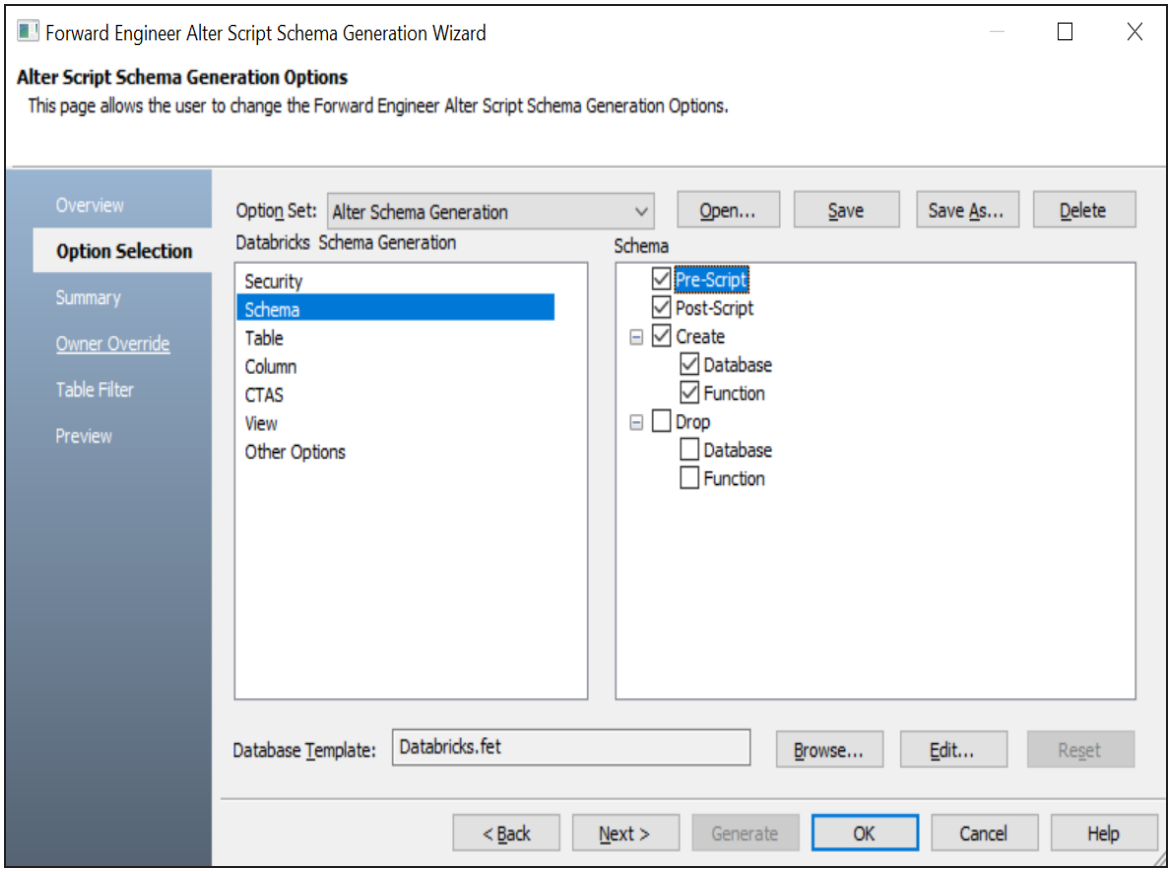
16. Click **Compare**.
    The comparison process runs, and the Resolve Differences dialog box appears. It displays the differences between your model and database.
    For example, the following image shows that the purchasinghistory table is available in your model but not in the database.



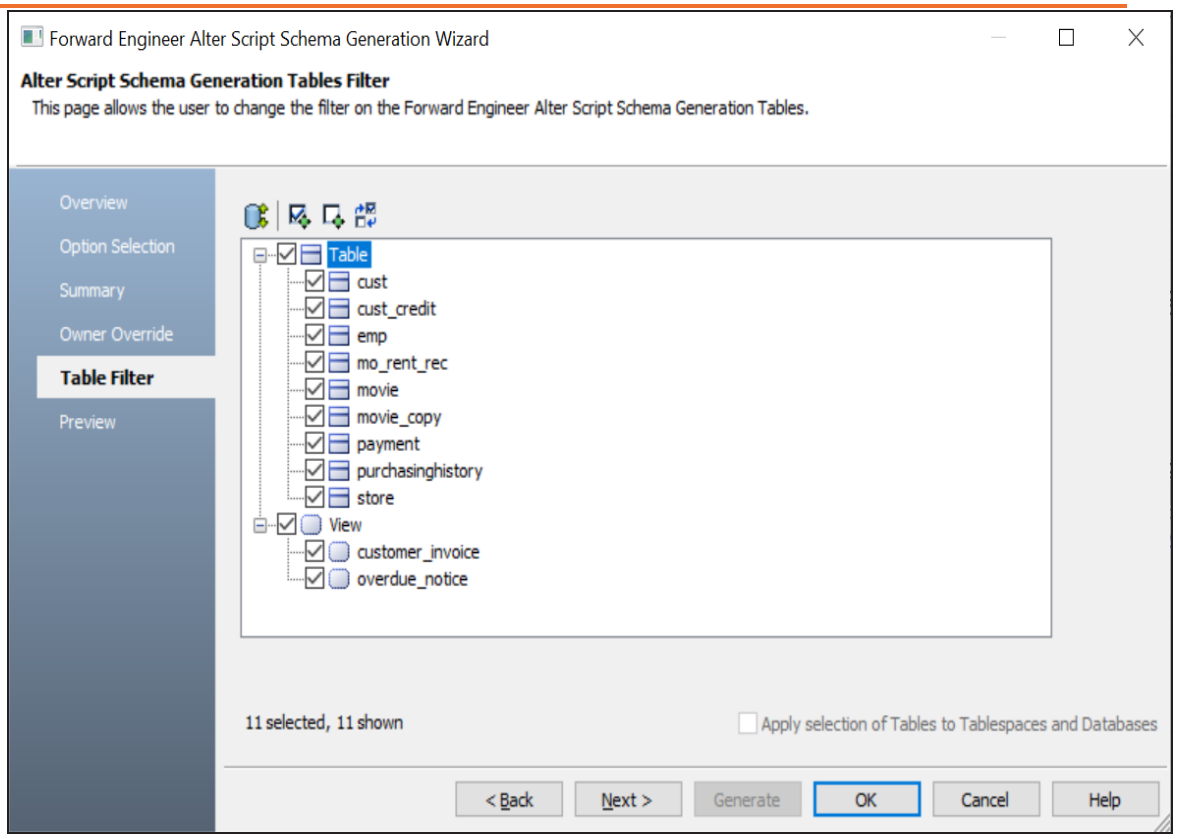
Select the purchasinghistory table and click . This will move the purchasinghistory table to the right model (from the database). Similarly, resolve other differences.

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 **Table Filter** and select or verify the tables 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 Databricks 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**.

DynamoDB Support

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

- Table
  - Item
  - Index

Following are the supported data types:

- STRING
- STRINGSET
- NUMBER
- NUMBERSET
- BINARY
- BINARYSET
- BOOLEAN
- LIST
- MAP
- NULL

DynamoDB implementation supports all erwin DM features and functions. The following sections walk you through these features:

- Reverse engineering models from database and script
- Forward engineering models to database
- Comparing changes using Complete Compare
- Converting relational models to DynamoDB models

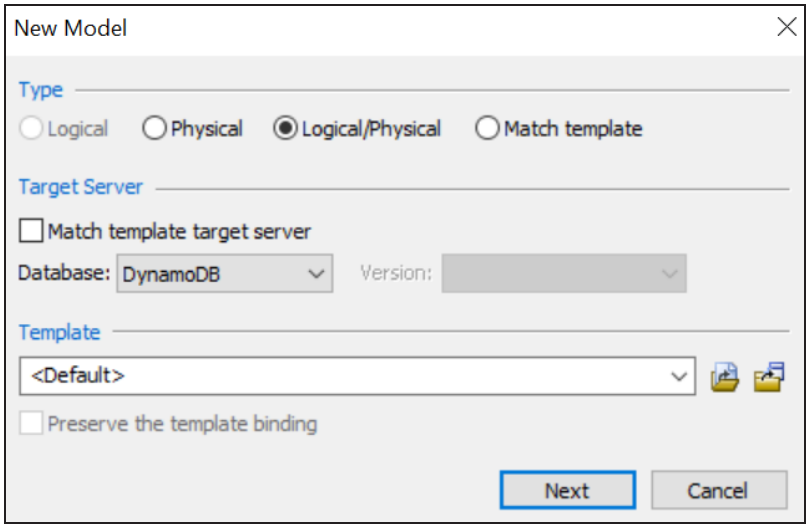
Reverse Engineering Models

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

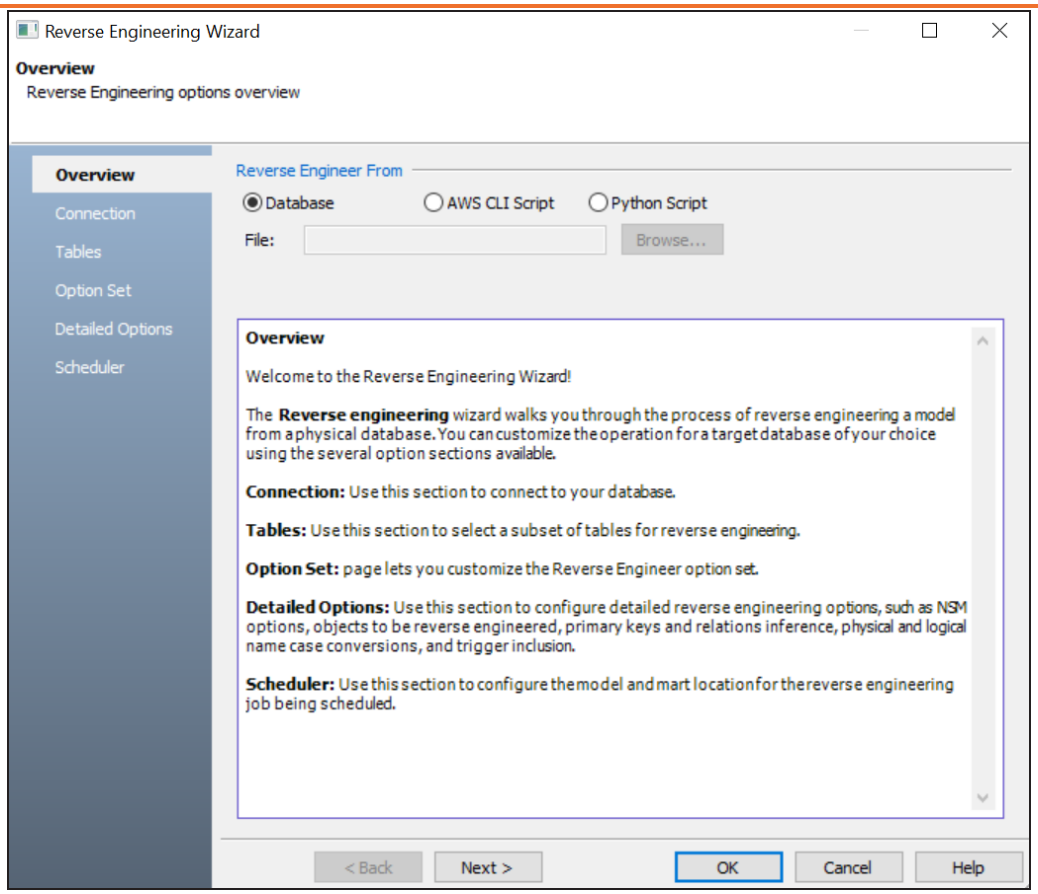
This topic walks you through the steps to reverse engineer a DynamoDB model. For detailed description of reverse engineering options, refer to the Reverse Engineering Options topic.

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 DynamoDB.



3. Click **Next**.
   The Reverse Engineer Process Wizard appears.



4. Click one of the following options:

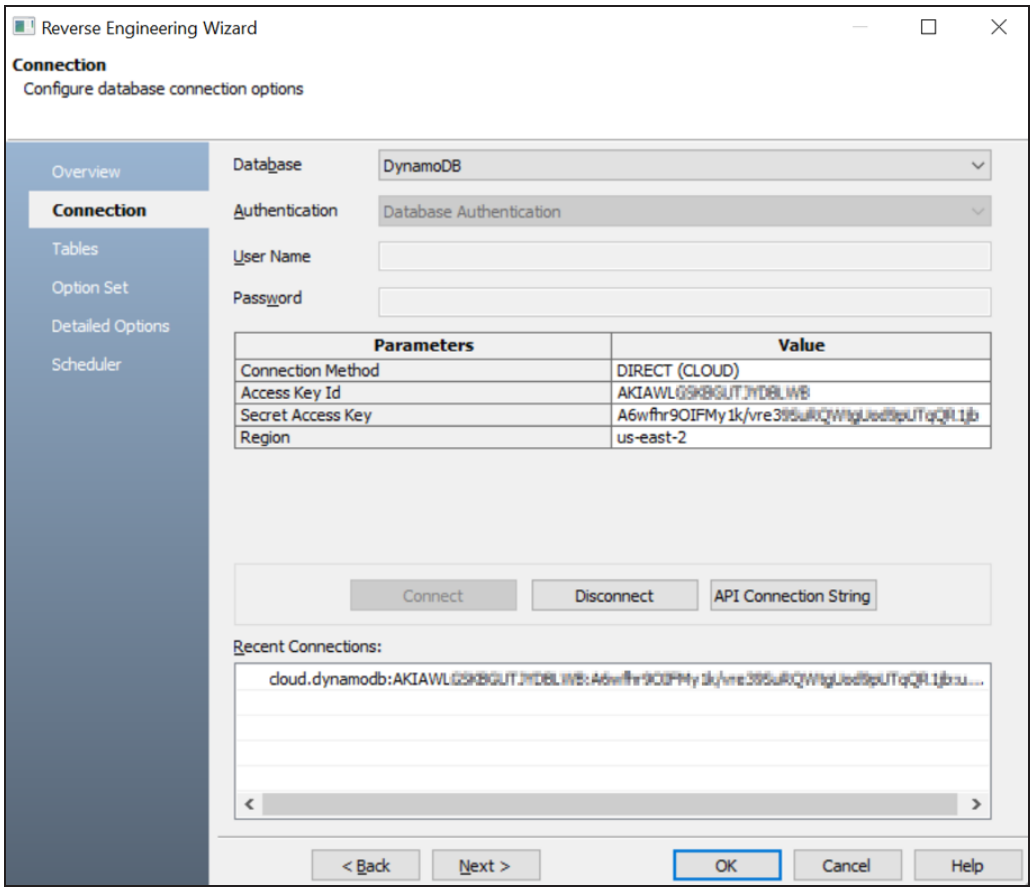
   - **Database**: Use this option to reverse engineer a model from a database.

     If you click **Database**, continue to step 5.

   - **<Script Name> Script**: Use one of these options to reverse engineer a model from a script. You can either use AWS CLI Script or Python Script option for reverse engineering. Selecting this option enables the **File** field. Click **Browse** and select the a script file from your directory.

     If you click **<Script Name> Script**, see step 11 below.

5. Click **Next**.
   The Connection tab appears.

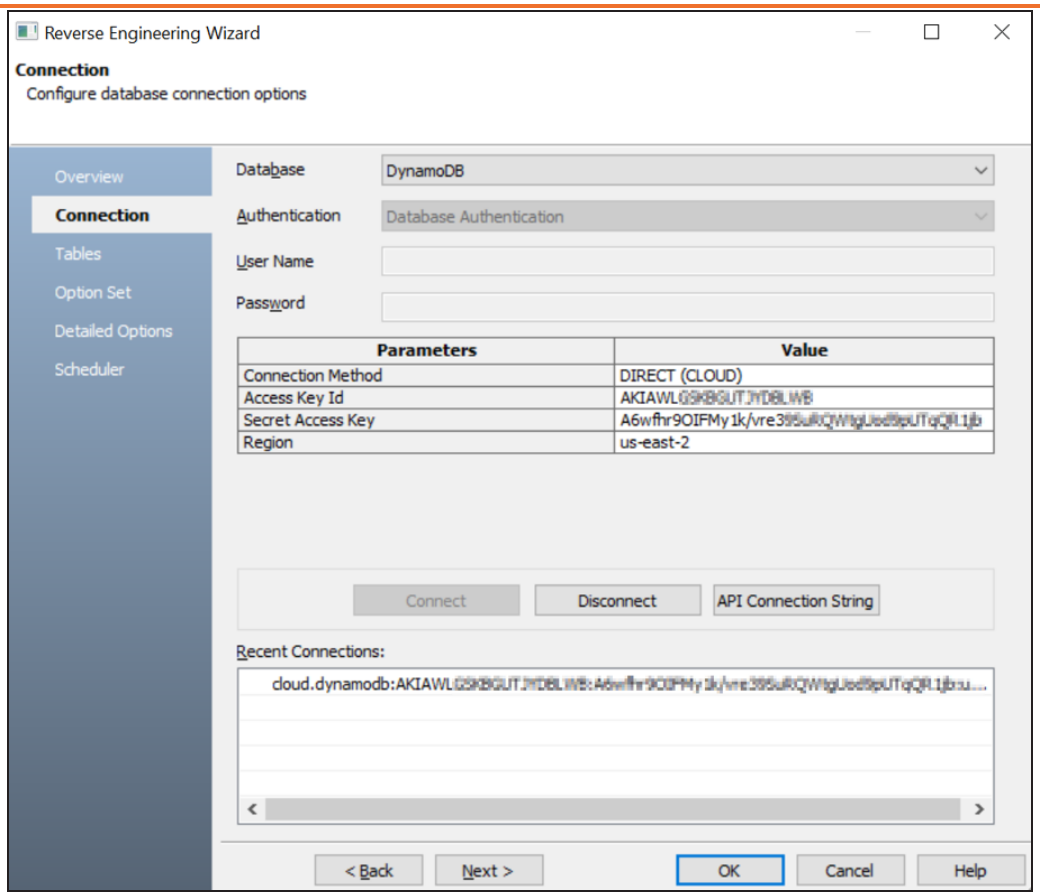


6. Enter your **User Name** and **Password**.
   The following table explains the connection parameters.

Parameter	Description	Additional Details
Connection Method	Specifies the type of connection you want to use. Select **DIRECT (CLOUD)** to connect to a database on a cloud. Or, select **DIRECT (LOCAL)** to	

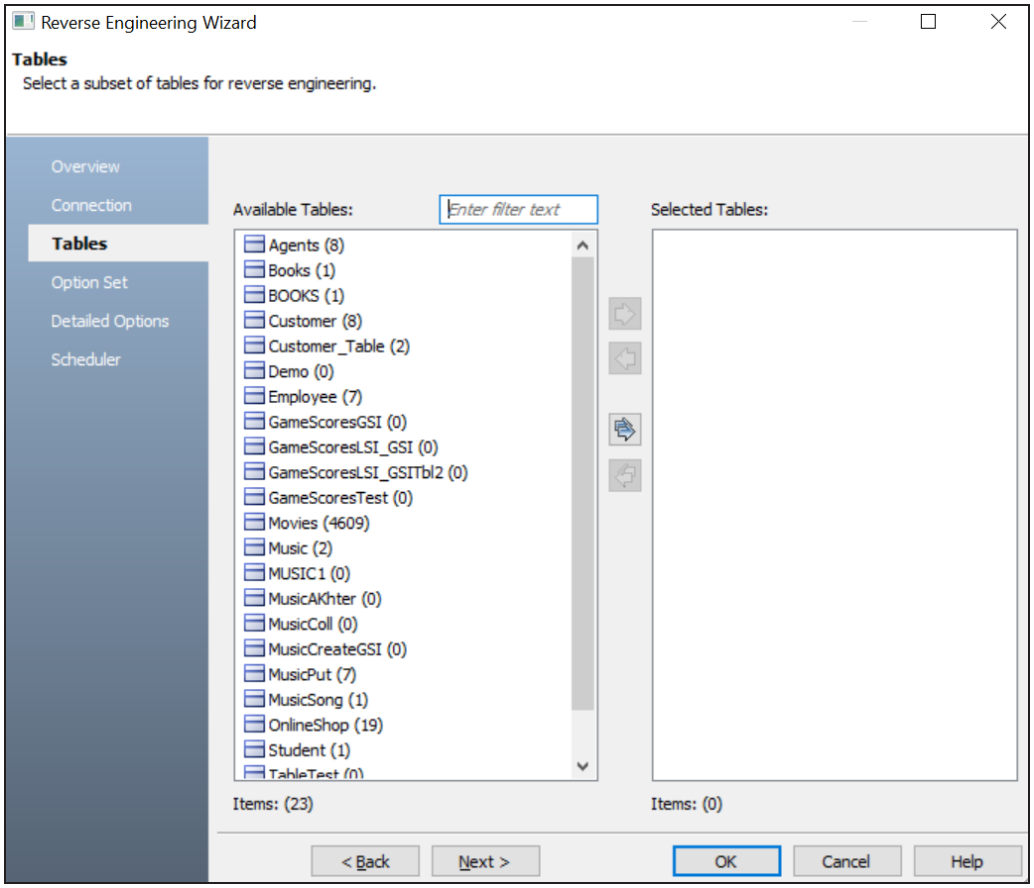
	connect to a local database.	
Hostname/IP	Specifies the host-name or IP address of the server where your database is hos-ted in the following format: *http://localhost*	This option is available when Connection Method is set to DIRECT (LOCAL).
Port	Specifies the port configured for your database	For example, *9142*. This option is available when Connection Method is set to DIRECT (LOCAL).
Access Key ID	Specifies access key id for connecting to a database on cloud	For example: *AKIAI00EXAMPLE56OSFODNN7* This option is available when Connection Method is set to DIRECT (CLOUD).
Secret Access Key	Specifies access key for authenticating the database con-nection	For example: *FEwJalrnMDMI/K7ENGXUt/EXAMPLEKEY/b1xwfiCu* This option is available when Connection Method is set to DIRECT (CLOUD).
Region	Specifies the loc-ation of the remote database	For example: *us-east-2* This option is available when Connection Method is set to DIRECT (CLOUD).

7. Click **Connect**.
   On successful connection, your connection information is displayed under Recent Con-nections.

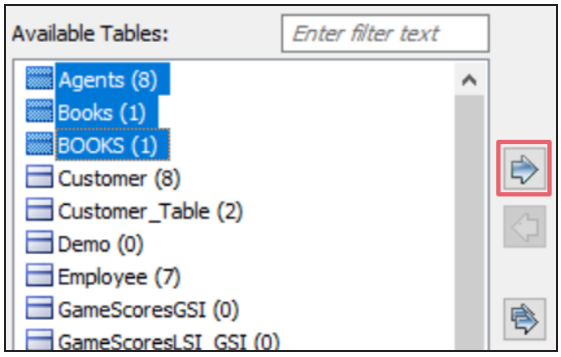

Reverse Engineering Wizard
Connection
Configure database connection options
Overview
Connection
Tables
Option Set
Detailed Options
Scheduler
Database
DynamoDB
Authentication
Database Authentication
User Name
Password
Parameters
Value
Connection Method
DIRECT (CLOUD)
Access Key Id
AKIAWL
Secret Access Key
A6wfhr9OIFMy1k/vre3
Region
us-east-2
Connect
Disconnect
API Connection String
Recent Connections:
cloud.dynamodb:AKIAWL
< Back
Next >
OK
Cancel
Help

8. Click **Next**.

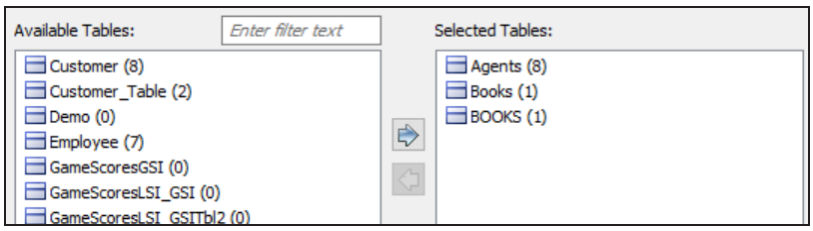
The Tables tab appears. It displays a list of available tables.



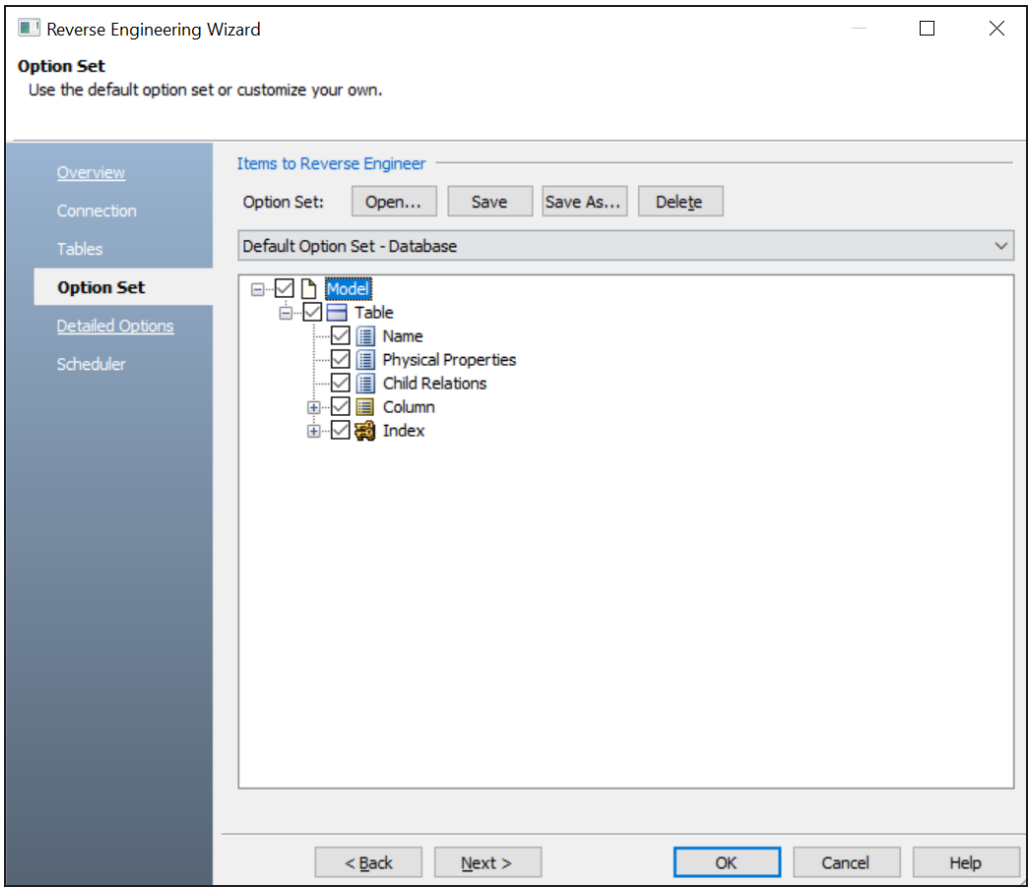
9. Under **Available Tables**, select the tables that you want to reverse engineer. Then, click  .



This moves the selected tables under Selected Tables.

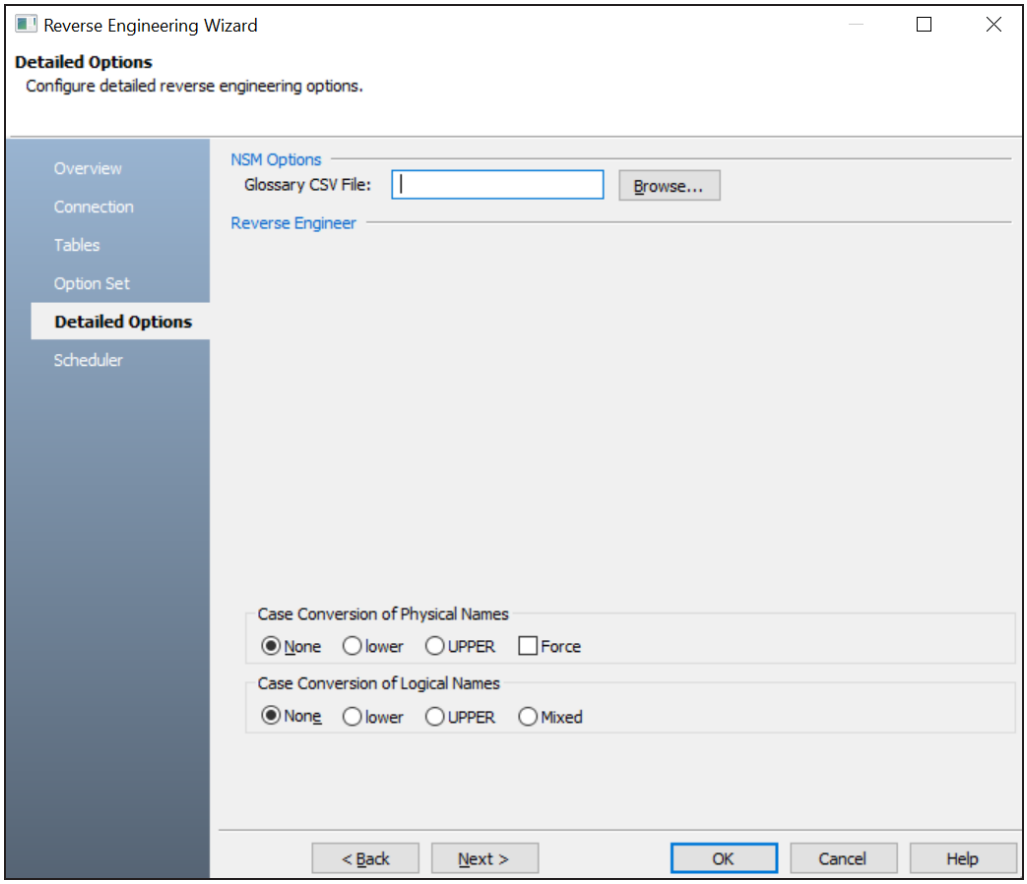


10. Click **Next**.
    The Option Set tab appears. It displays the default option set. You can either use the default or a custom option set.



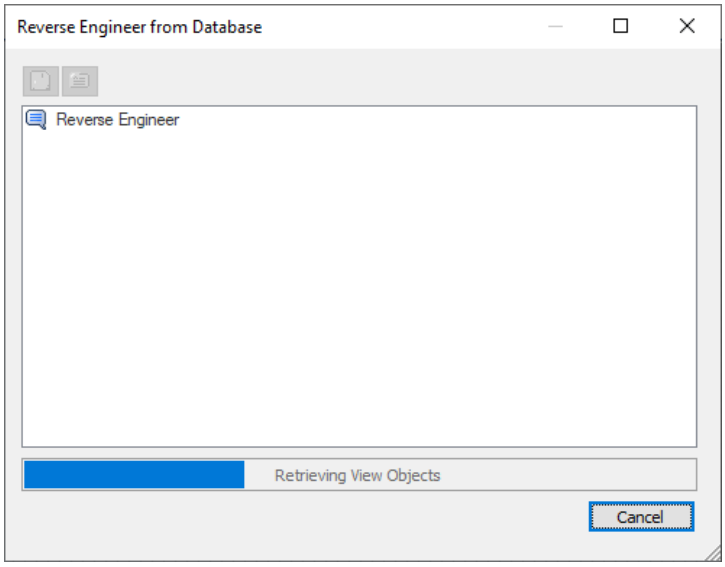
11. Click **Next**.
    The Detailed Options tab appears. Set up appropriate options based on your

requirement.

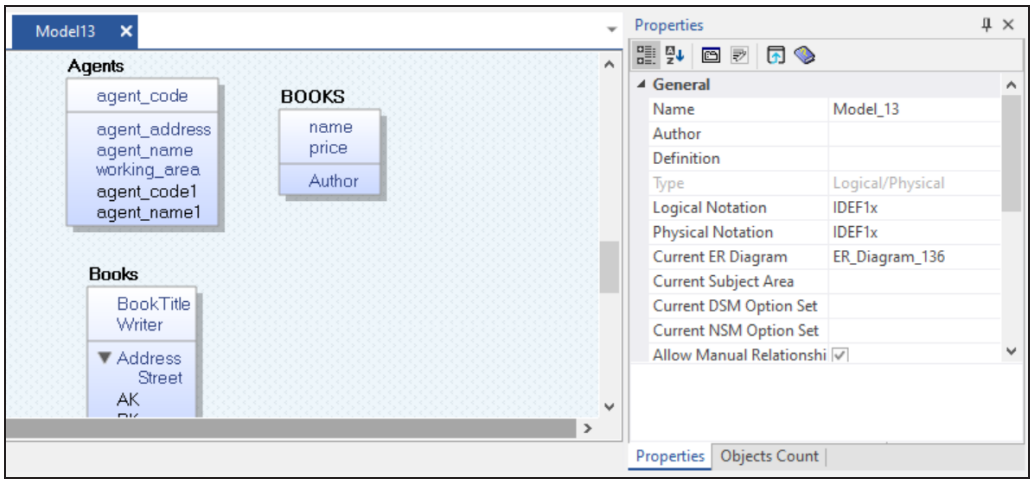


12. Click **OK**.

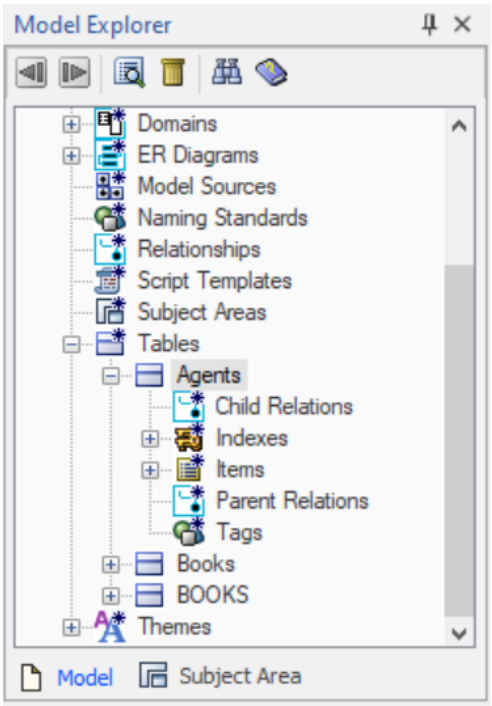
The reverse engineering process starts.



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

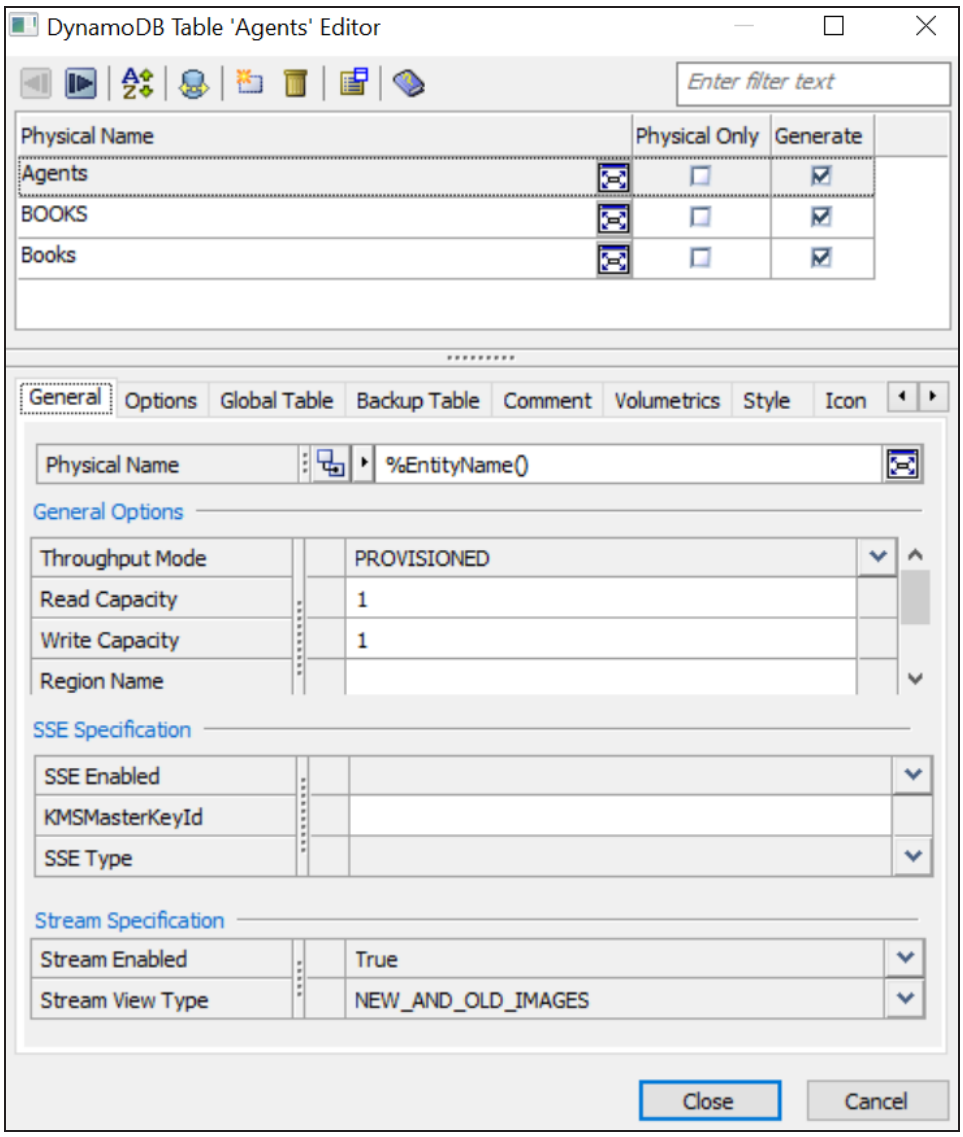


Along with Tables, other object such as Indexes and Items are also retrieved.



You can view these objects via the model diagram or view their properties via the Model Explorer. Right-click an object and then, click the required Properties option. For example, on the model diagram, right click a table and then, click Table Properties. The DynamoDB

Table Editor appears. You can view the table properties here.



Reverse Engineering Options for DynamoDB

The following are the reverse engineering options for DynamoDB in erwin DM.

Overview

Parameter	Description	Additional Information
Reverse Engineer From	Specifies whether you want to reverse engineer from a script or database	**Database**: Indicates that the model is reverse engineered from a remote or local database **AWS CLI Script**: Indicates that the model is reverse engineered from a Command Line Interface (CLI) script **Python Script**: Indicates that the model is reverse engineered from a python script
File	Specifies the script file location	This option is available when Script File is selected.

Connection

Parameter	Description	Additional Details
Connection Method	Specifies the type of connection you want to use. Select **DIRECT (CLOUD)** to connect to a database on a cloud. Or, select **DIRECT (LOCAL)** to connect to a local database.	
Hostname/IP	Specifies the hostname or IP address of the server where your database is hosted in the following format: *http://localhost*	This option is available when Connection Method is set to DIRECT (LOCAL).
Port	Specifies the port con-	For example, *9142*.

	figured for your database	This option is available when Connection Method is set to DIRECT (LOCAL).
Access Key ID	Specifies access key id for connecting to a database on cloud	For example: *AKIAI00EXAMPLE56OSFODNN7* This option is available when Connection Method is set to DIRECT (CLOUD).
Secret Access Key	Specifies access key for authenticating the database connection	For example: *FEwJalrnMDMI/K7ENGXUt/EXAMPLEKEY/b1xwfiCu* This option is available when Connection Method is set to DIRECT (CLOUD).
Region	Specifies the location of the remote database	For example: *us-east-2* This option is available when Connection Method is set to DIRECT (CLOUD).

Tables

Parameter	Description	Additional Information
Available Tables	Specifies a list of available tables	
Selected Tables	Specifies a list of selected tables for reverse engineering	

Option Sets

Parameter	Description	Additional Information
Option Set	Specifies the option set template for reverse engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save the configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at some external location. **Delete**: Use this option to delete an

		option set.
<Option Set Name>	Specifies the objects to be reverse engineered according to the selected option set. You can edit this list.	

Detailed Options

Parameter	Description	Additional Information
NSM Options	Specifies the naming standard glossary file in the .CSV format	
Case Conversion of Physical Names	Specifies how the case conversion of physical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Force**: Indicates whether the physical name property of all the logical/physical models is overridden. If this option is enabled, the logical/physical link is broken between the logical and physical name. If this option is not enabled, all logical and physical names are set to the same value after the process completes.
Case Conversion of Logical Names	Specifies how the case conversion of logical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Mixed**: Indicates that the mixed-case logical names are preserved

Scheduler

Parameter	Description	Additional Information

Model	Specifies the location and name of the reverse engineered model	For example: C:\Scheduler\<Model Name>.erwin When you schedule a job on a remote server, ensure the model path is same for remote and local server.
Mart Folder	Specifies the location or library in your mart where the reverse engineered model is saved	To use this option, ensure that you are connected to a mart. For more information, refer to the Connecting to Mart topic.
Complete Compare	Specifies whether the Complete Compare (CC) process should run while reverse engineering	
Output File	Specifies the location of the CC output file generated	
File	Specifies that the target model location is on the local system	
Mart	Specifies that the target model location is in the mart	
Using Latest Version	Specifies whether the target model is the latest version of the model in the mart	This option is available only when Mart is selected.
Save To Mart	Specifies whether the reverse engineered model is saved to the mart	This option is available only when Using Latest Version is selected.
Target Model	Specifies the location of the target model for CC	
Option Set	Specifies the option set that is used for CC	**Advanced Default Option Set**: Indicates that all erwin DM metadata is included. CC works slowest with this option. **Speed Option Set**: Indicates that only the essential metadata is included. CC works the fastest

		with this option set. **Standard Default Option Set**: Indicates that standard metadata is included. CC works fast with this option set compared to the Advanced option set.

Forward Engineering Models

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

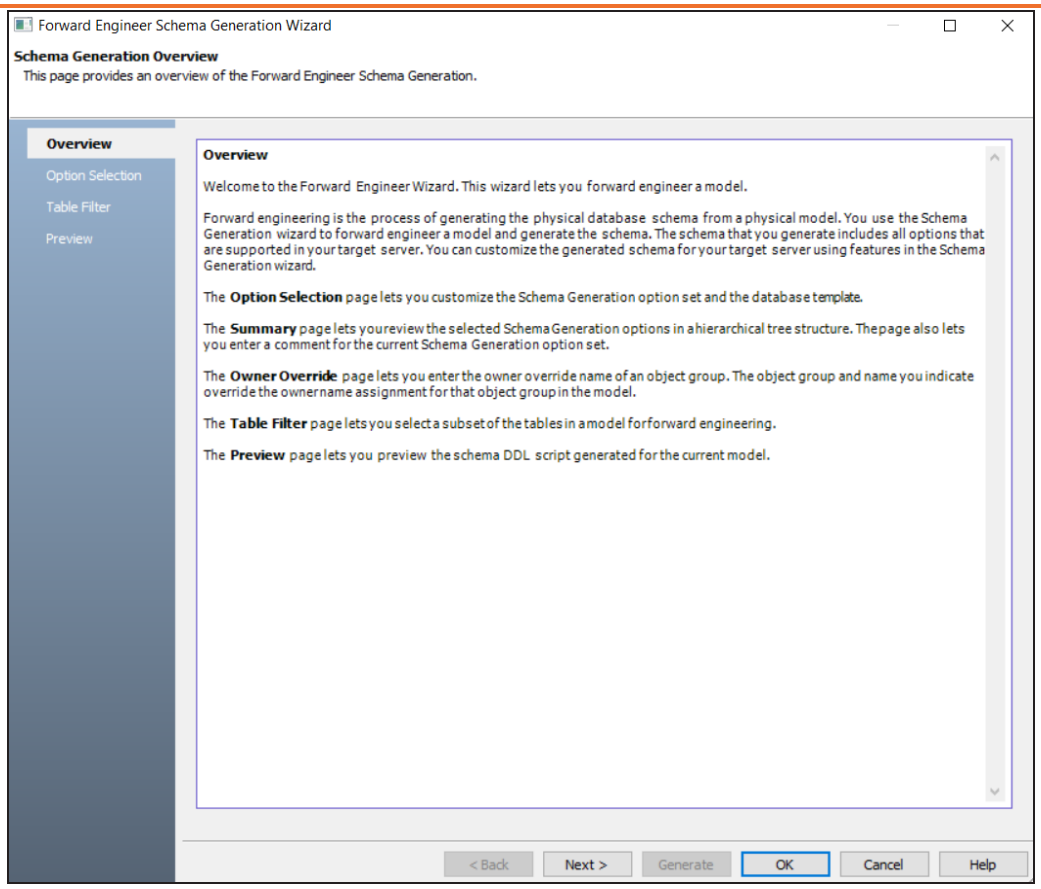
This topic walks you through the steps to forward engineer a DynamoDB model. For detailed description of forward engineering options, refer to the Forward Engineering Options topic.

To forward engineer a DynamoDB model:

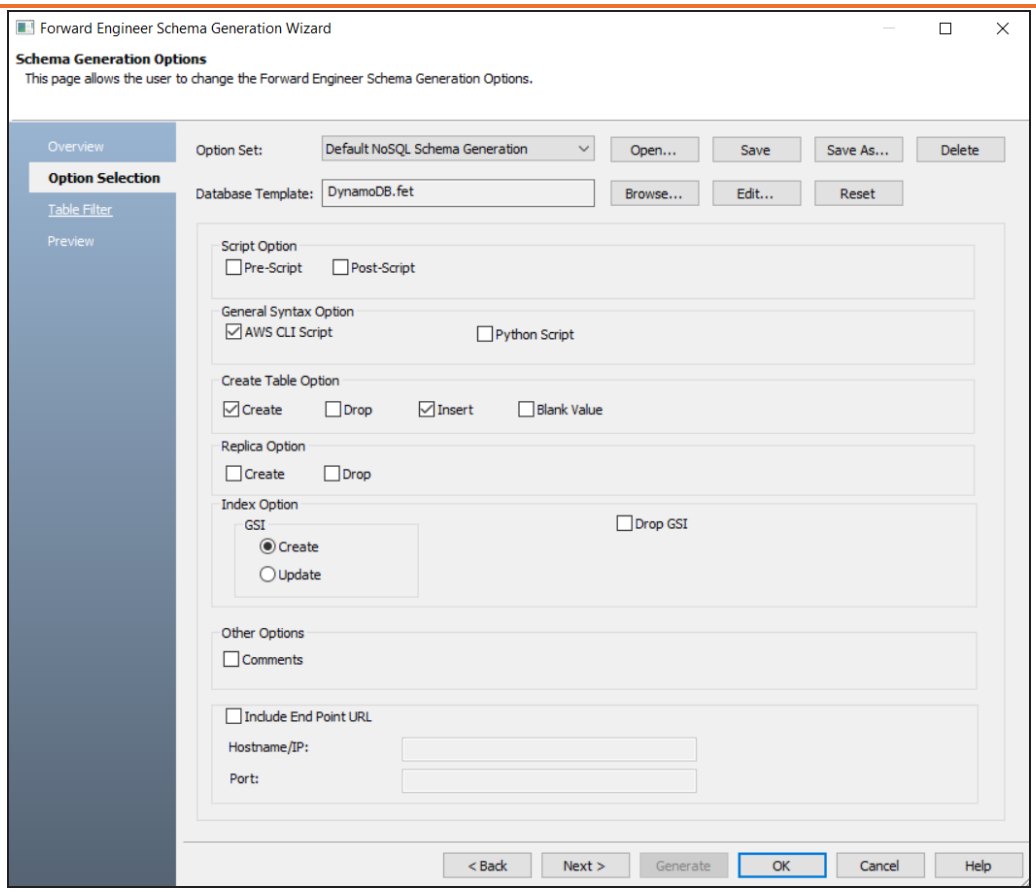
1. Open your DynamoDB model.

   
   Ensure that you are in the Physical mode.

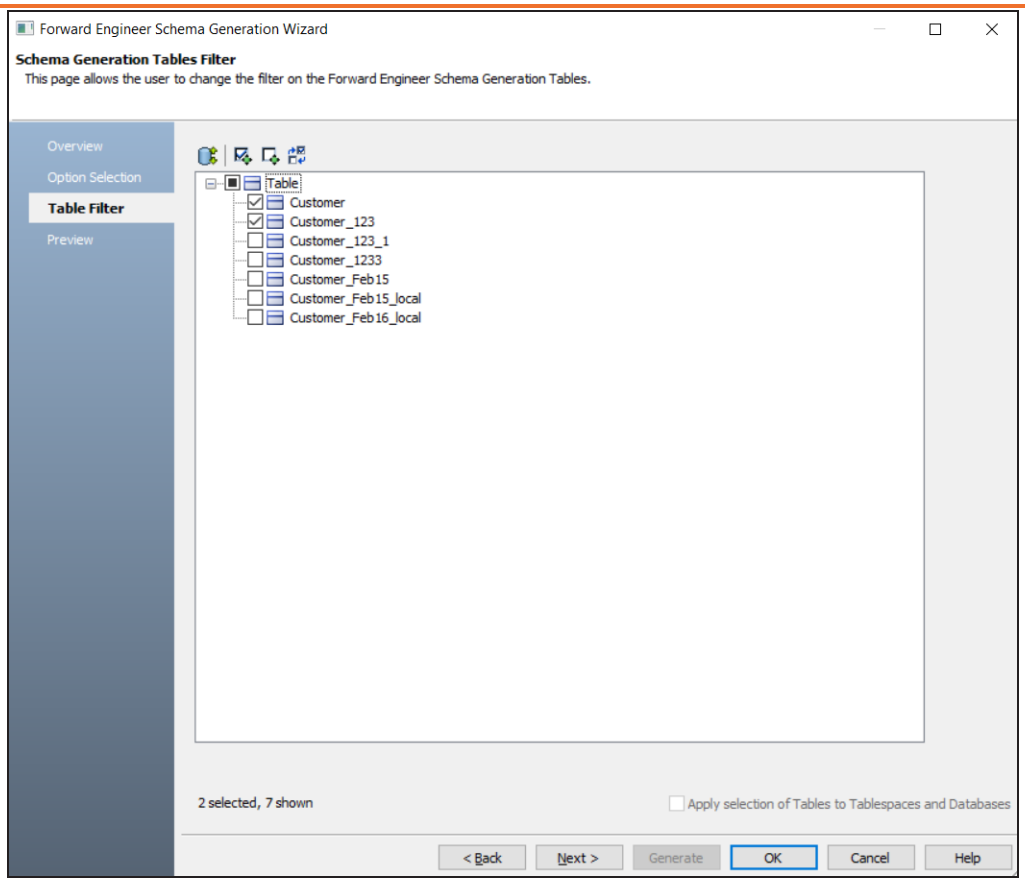
2. Click **Actions** > **Schema**.
   The Forward Engineer Schema Generation Wizard appears.



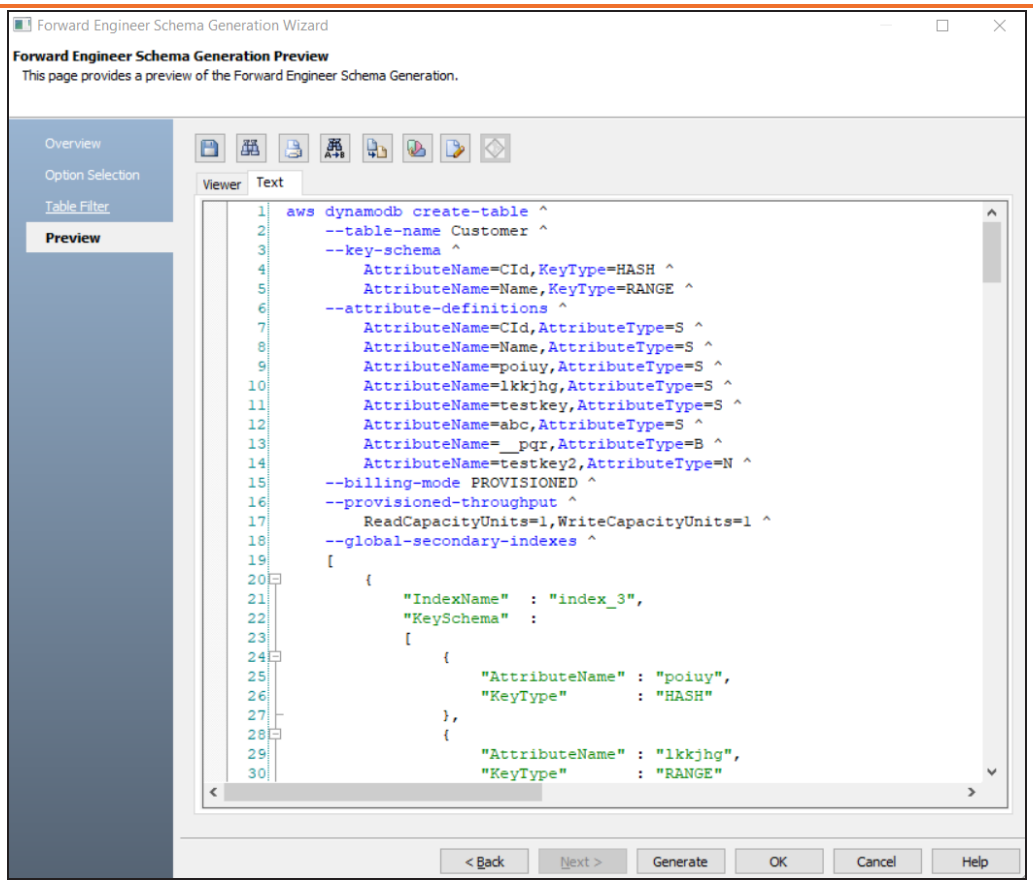
3. Click **Option Selection**.
   The Option Selection tab displays the default option set. Clear the **Drop** check boxes and select other syntax check boxes as required.



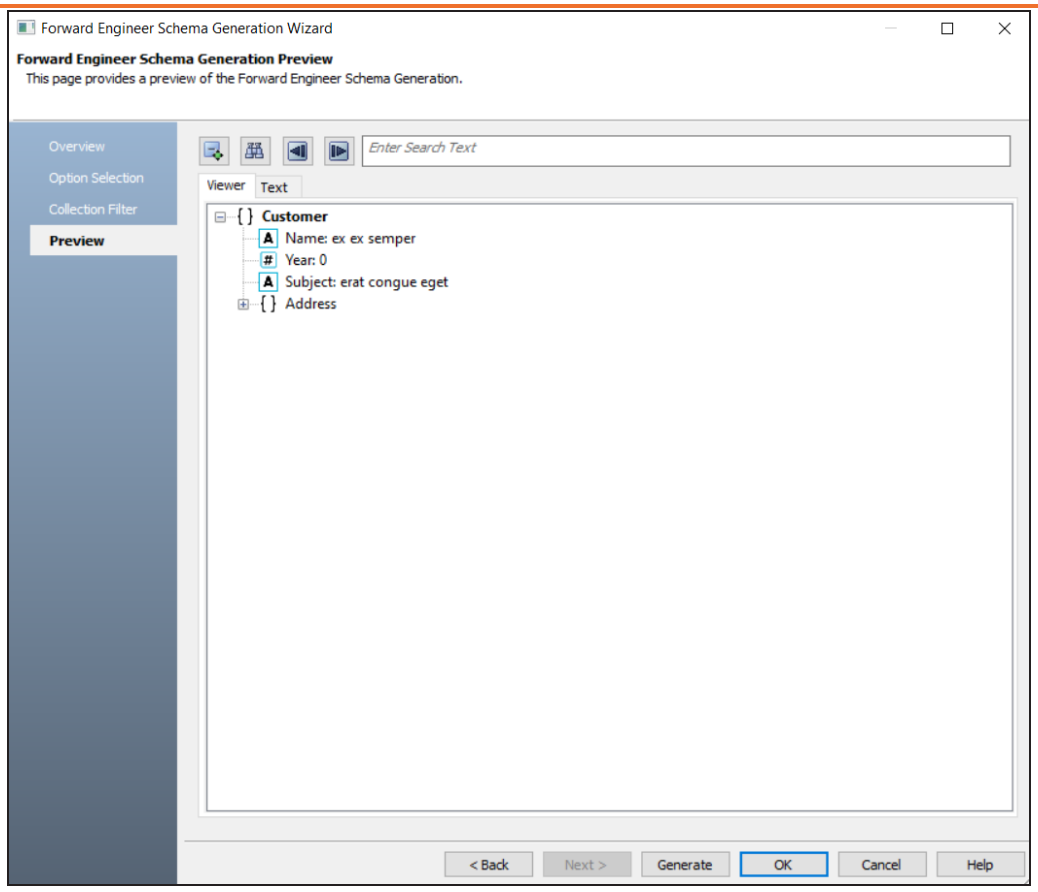
4. Click **Next**.
   The Table Filter tab appears. It displays a list of tables available in your model.



5. Select the tables that you want to forward engineer.
6. Click **Preview** to view the schema and its script.
   The schema is available in the Text tab.



The schema is available in the Viewer tab. To view schema on this tab, select the JSON script of a statement in the Text tab.

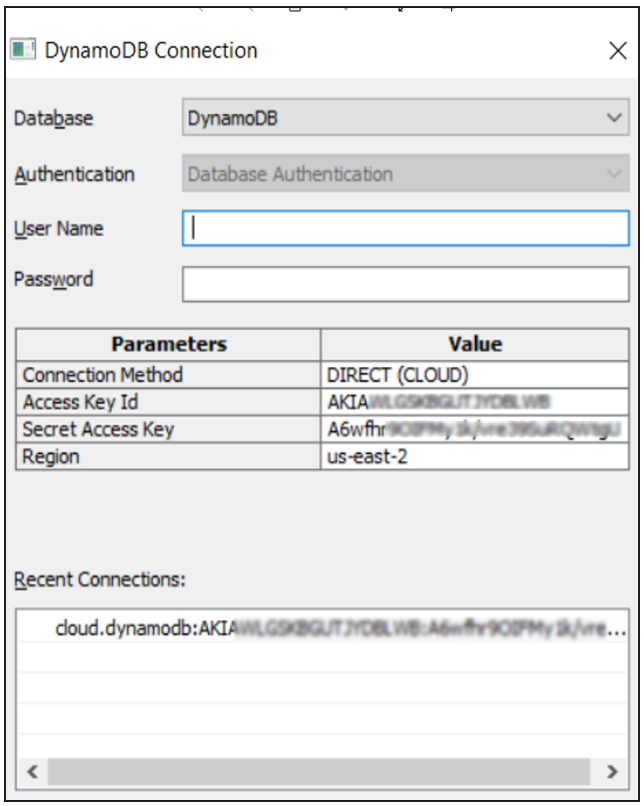


Use the following options:

- **Copy** ( ): Use this option to copy the script.
- **Save** ( ): Use this option to save the generated script into a single or multiple files in the TXT format.
- **Search** ( ): Use this option to search through the generated schema.
- **Print** ( ): Use this option to print the generated schema.
- **Replace** ( ): Use this option to find and replace in the generated schema.

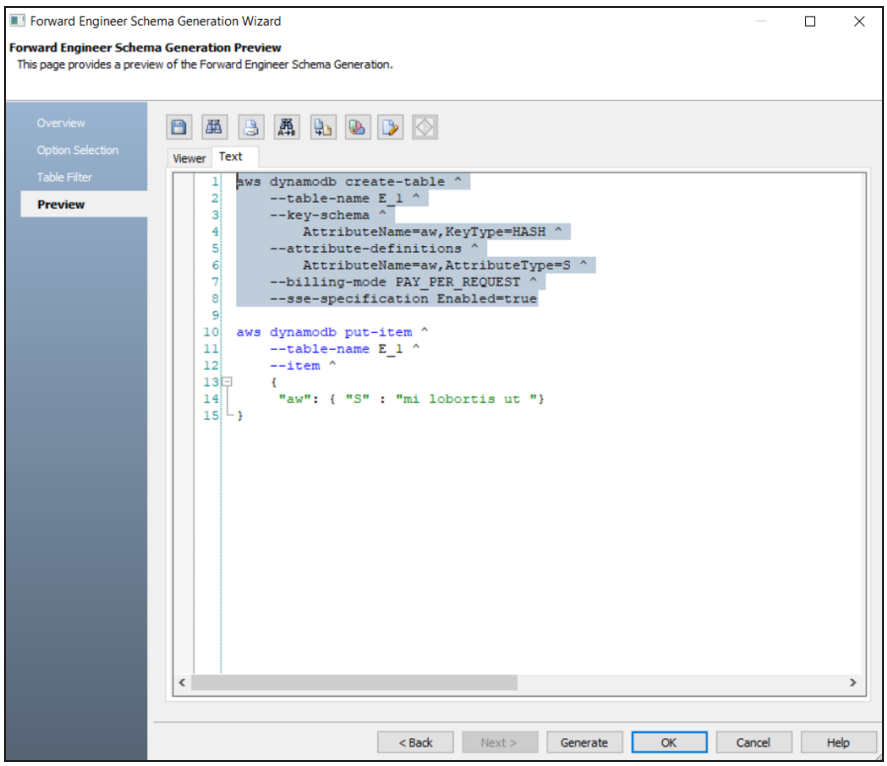
- **Text Options** (): Use this option to configure the preview text editor's look and feel, such as window, font, syntax color settings.
- **Error Check** (): Use this option to run an error check. Based on the results, you can correct the generated script.

7. Click **Generate**.
   The DynamoDB Connection editor appears.



Generating forward engineering script for DynamoDB is not executed as expected and displays error when you click Generate. Use one of the following methods to generate DynamoDB script:

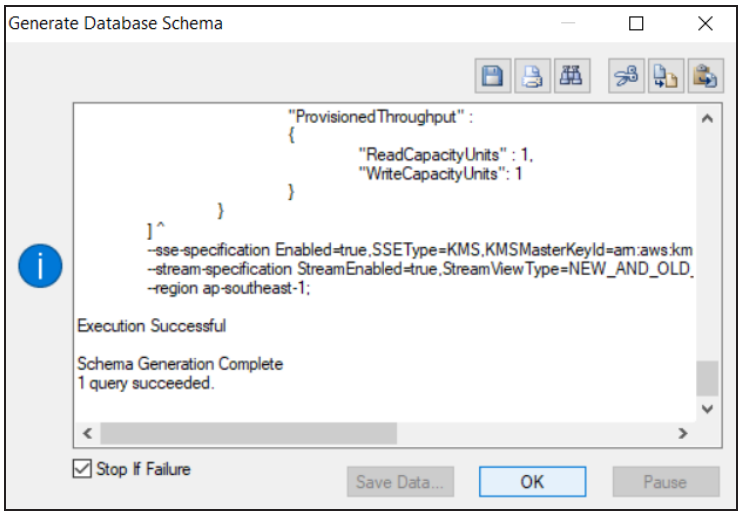
- Save the script and execute the file using a command line.
- Select and execute one statement at time to generate script. For example, in the below image, one statement is selected in the script.



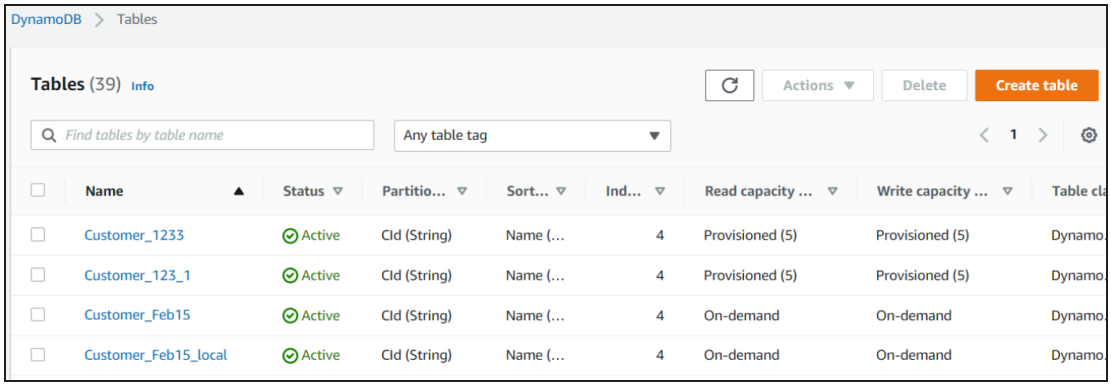
8. Enter username, password, and appropriate connection parameters to connect the required database. Then, click **Connect**.

> The objects move to a database entered on the DynamoDB Connection page irrespective of the databases entered on the object editor pages. If you want to move objects to databases as entered on object editors page then do not enter any database on the DynamoDB Connection page.

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



Based on the generated schema, the tables are added to the database based on the AWS region.



Forward Engineering Options for DynamoDB

Following are the forward engineering options for DynamoDB.

Option Selection

Parameter	Description	Additional Information
Option Set	Specifies the option set template for forward engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save a configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at an external location. **Delete**: Use this option to delete an option set.
Database Template	Specifies the database template for controlling schema generation	**Browse**: Use this option to browse and select a database template. **Edit**: Use this option to edit a template in the Template Editor. **Reset**: Use this option to reset the Database Template option.
Script Option	Specifies the script option for schema generation	**Pre-Script**: Indicates whether pre-scripts attached to the schema are executed **Post-Script**: Indicates whether the post-scripts attached to the schema are executed
General Syntax Option	Specifies the syntax options for schema generation	**AWS CLI**: Indicates whether the AWS CLI syntax for databases is executed

		Python: Indicates whether the Python syntax for databases is executed
Create Table Option	Specifies the table options for schema generation	**Create**: Indicates whether the create syntax for tables is executed **Drop**: Indicates whether the drop syntax for tables is executed **Insert**: Indicates whether to include fields in respective table in the schema **Blank Value**: Indicates whether to replace the random values in a field with a blank value
BackUp Table Option	Specifies the back up table options for schema generation	**Create**: Indicates whether the create syntax for tables is executed **Drop**: Indicates whether the drop syntax for tables is executed **Create Continuous BackUps**: Indicates whether to create continuous back up of the tables
Replica Option	Specifies the replica options for schema generation	**Create**: Indicates whether the create syntax for tables is executed **Drop**: Indicates whether the drop syntax for tables is executed
GSI	Specifies whether the Global Secondary Index (GSI) option is enabled for schema generation. Use one of the following options: **Create**: Specifies to create GSI for the new and existing tables during schema generation	**Drop GSI**: Indicates whether the Drop syntax for GSI is executed

	Update: Specifies to create GSI using update statement for a tables during schema generation	
Comments	Indicates whether comments are included in the schema	
Include End Point URL	Specifies whether to include end point URL information such as hostname and port number in the script during schema generation	

Table Filter

Parameter	Description	Additional Information
Tables	Specifies the selected tables for schema generation	
Display either Logical Names or Physical Names	Specifies the database template for controlling schema generation	**Logical Names**: Indicates that only logical names of the tables are included in the generated schema **Physical Names**: Indicates that only physical names of the tables are included in the generated schema **Physical Names, show owner**: Indicates that physical names and owners of the tables are included in the generated schema **Physical Names, show owner using User**: Indicates that the physical names and owners of the tables are included in the generated schema. Owners of the tables are displayed using User.
Select all of the items in the list	Use this option to select all the tables in the list.	
Unselect all of the items in	Use this option to clear all the tables.	

the list		
Select all unselected items, and unselect all selected items	Use this option to select all the unselected tables and clear all the previously selected tables.	

Preview

Parameter	Description	Additional Information
Text	Displays the schema in the text editor	**Save**: Use this option to save the generated schema into single or multiple files. **Search**: Use this option to search through the generated schema. **Print**: Use this option to print the generated schema. **Replace**: Use this option to find and replace text in the generated schema. **Copy**: Use this option to copy the selected text in the schema. **Text Options**: Use this option to edit window settings, fonts, syntax color. **Error Check**: Use this option to view error report for the generated schema. **Git**: Use this option to commit the FE script to a Git repository.

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.

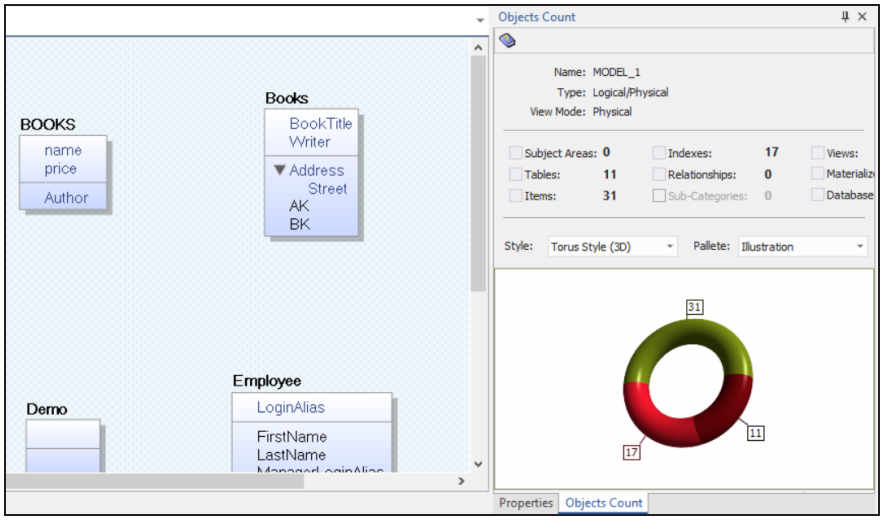
This topic walks you through the steps to compare a DynamoDB model with database.

To compare models with database:

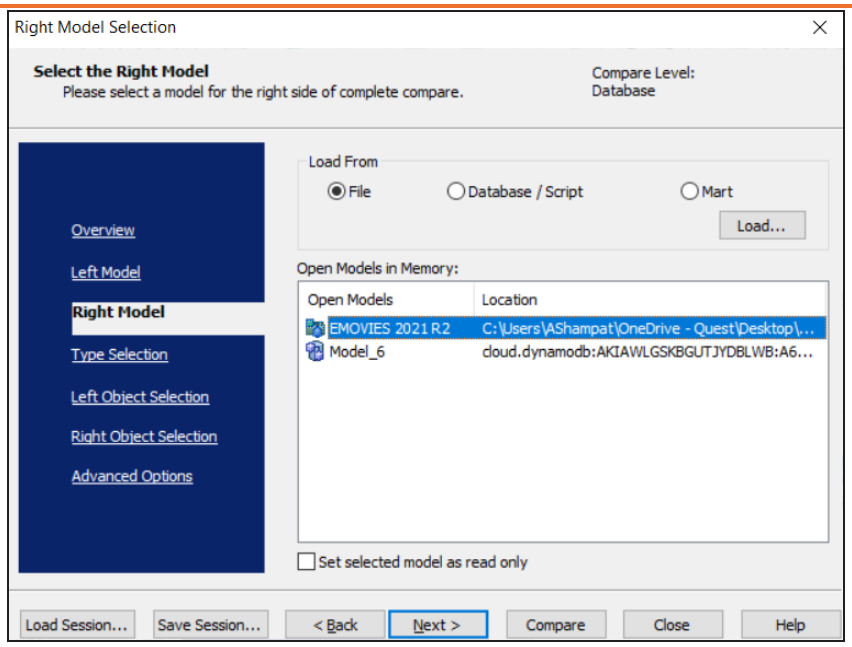
1. Open your DynamoDB model.

   Ensure that you are in Physical mode.

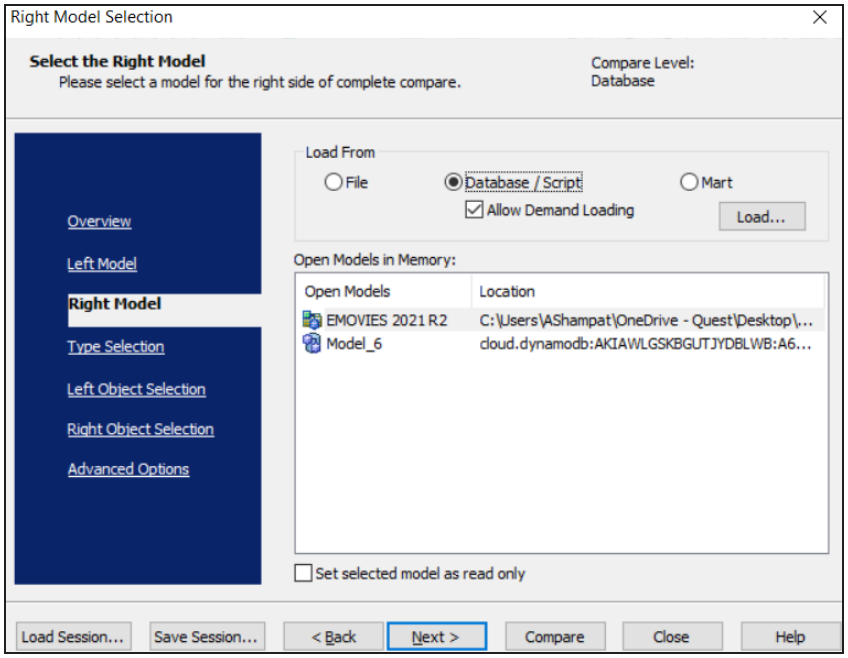
   For example, the following image uses a DynamoDB model with 11 tables.



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.

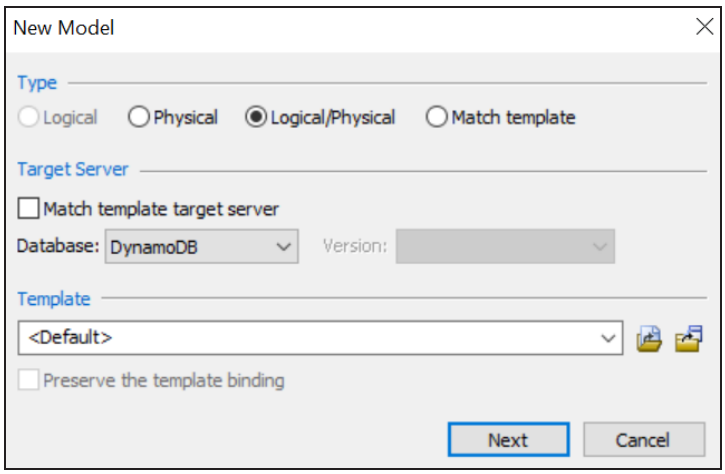


3. Click **Database/Script**.
   By default, the Allow Demand Loading option is selected.

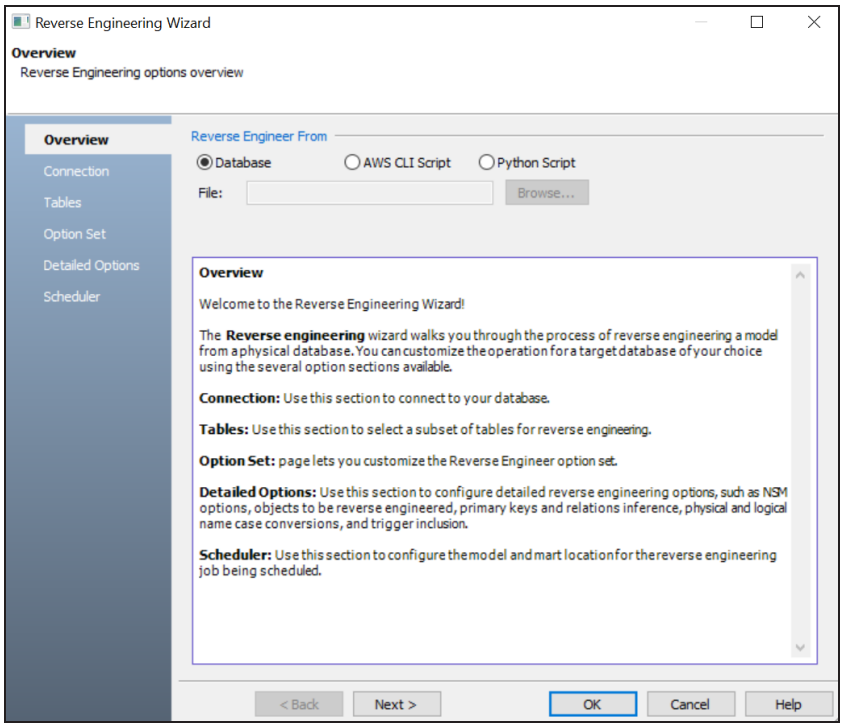


4. Click **Load**.
   The New Model dialog box appears. This starts the reverse engineering process to pull

a model from the database to compare.



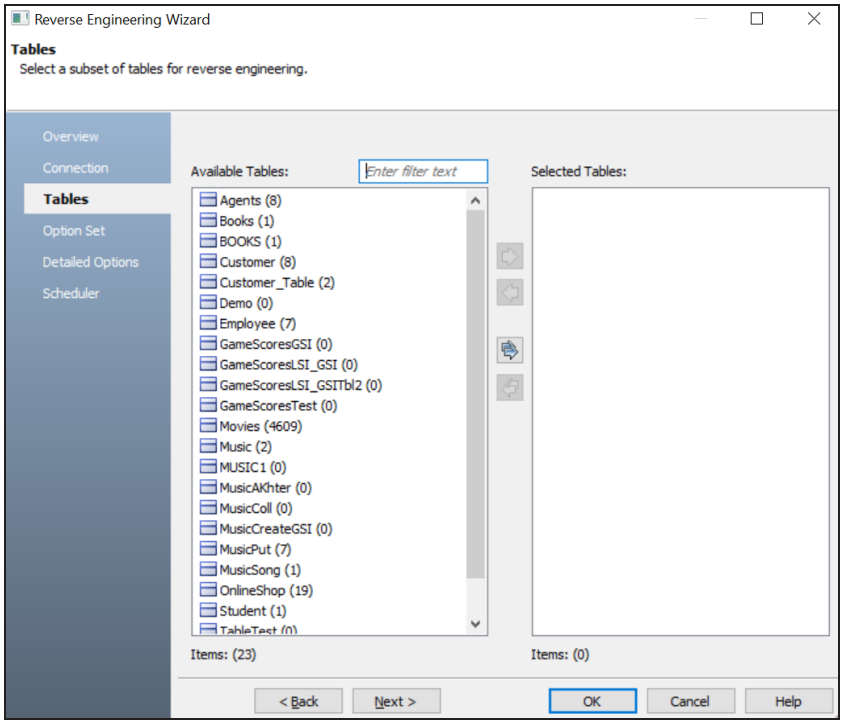
5. Ensure that the Database is set to DynamoDB. Then, click **Next**.
   The Reverse Engineer Wizard appears.



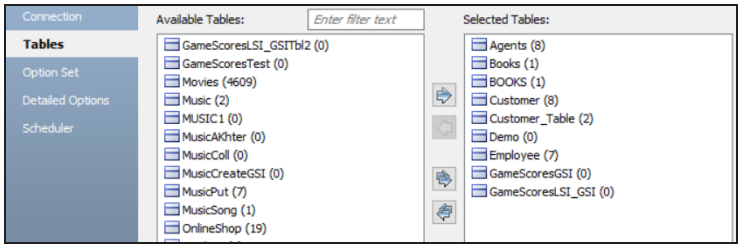
6. Click **Database**. Then, click **Next**.
   The Connection section appears. Use this section to connect to the database from

which you want to reverse engineer the model.

7. After the connection is established, click **Next**.
   The Tables section appears. It displays a list of available tables.

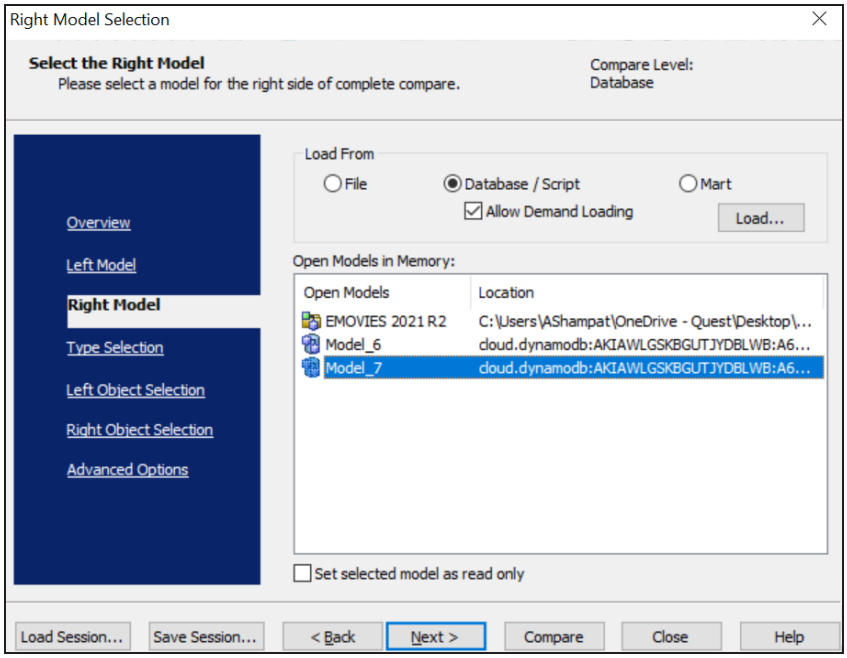


8. Under **Available Tables**, select the tables that you want to reverse engineer. Then, click  .
   This moves the tables under Selected Tables.



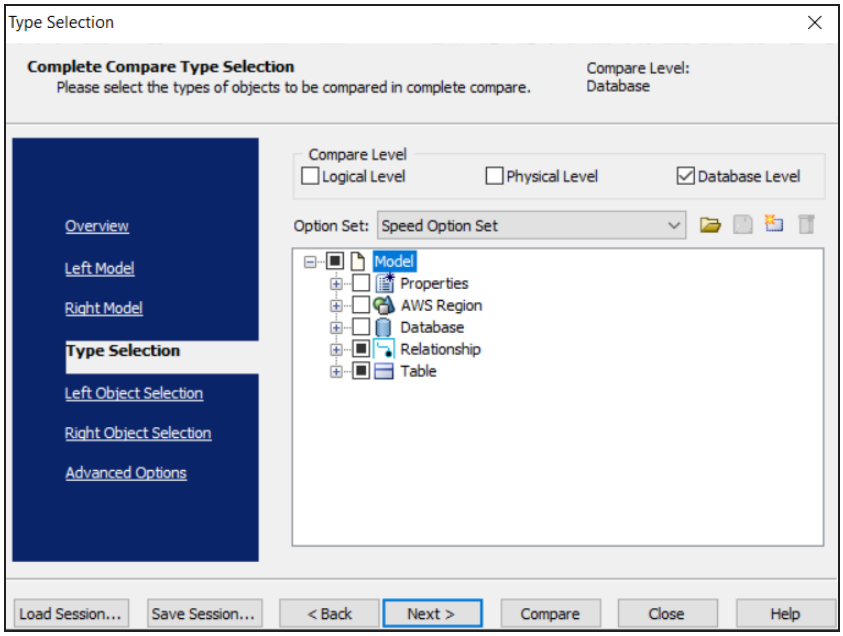
9. Click **Next** and in the Option Set section, keep the default configuration.
10. Click **Next** and in the Detailed Options section, keep the default configuration.
11. Click **OK**.
    The reverse engineering process starts. Once the process is complete, the Right Model

is set to the one that you reverse engineered.



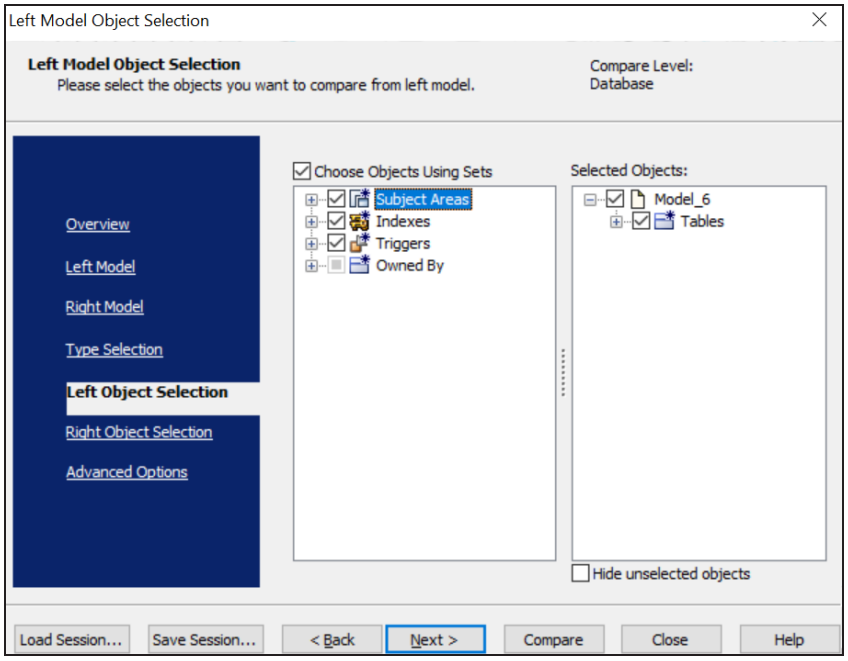
12. Click **Next** and in the Type Selection section, select the appropriate options.

    For example, the following image shows the default options.



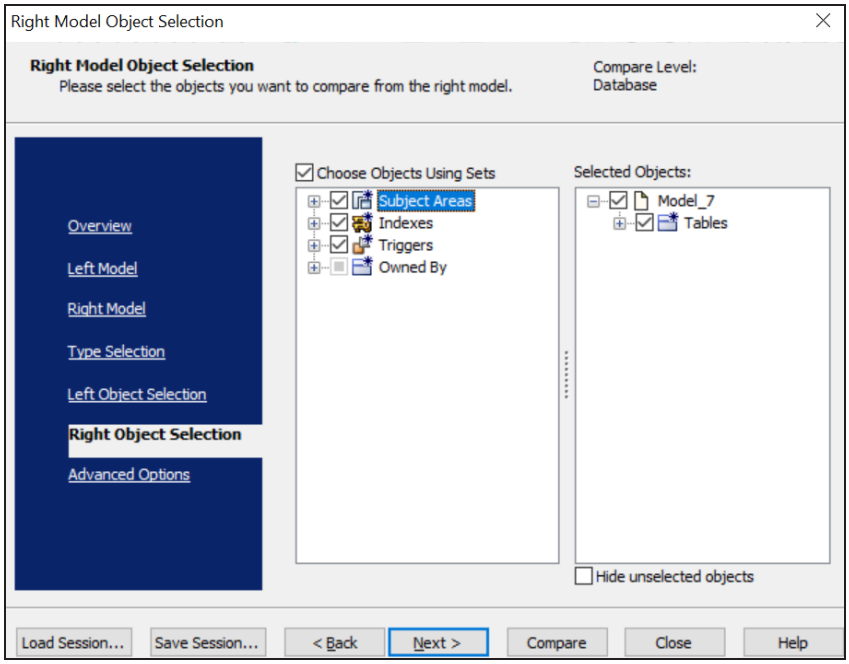
13. Click **Next** and in the Left Object Selection section, select the appropriate options.

    For example, the following image shows the default options.



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

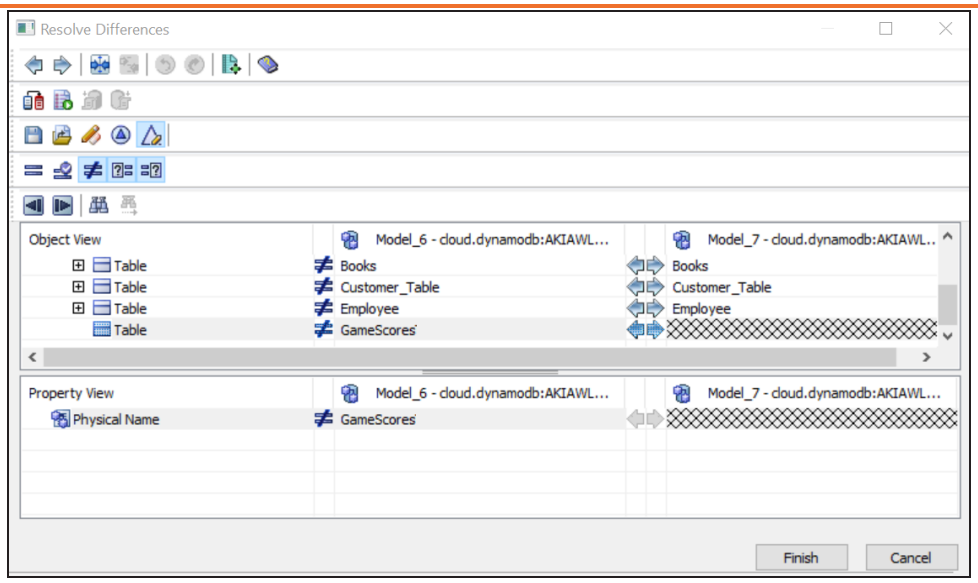
For example, the following image shows the default options.



15. Click **Compare**.

    The comparison process runs, and the Resolve Differences dialog box appears. It displays the differences between your model and database.

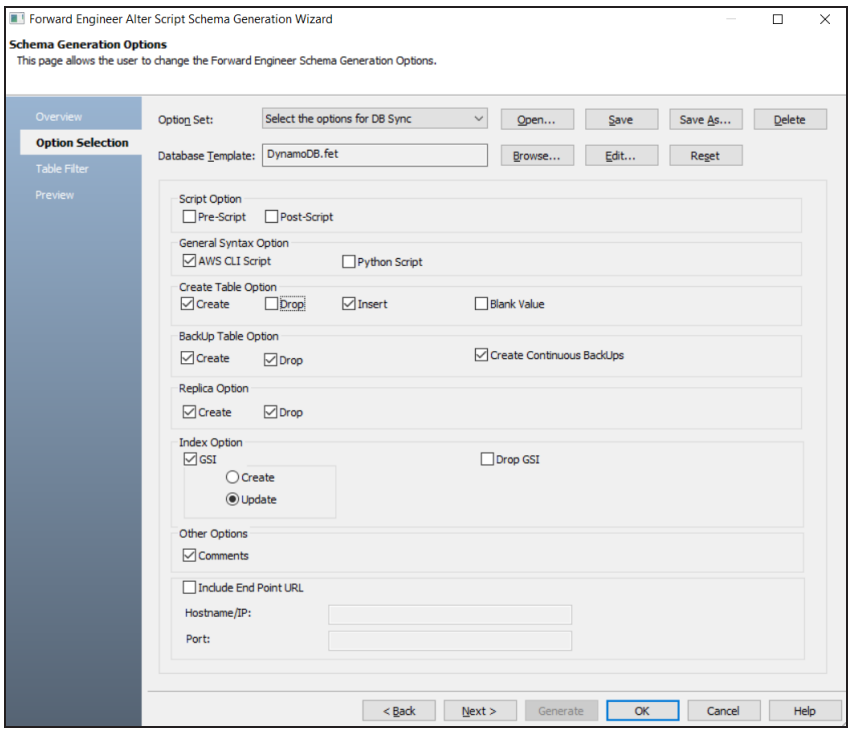
    For example, the following image shows that the GameScores table is available in your model but not in the database.



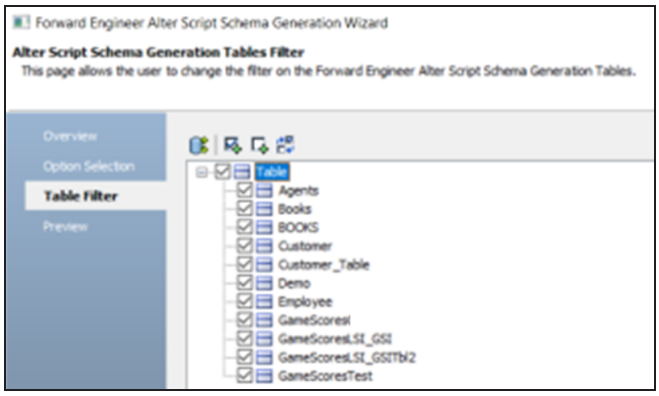
Select the missing table and click . This will move the GameScores table to the right model (from the database). Similarly, resolve other differences.

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

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



18. Click **Table Filter** and select or verify the tables to be included on the forward engineering script.



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

20. Click **Generate** and connect to your DynamoDB.
    The forward engineering process starts. The script generates your physical database

schema. You can access your database and verify the newly generated schema.

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

Migrating Relational Models to DynamoDB Models

You can convert and migrate your relational models to DynamoDB 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 DynamoDB model.

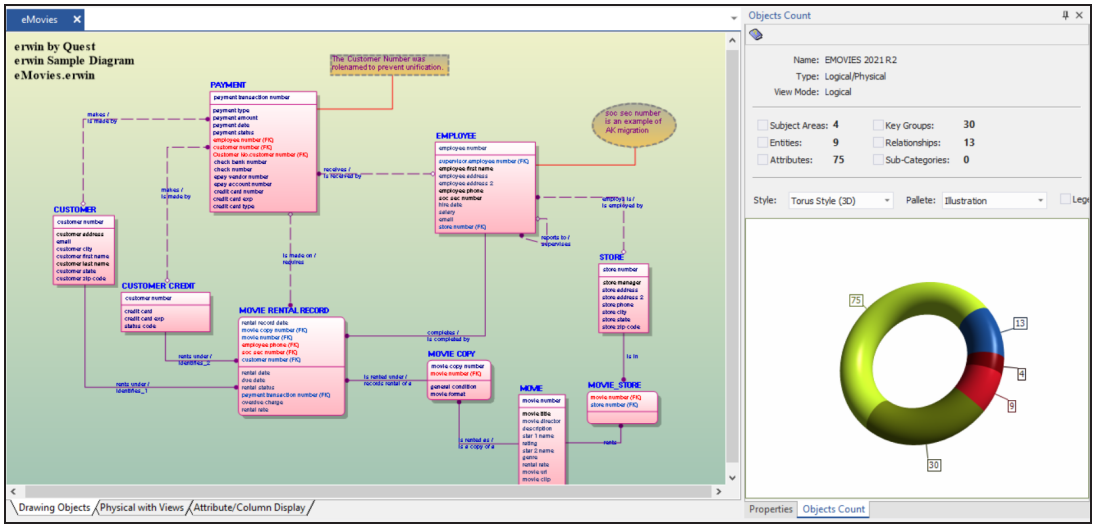
Migration by Changing the Target Database

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

1. Open your relational model.

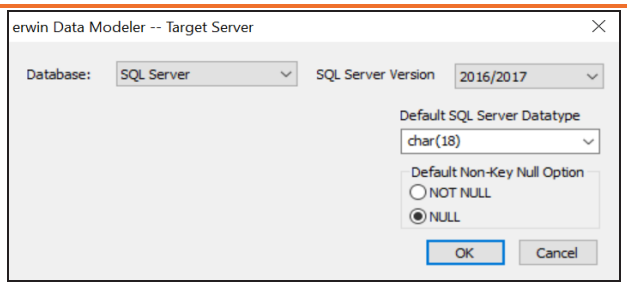
   Ensure that you are in Physical mode.

   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.

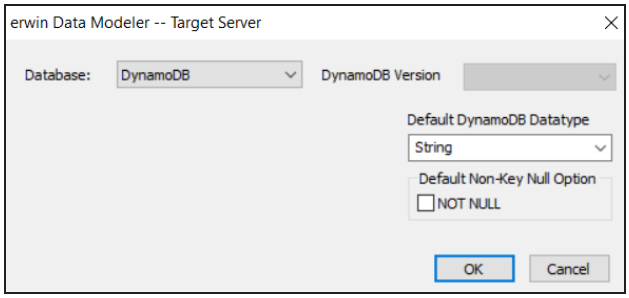


2. On the ribbon, click **Actions** > **Target Database** or on the status bar, click the database name.

   The erwin Data Modeler -- Target Server screen appears.



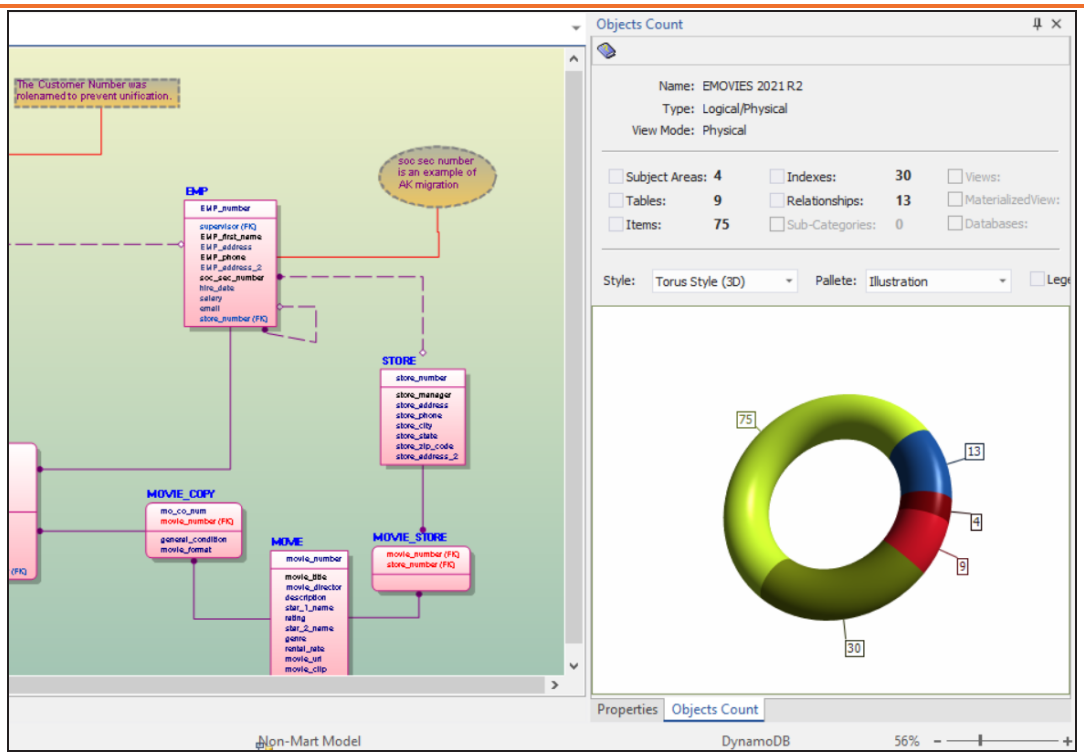
3. In the **Database** drop-down list, select DynamoDB.



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

> If the erwin Data Modeler dialog box appears, do one of the following:
>
> - Click **Yes** to view the report of unmapped datatypes.
> - Click **No** to skip this report.

Once the conversion is complete, the existing model is migrated to a DynamoDB model.



In the **Objects Count** pane, instead of columns we have items. The migration process converts and merges multiple tables, columns, and relationships to the NoSQL format according to the DynamoDB format.

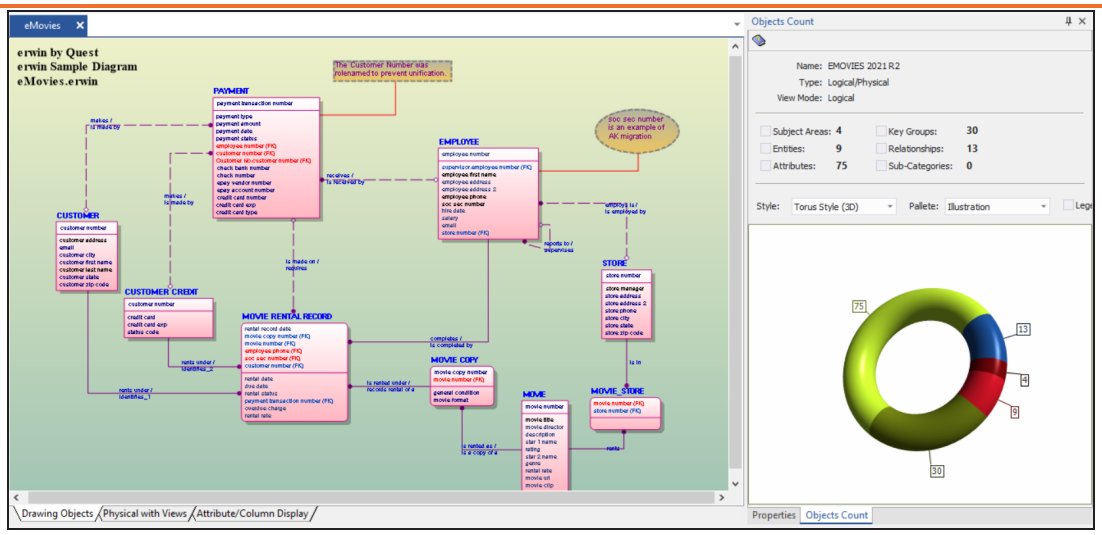
Migration by Deriving a Model

To migrate by deriving a model, follow these steps:

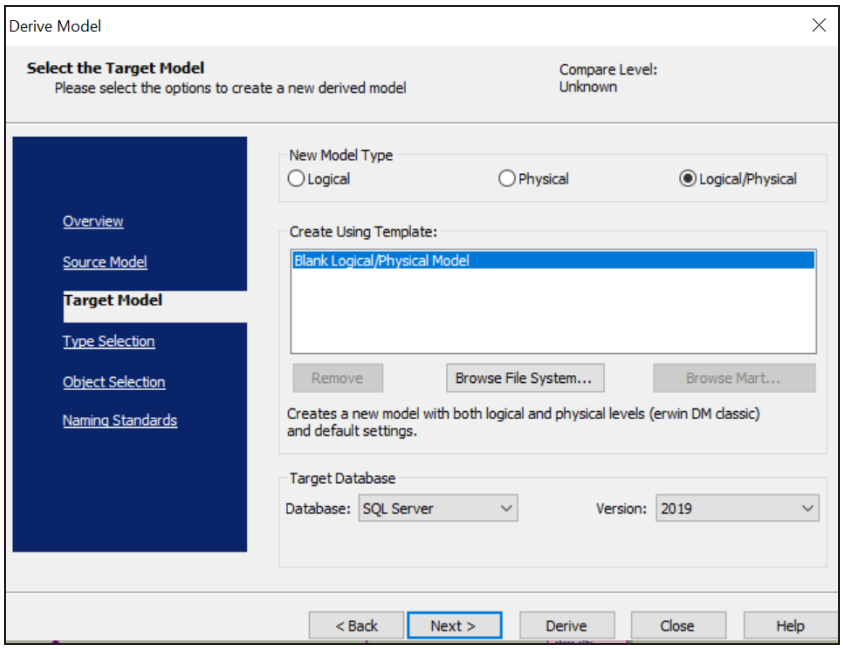
1. Open your relational model.

   Ensure that you are in Physical mode.

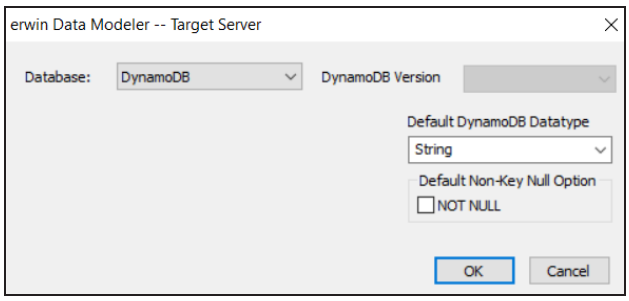
   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.



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.



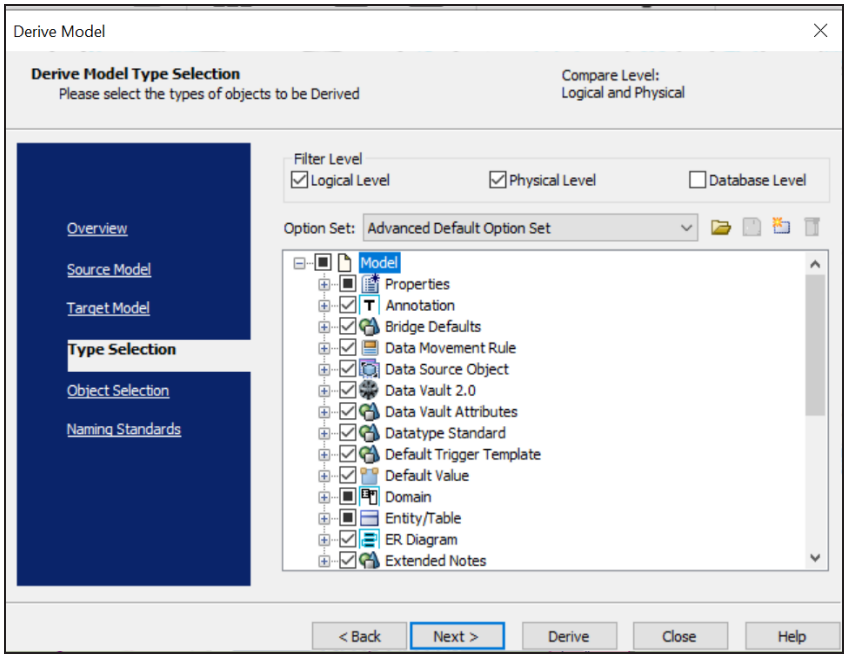
3. In the **Database** drop-down list, select **DynamoDB**.



4. Click **Next**.

> If the Type Resolution screen appears, click **Finish**.

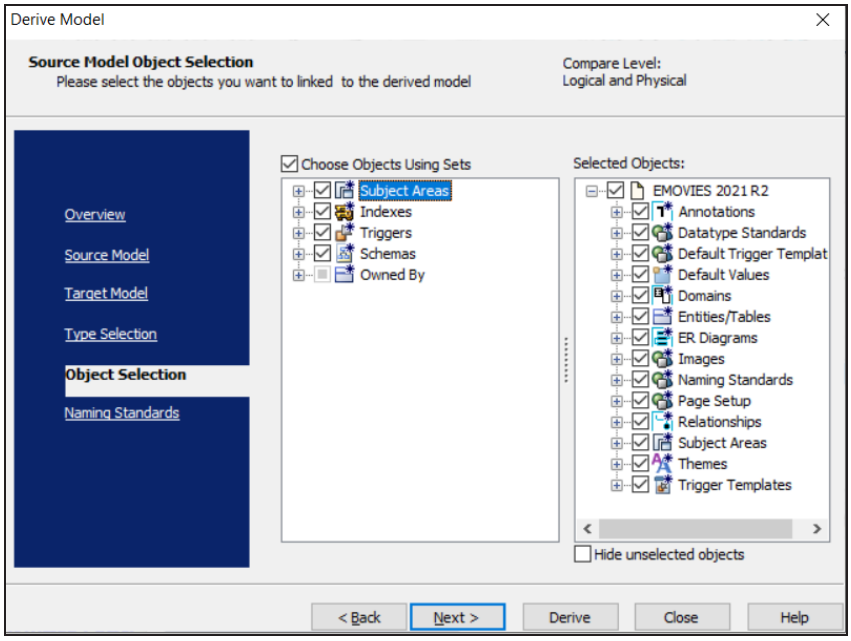
The Type Selection section appears.



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

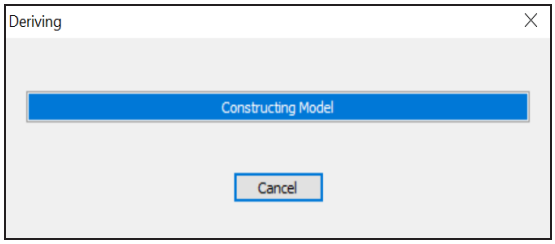
6. Click **Next**.
   The Object Selection section appears. Based on the object types you selected in step

5, it displays a list of objects.

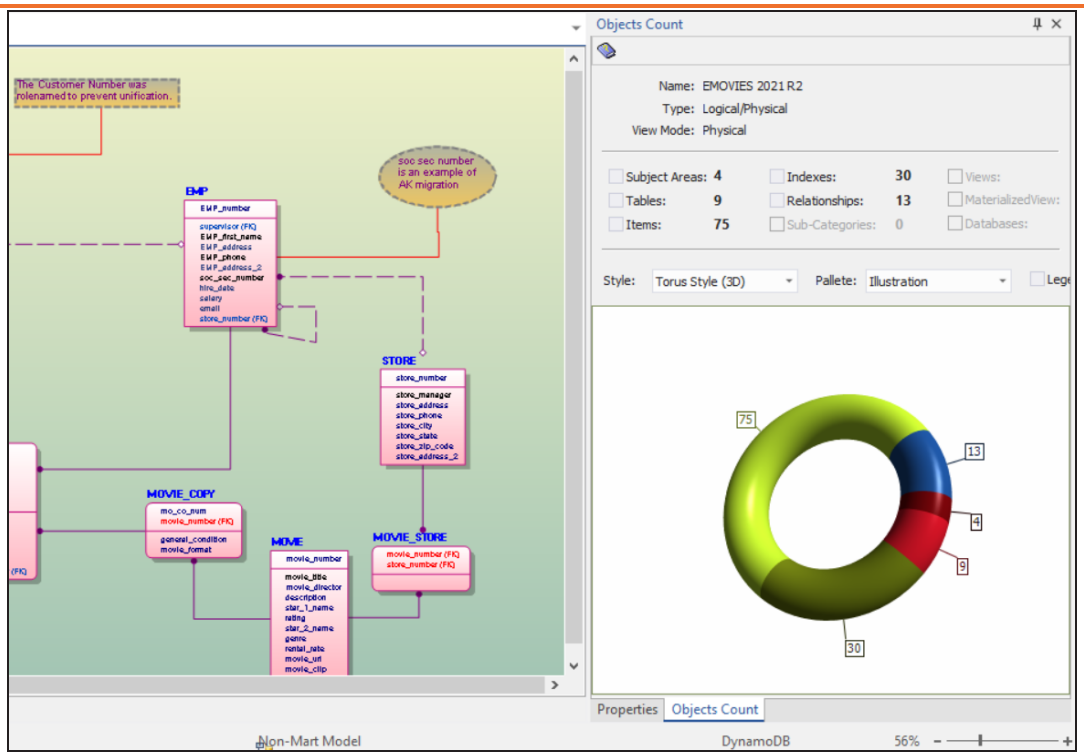


7. Select the objects that you want to derive into the target DynamoDB model.

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



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



In the **Objects Count** pane, note that instead of columns, we now have items. The migration process converts and merges multiple tables, columns, and relationships to NoSQL format according to the DynamoDB format.

Neo4j Support

erwin Data Modeler (DM) now supports Neo4j 4.2.x and 4.3.x as a target database. This implementation supports the following objects:

- Database
- Global Constraint
- Global Index
- Label
- Node
  - Field
  - Index
- Relationship
- Role
- User ID

Neo4j follows the Database > Label > Node hierarchy. A database can contain multiple labels, each with one or more nodes. A node represents data or information and a label groups the information in nodes together. Each node can have multiple properties (key-value pairs) that represent data.

Nodes can have one or more relationships between them. These relationships describe the connection between source and target nodes. Relationships are always specified with a direction using the "->" notation.

erwin DM focuses on the schema rather than data. Hence, the reverse engineering process retrieves the schema and forward engineering generates the schema; instead of data.

Following are the supported data types:

- ARRAY
- BOOLEAN
- DATE

- DURATION
- FLOAT
- INTEGER
- POINT
- STRING
- DATETIME
- LOCALDATETIME
- LOCALTIME
- TIME

Neo4j implementation supports all erwin DM features and functions. The following sections walk you through these features:

- Reverse engineering models from database and script
- Forward engineering models to database
- Comparing changes using Complete Compare
- Converting relational models to Neo4j models

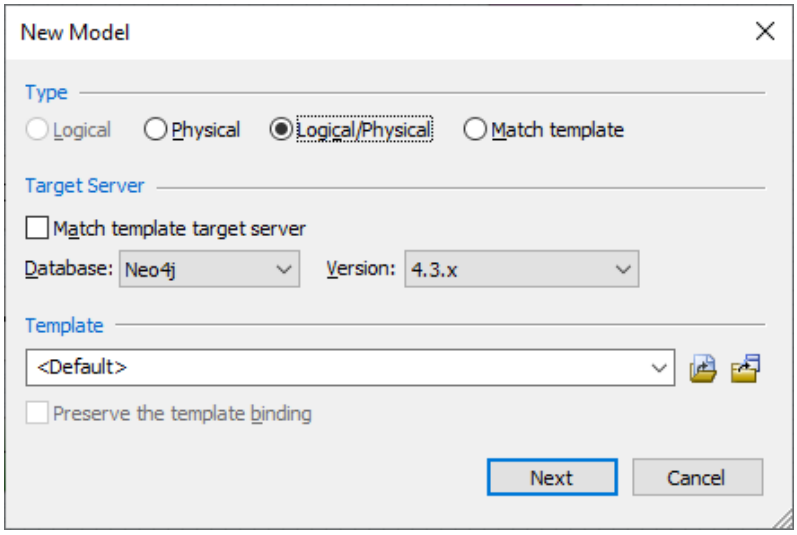
Reverse Engineering Models

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 Neo4j model. While reverse engineering erwin Data Modeler focuses on schema generation rather than data or information.

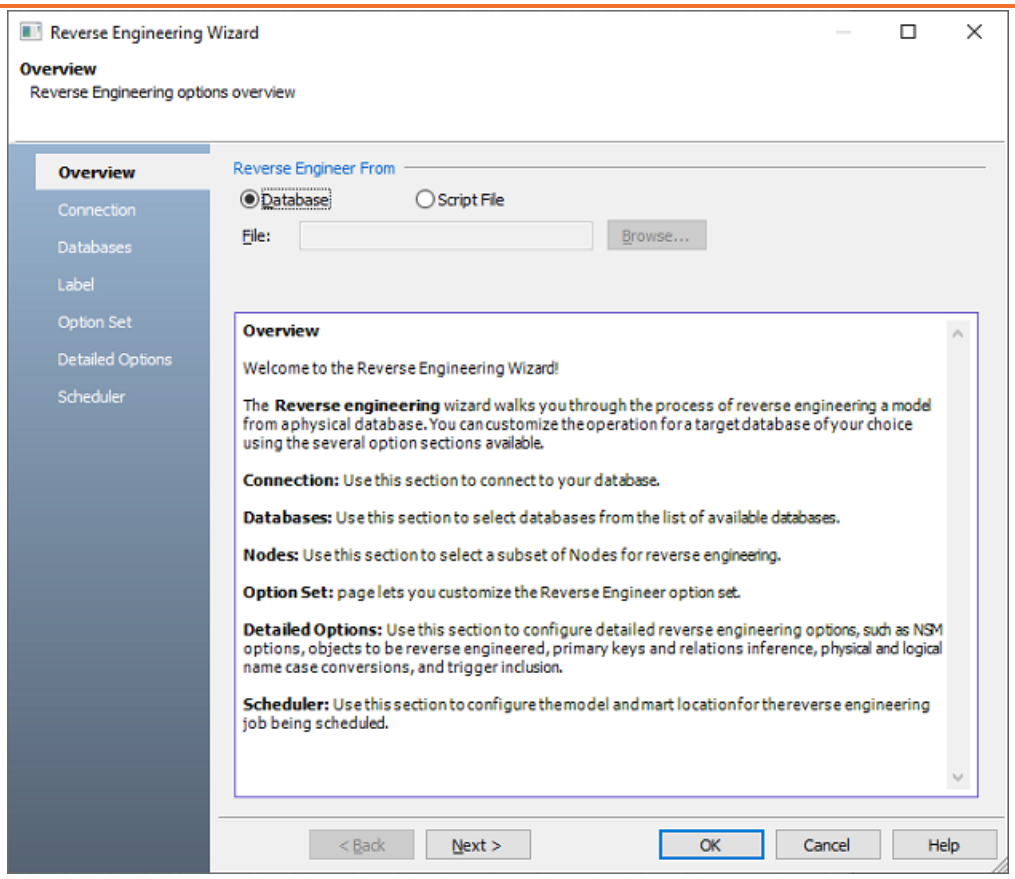
For detailed description of reverse engineering options, refer to the Reverse Engineering Options topic.

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 Neo4j.



3. Click **Next**.
   The Reverse Engineering Wizard appears.



4. Click one of the following options:
   - **Database**: Use this option to reverse engineer a model from your database.

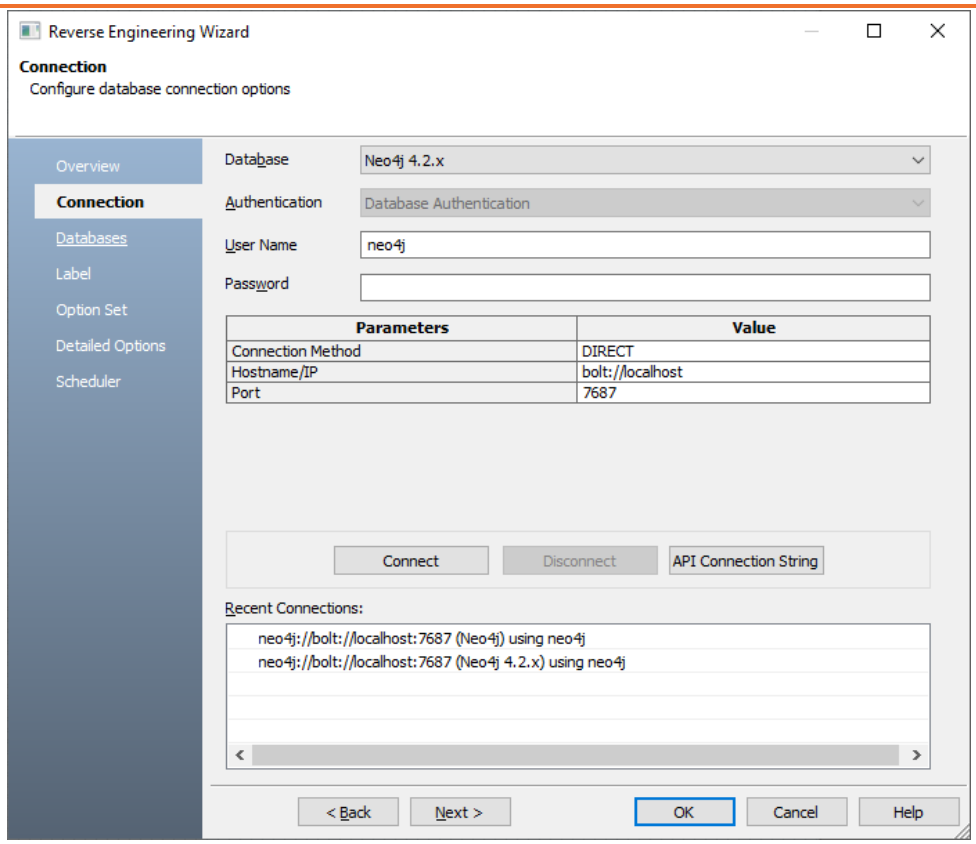
     If you click **Database**, continue to step 5.

   - **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.

     If you click **Script File**, see step 13 below.

5. Click **Next**.

   The Connection tab appears.



6. Enter your **User Name** and **Password**.

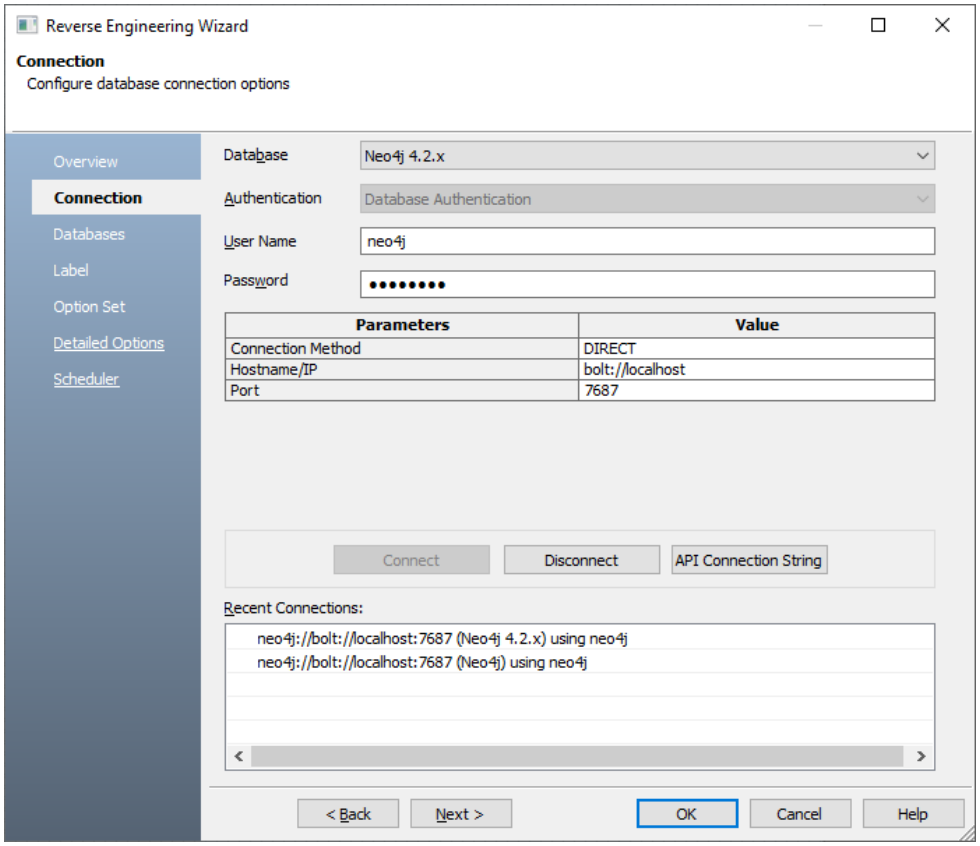
   The following table explains the connection parameters:

Parameter	Description	Additional Information
Connection Method	Specifies the type of connection you want to use. Using **Direct** connects to your database directly.	
Hostname/IP	Specifies the hostname or IP address of the server where your database is hosted in one of the following formats: • *Neo4j://<IP address>* • *Neo4j+s://<IP address>*	For example: • *Neo4j://localhost* • *Neo4j+s://localhost* • *bolt://localhost*

	• *bolt://<IP address>*	
Port	Specifies the port for your database based on your Hostname/IP mechanism	Default port number is 7687.

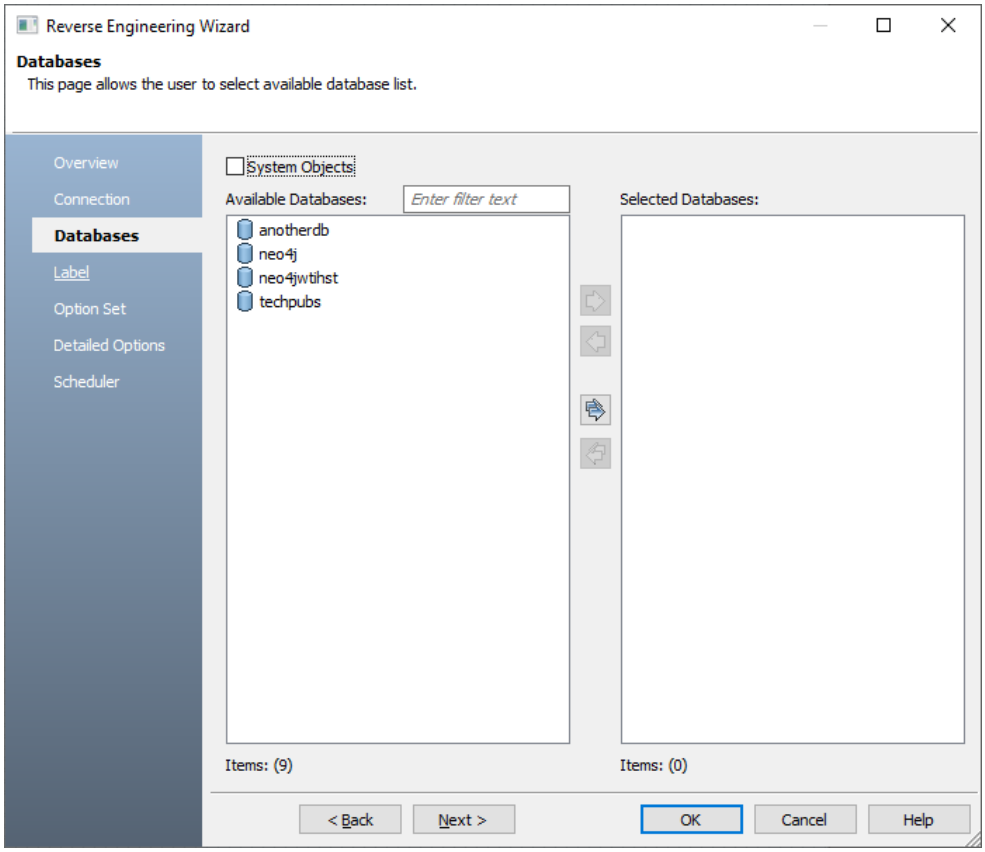
Other than the above Hostname/IP formats, Neo4j supports the Bolt+s://<IP address> format. However, erwin Data Modeler does not support it at the moment.

7. Then, click **Connect**.
   On successful connection, your connection information is displayed under Recent Connections.

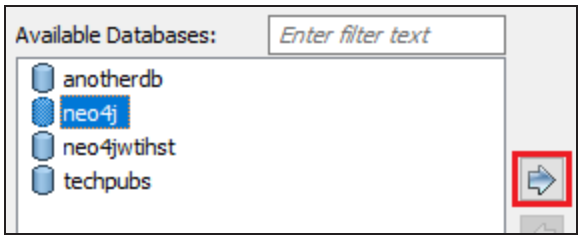


8. Click **Next**.

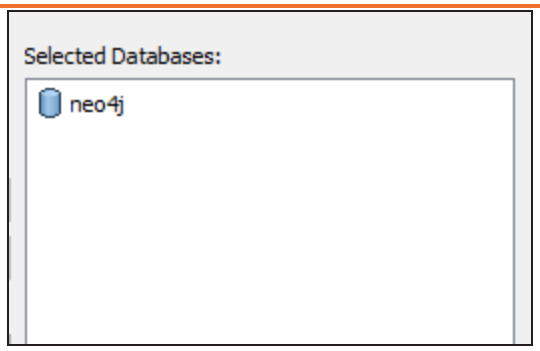
The Databases tab appears. It displays a list of available databases.



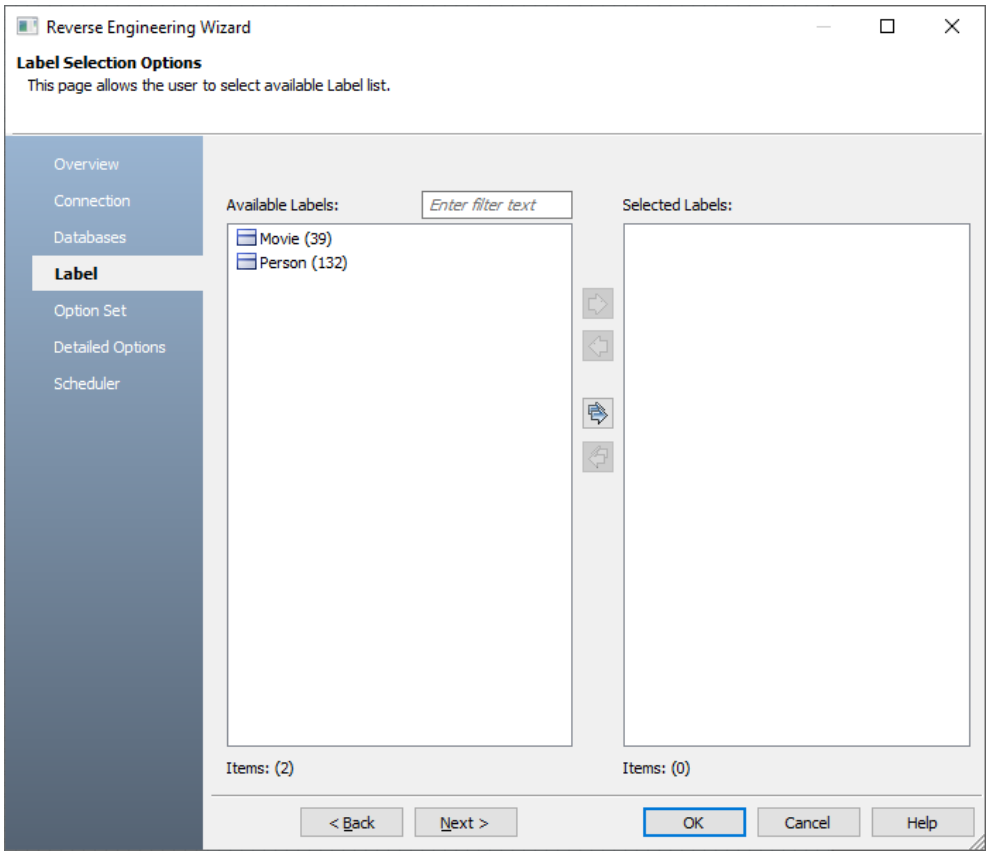
9. Under **Available Databases**, select the databases that you want to reverse engineer. Then, click  .



This moves the selected database under Selected Databases.

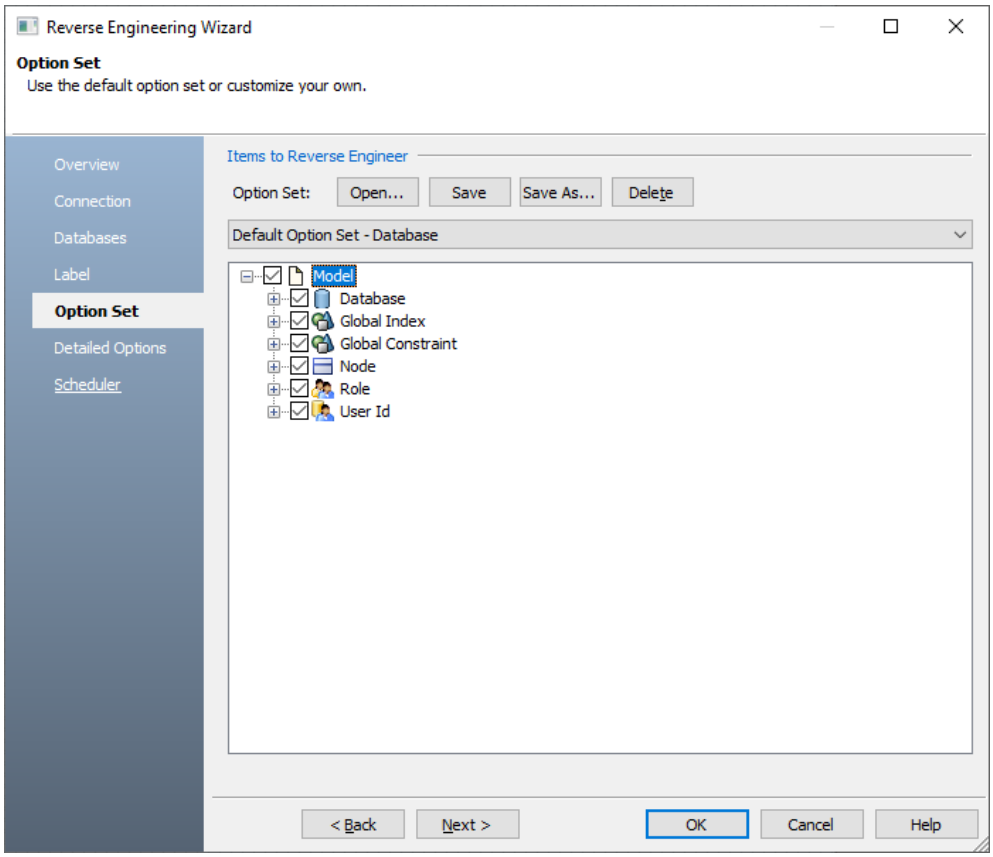


10. Click **Next**.
    The Label tab appears. It displays a list of available labels in the databases that you selected in step 9.



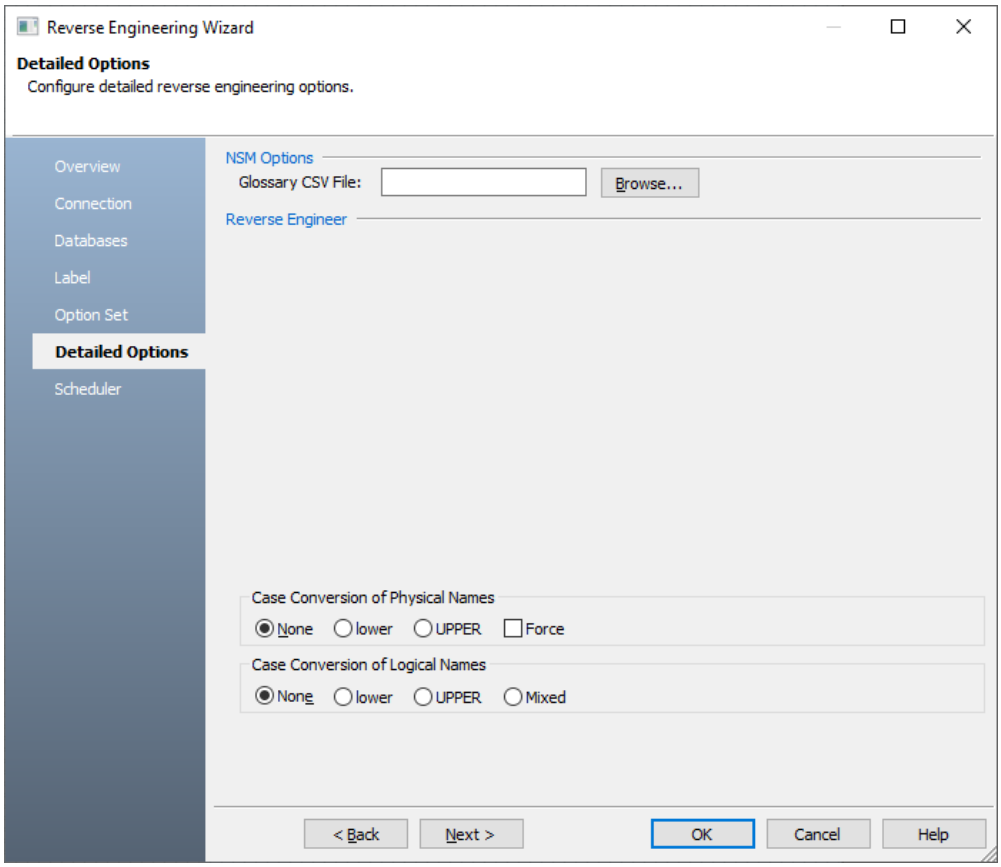
11. Under **Available Labels**, select the labels that you want to reverse engineer. Then, click  .

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



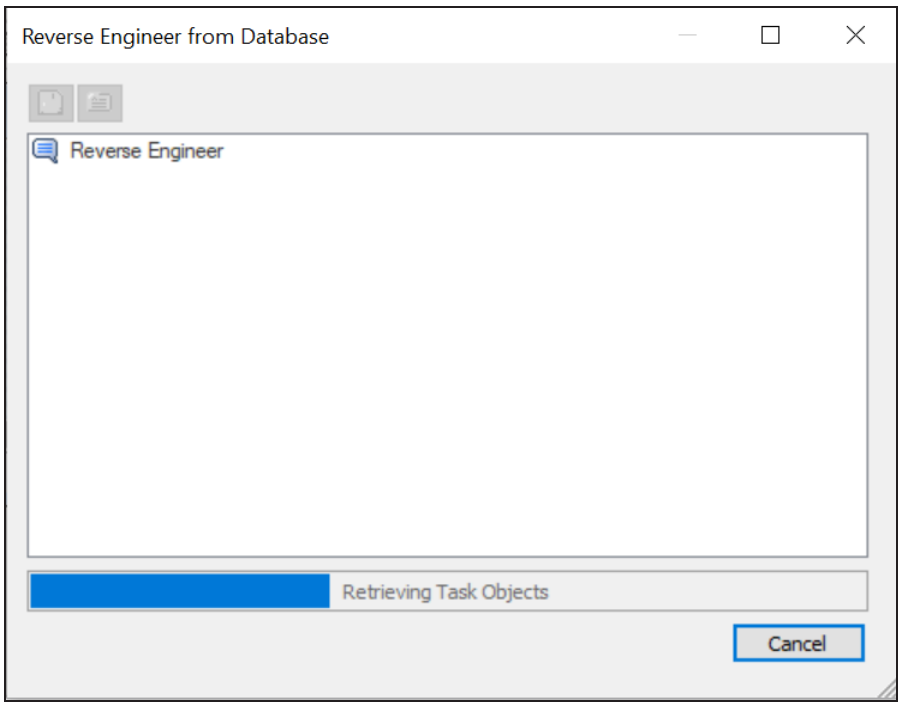
13. Click **Next**.
    The Detailed Options tab appears. Set up appropriate options based on your

requirement.

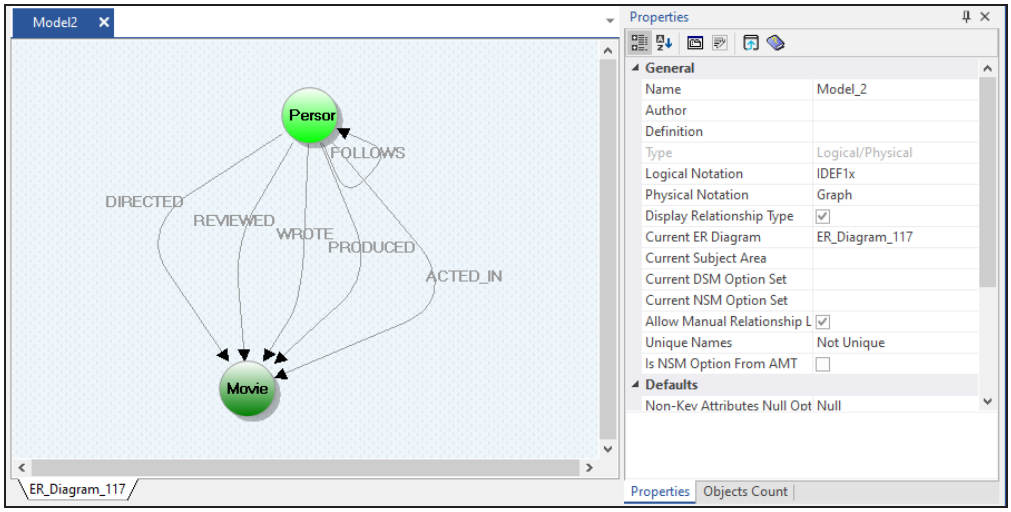


14. Click **OK**.

The reverse engineering process starts.

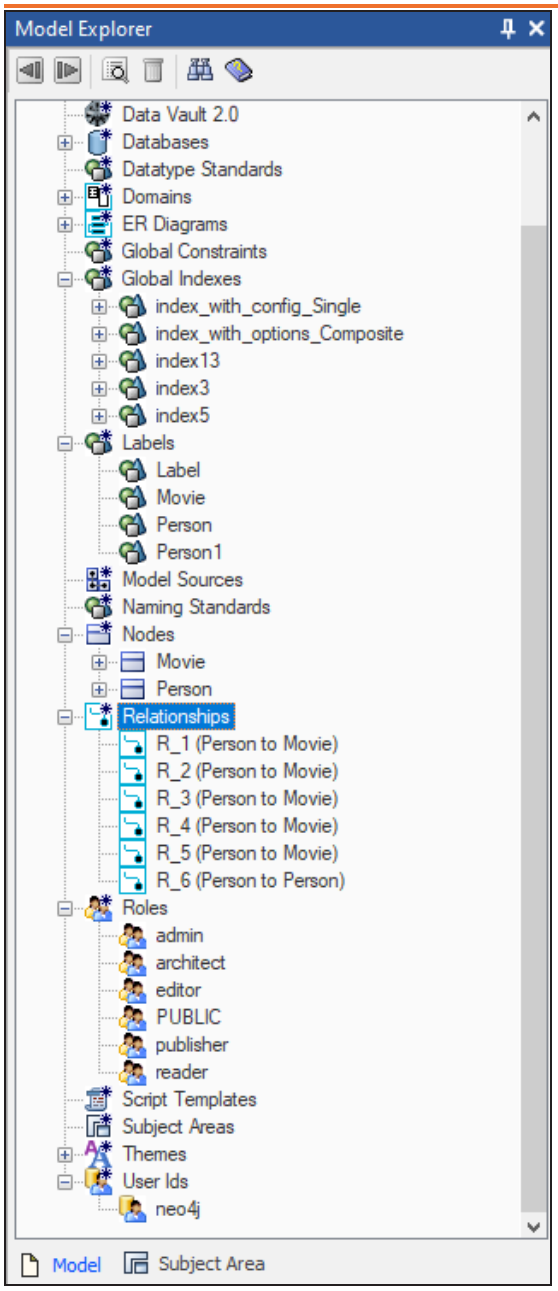


Once the process is complete, based on your selections, a schema is generated and a model is created.



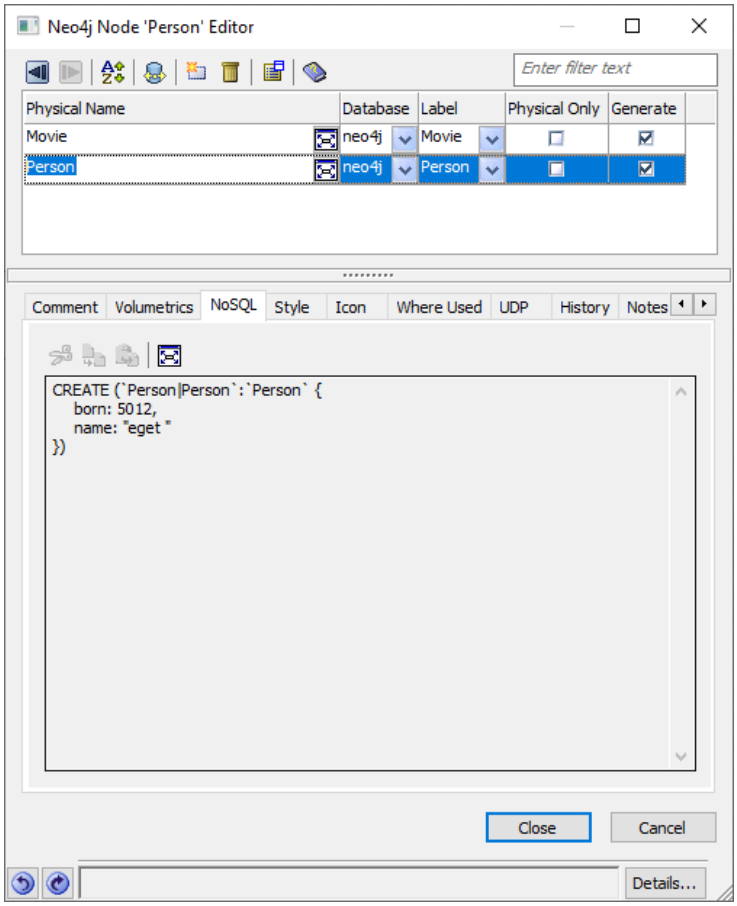
You can edit the shape of the nodes to look like the standard table-like structure. To change the node shape, on the ribbon, click **View** > **Field**. You can also change the label color, size, and caption using the Properties pane.

Along with Database, Labels, and Nodes, other objects, such as Global Indexes, Global Constraints, Users, and Roles are retrieved.



You can view these objects via the model diagram or view their properties via the Model Explorer. Right-click an object and then, click the required Properties option. For example, on the model diagram, right click a node and then, click **Node Properties**. The Neo4j Node

Editor appears. You can view the node's CREATE statement on the NoSQL tab. As seen, the node, Person has two properties, born and name to store additional information.



Reverse Engineering Options for Neo4j

Following are the reverse engineering options for Neo4j.

Overview

Parameter	Description	Additional Information
Reverse Engineer From	Specifies whether you want to reverse engineer from a script or database	**Database**: Indicates that the model is reverse engineered from database **Script File**: Indicates that the model is reverse engineered from a script
File	Specifies the script file's location	This option is available only when the Script File option is selected.

Connection

Parameter	Description	Additional Information
Connection Method	Specifies the type of connection you want to use. Using **Direct** connects to your database directly.	
Hostname/IP	Specifies the hostname or IP address of the server where your database is hosted in one of the following formats: • *Neo4j://<IP address>* • *Neo4j+s://<IP address>* • *bolt://<IP address>*	For example: • *Neo4j://localhost* • *Neo4j+s://localhost* • *bolt://localhost*
Port	Specifies the port for your database based on your Hostname/IP mechanism	Default port number is 7687.

Other than the above Hostname/IP formats, Neo4j supports the Bolt+s://<IP address> format. However, erwin Data Modeler does not support it at the moment.

Databases

Parameter	Description	Additional Information
System Objects	Specifies whether system databases are included under the Available Databases	
Available Databases	Specifies a list of available databases	
Selected Databases	Specifies a list of selected databases for reverse engineering	

Label

Parameter	Description	Additional Information
Available Labels	Specifies a list of available labels in the selected databases	
Selected Labels	Specifies a list of selected labels for reverse engineering	

Option Set

Parameter	Description	Additional Information
Option Set	Specifies the option set template for reverse engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save the configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at some external location. **Delete**: Use this option to delete an option set.
<Option Set Name>	Specifies the objects to be reverse engineered according to the selected	

	option set. You can edit this list.	

Detailed Options

Parameter	Description	Additional Information
NSM Options	Specifies the naming standard glossary file in the .CSV format	
Case Conversion of Physical Names	Specifies how the case conversion of physical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Force**: Indicates whether the physical name property of all the logical/physical models is overridden. If this option is enabled, the logical/physical link is broken between the logical and physical name. If this option is not enabled, all logical and physical names are set to the same value after the process completes.
Case Conversion of Logical Names	Specifies how the case conversion of logical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Mixed**: Indicates that the mixed-case logical names are preserved

Scheduler

The options on this tab are available only while reverse engineering via erwin DM Scheduler.

Parameter	Description	Additional Information
Model	Specifies the location and name of the reverse engin-	For example: C:\Scheduler\<Model Name>.erwin

	eered model	When you schedule a job on a remote server, ensure the model path is same for remote and local server.
Mart Folder	Specifies the location or library in your mart where the reverse engineered model is saved	To use this option, ensure that you are connected to a mart. For more information, refer to the Connecting to Mart topic.
Complete Compare	Specifies whether the Complete Compare (CC) process should run while reverse engineering	
Output File	Specifies the location of the CC output file generated	
File	Specifies that the target model location is on the local system	
Mart	Specifies that the target model location is in the mart	
Using Latest Version	Specifies whether the target model is the latest version of the model in the mart	This option is available only when Mart is selected.
Save To Mart	Specifies whether the reverse engineered model is saved to the mart	This option is available only when Using Latest Version is selected.
Target Model	Specifies the location of the target model for CC	
Option Set	Specifies the option set that is used for CC	**Advanced Default Option Set**: Indicates that all erwin DM metadata is included. CC works slowest with this option. **Speed Option Set**: Indicates that only the essential metadata is included. CC works the fastest with this option set. **Standard Default Option Set**: Indicates that

		standard metadata is included. CC works fast with this option set compared to the Advanced option set.

Forward Engineering Models

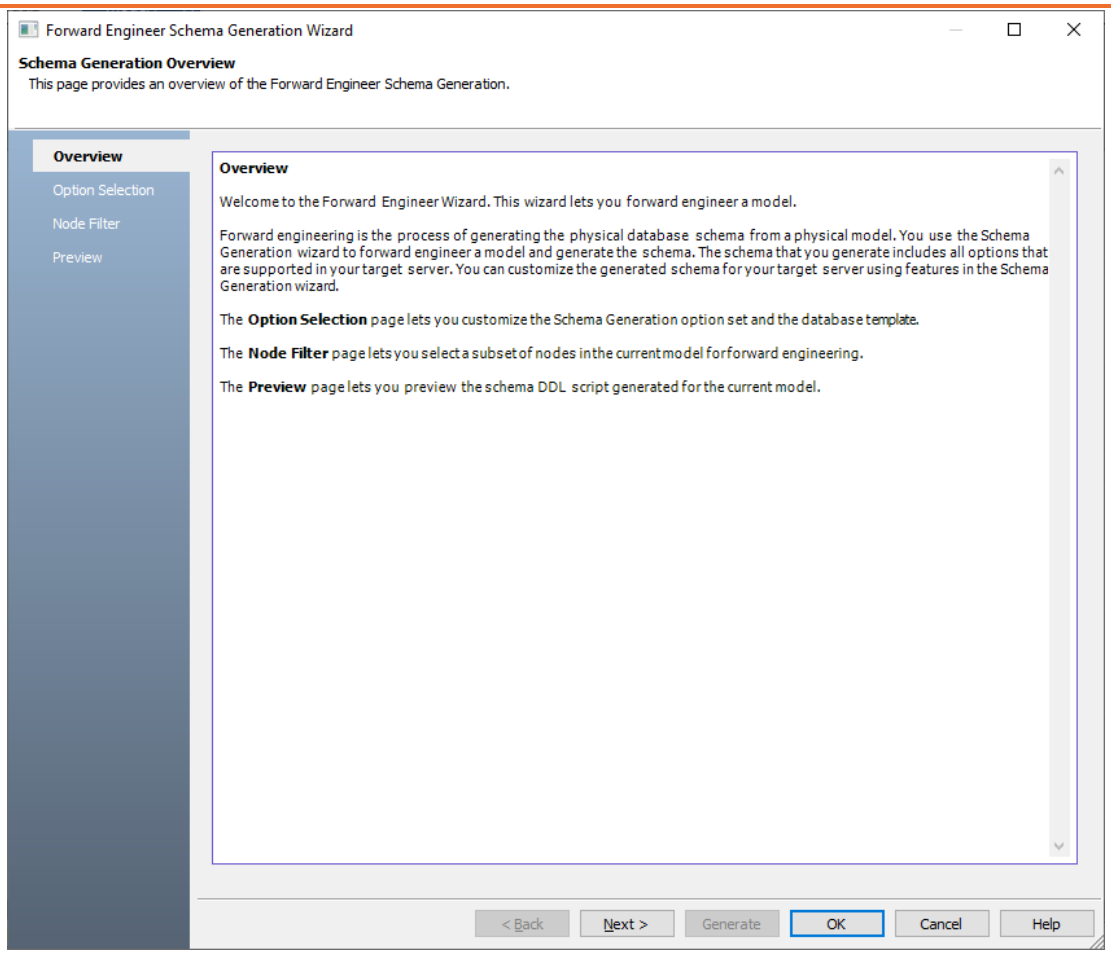
You can generate a physical database schema from a physical model using the Forward Engineering process. This topic walks you through the steps to forward engineer a Neo4j model. For detailed description of forward engineering options, refer to the Forward Engineering Options topic.

To forward engineer a model:

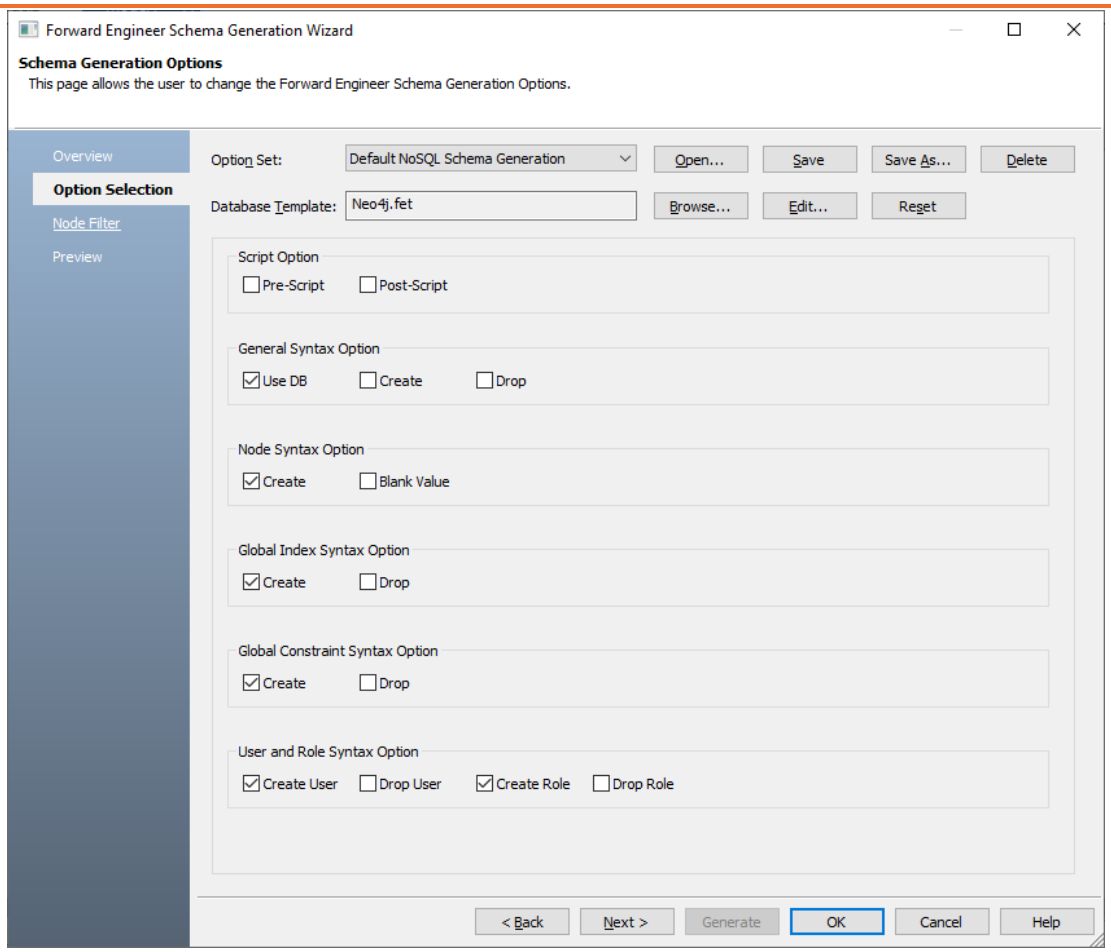
1. Open your Neo4j model.

   
   Ensure that you are in the Physical mode.

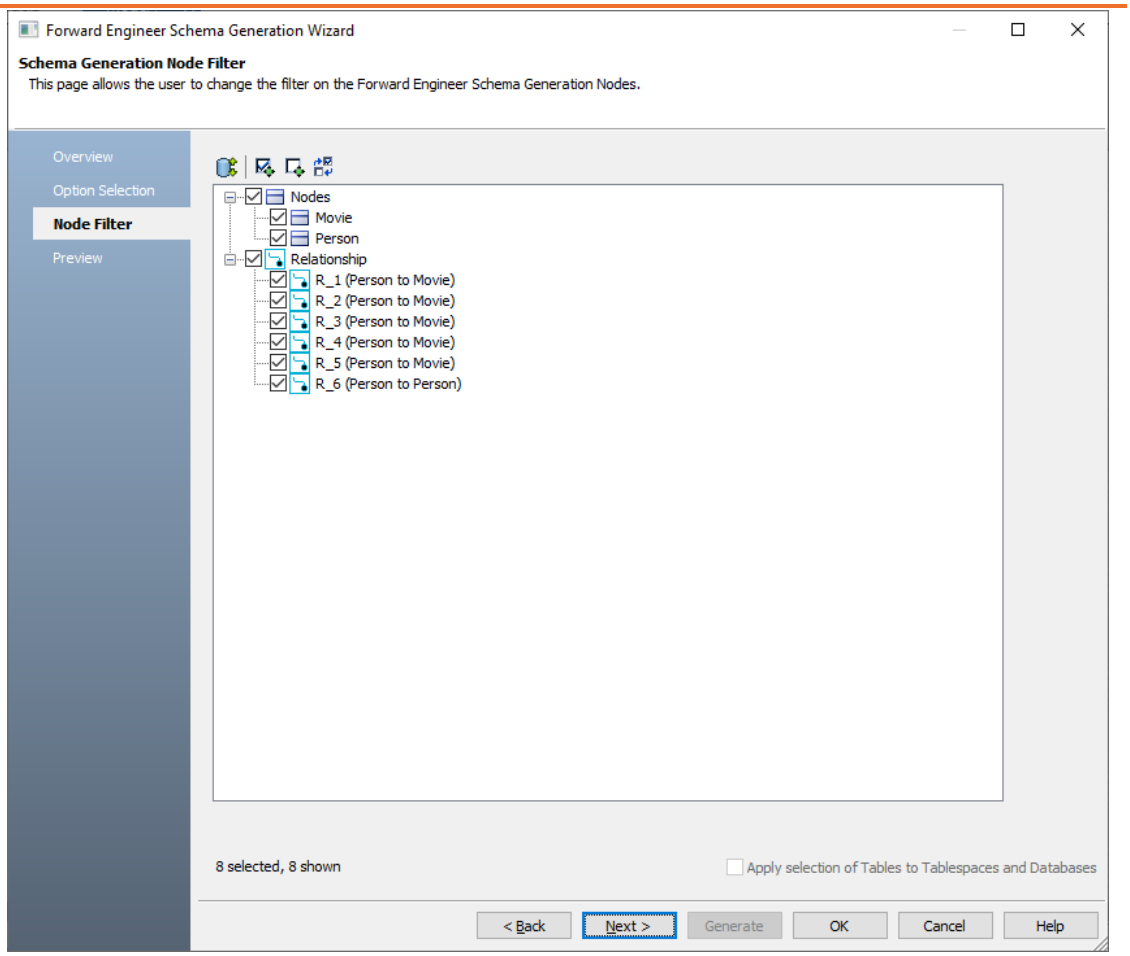
2. Click **Actions** > **Schema**.
   The Forward Engineer Schema Generation Wizard appears.



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

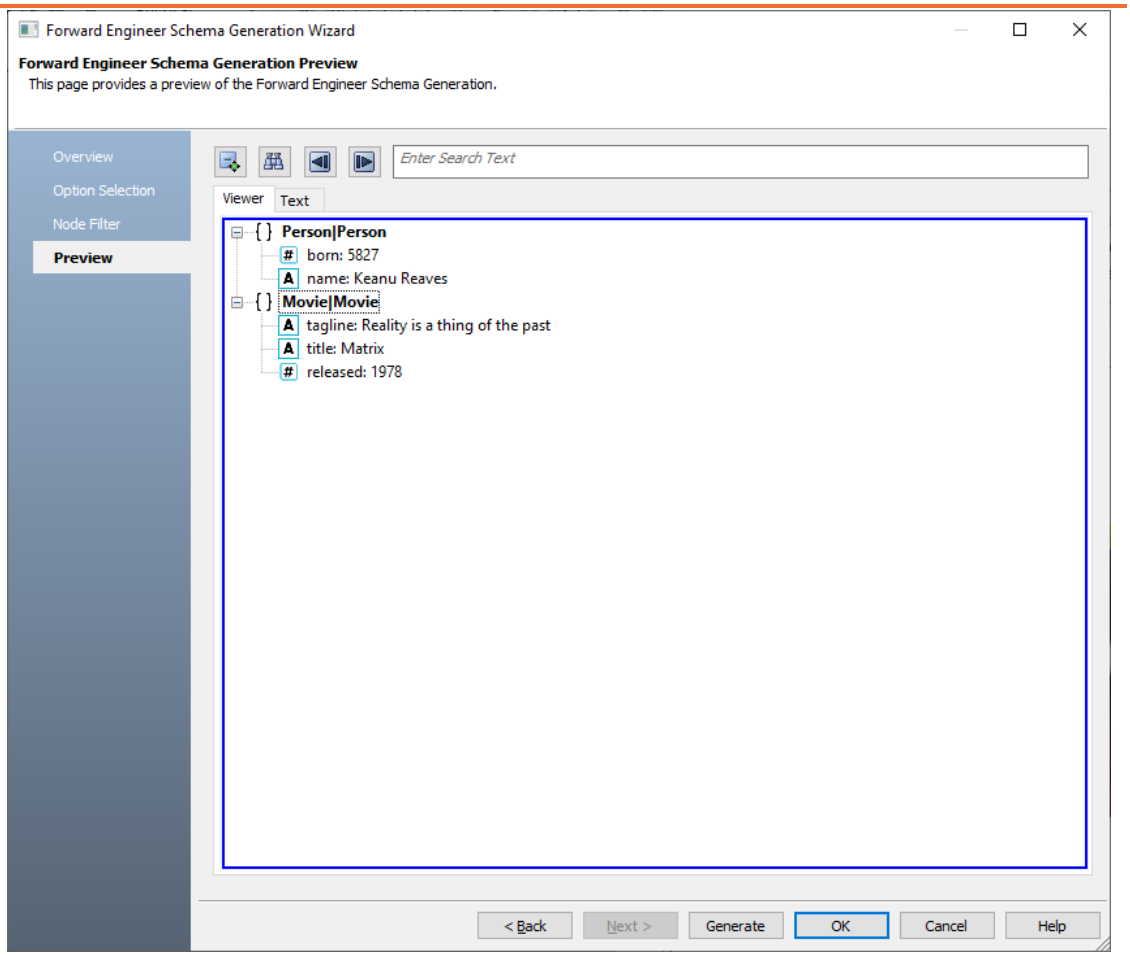


4. Click **Next**.
   The Node Filter tab appears. It displays a list of nodes available in your model.

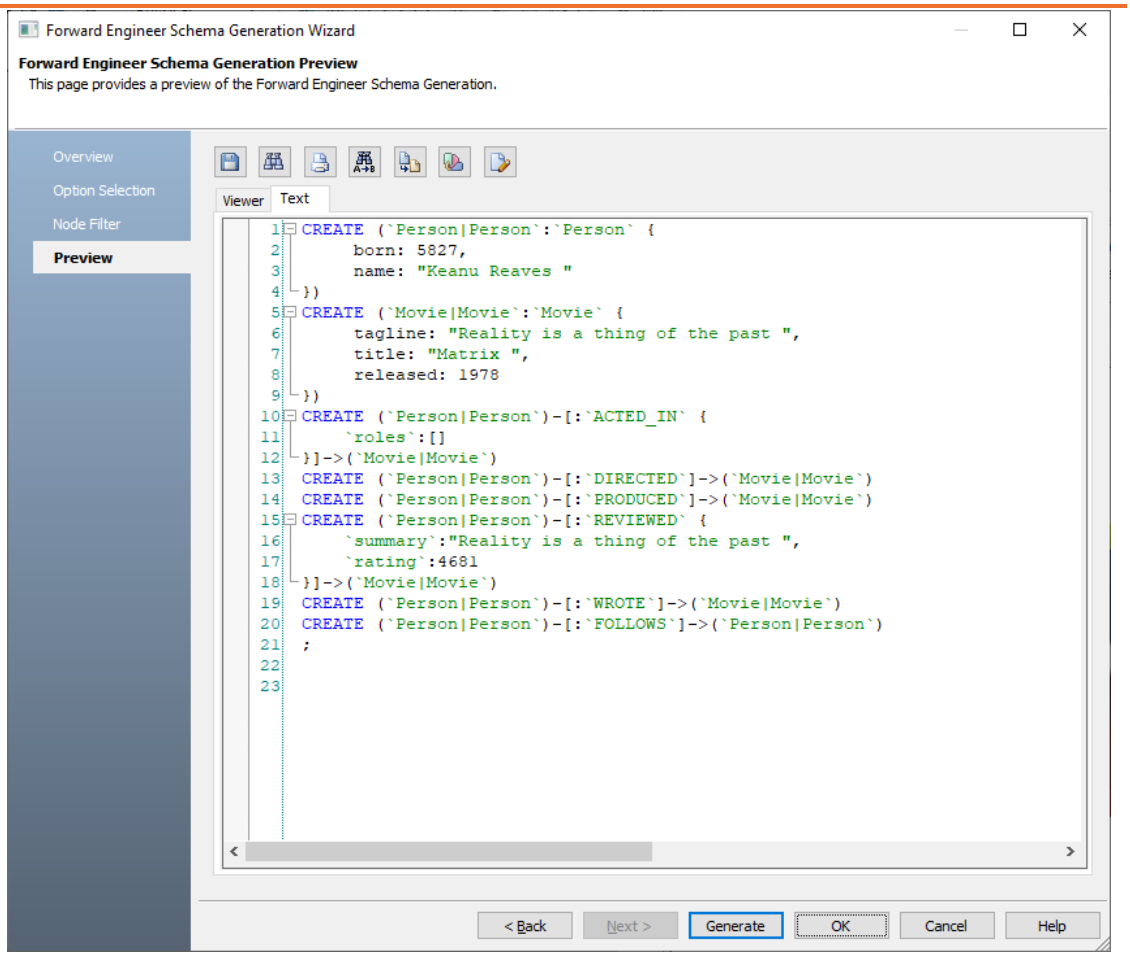


5. Select the nodes that you want to forward engineer.
6. Click **Preview** to view the schema and its script.

   The schema is available on the Viewer tab.



The script is available on the Text tab.

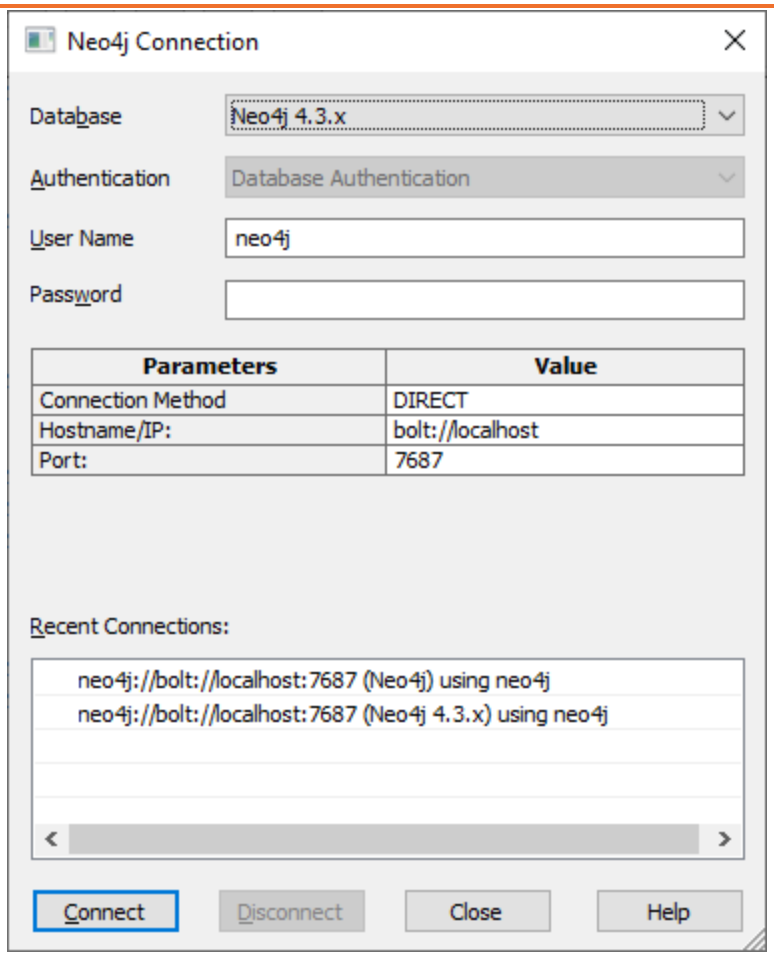


Use the following options:

- **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 Cypher format.

7. Click **Generate**.

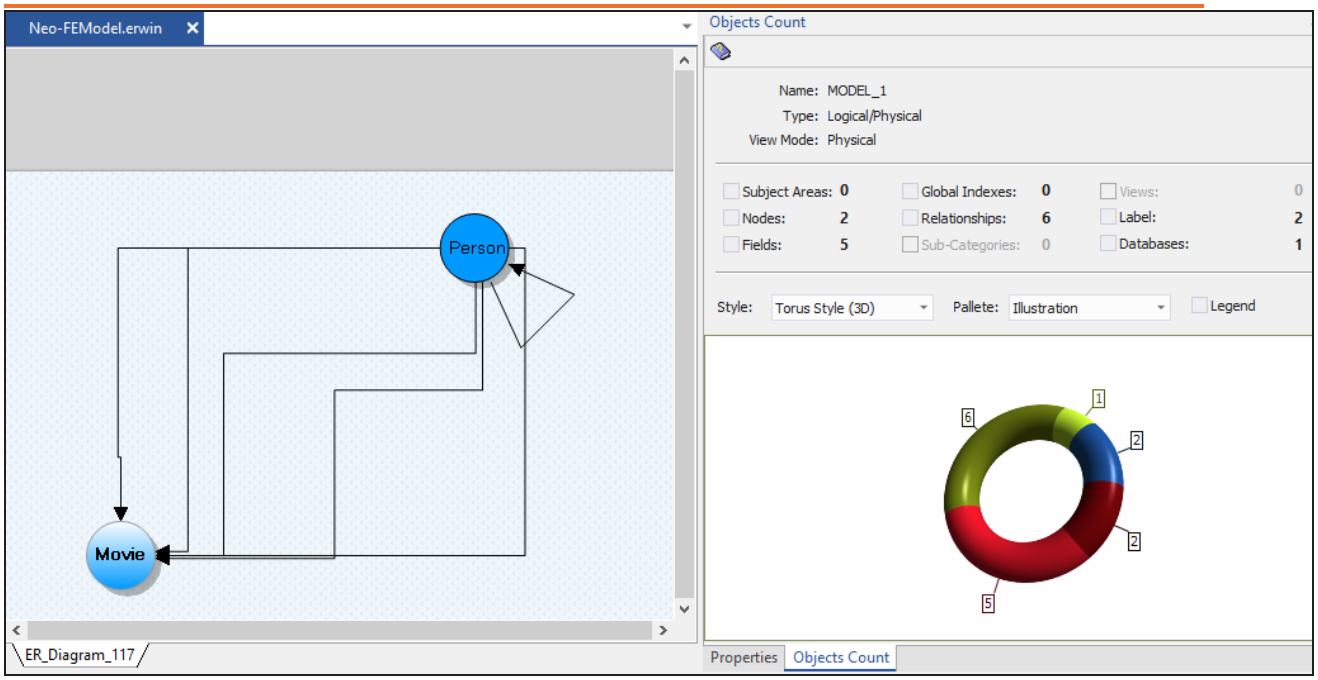
   The Neo4j Connection page appears.



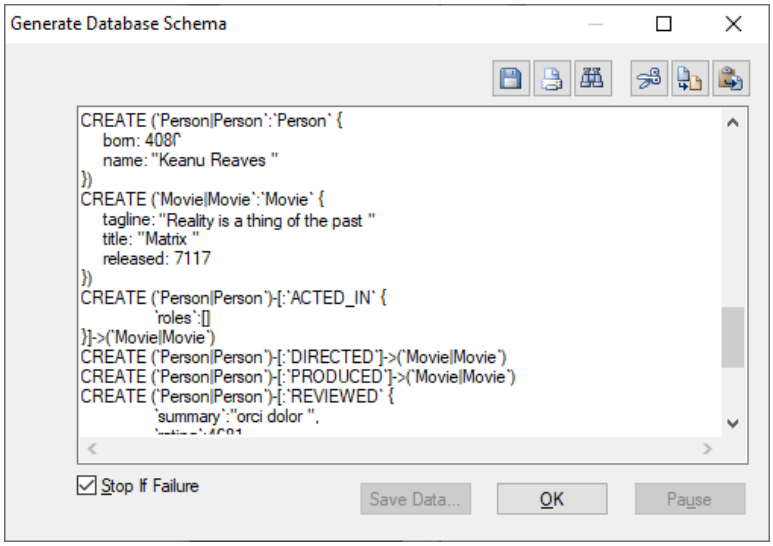
8. Enter User Name, Password, and appropriate connection parameters to connect the required database. Then, click **Connect**. For more information on connection parameters, refer to the connection parameters topic.

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

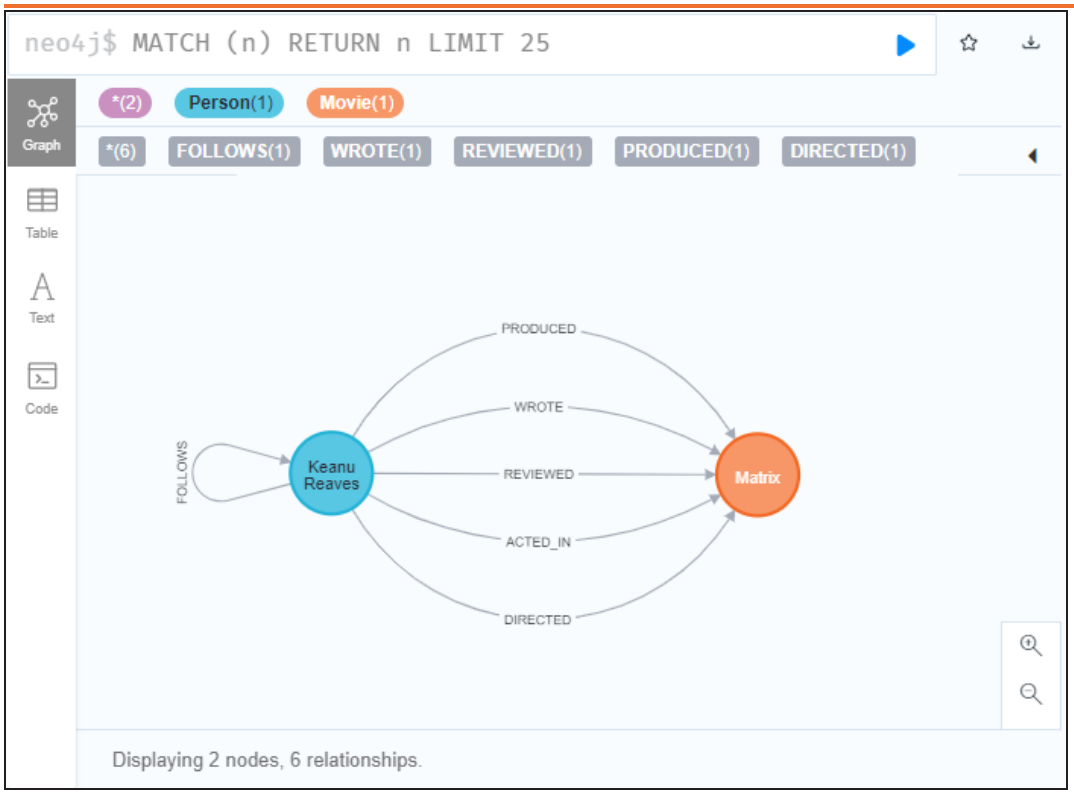
For example, the following model has two nodes with five fields and six relationships. Apart from this, it has two labels and one database.



On forward engineering, the following script was generated:



Based on the generated schema, the graph looks as follows in your database. It has two nodes, movie (Matrix) and person (Keanu Reaves), with six relationships.


neo4j$ MATCH (n) RETURN n LIMIT 25
Graph
Table
Text
Code
*(2)
Person(1)
Movie(1)
*(6)
FOLLOWS(1)
WROTE(1)
REVIEWED(1)
PRODUCED(1)
DIRECTED(1)
PRODUCED
WROTE
REVIEWED
ACTED_IN
DIRECTED
FOLLOWS
Keanu Reaves
Matrix
Displaying 2 nodes, 6 relationships.

Forward Engineering Options for Neo4j

Following are the forward engineering options for Neo4j.

Option Selection

Parameter	Description	Additional Information
Option Set	Specifies the option set template for forward engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save a configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at an external location. **Delete**: Use this option to delete an option set.
Database Template	Specifies the database template for controlling schema generation	**Browse**: Use this option to browse and select a database template. **Edit**: Use this option to edit a template in the Template Editor. **Reset**: Use this option to reset the Database Template option.
Script Option	Specifies the script option for schema generation	**Pre-Script**: Indicates whether pre-scripts attached to the schema are executed **Post-Script**: Indicates whether the post-scripts attached to the schema are executed
General Syntax Option	Specifies the general syntax options for schema generation	**Use DB**: Indicates whether the Use DB syntax for databases is executed **Create**: Indicates whether the Create syntax for databases is executed **Drop**: Indicates whether the Drop syntax for databases is executed
Node Syntax Option	Specifies the node syntax options for schema	**Create**: Indicates whether the Create syntax for nodes is executed

	generation	**Blank Value**: Indicates whether the Blank Value syntax for nodes is executed. Using this creates a syntax with blank node properties instead of random values.
Global Index Syntax Option	Specifies the global index syntax options for schema generation	**Create**: Indicates whether the Create syntax for global indexes is executed **Drop**: Indicates whether the Drop syntax for global indexes is executed
Global Constraint Syntax Option	Specifies the global constraint syntax options for schema generation	**Create**: Indicates whether the Create syntax for global constraints is executed **Drop**: Indicates whether the Drop syntax for global constraints is executed
User and Role Syntax Option	Specifies the user and role syntax options for schema generation	**Create User**: Indicates whether the Create syntax for users is executed **Drop User**: Indicates whether the Drop syntax for users is executed **Create Role**: Indicates whether the Create syntax for roles is executed **Drop Role**: Indicates whether the Drop syntax for roles is executed

Node Filter

Parameter	Description	Additional Information
Nodes	Specifies the selected nodes for schema generation	
Display either Logical Names or Physical Names		**Physical Names**: Indicates that only physical names of the nodes are included in the generated schema **Physical Names, show owner**: Indicates that physical names and owners of the nodes are included in the generated schema

		Physical Names, show owner using User: Indicates that the physical names and owners of the nodes are included in the generated schema. Owners of the nodes are displayed using User.
Select all of the items in the list	Use this option to select all the nodes in the list.	
Unselect all of the items in the list	Use this option to clear all the nodes.	
Select all unselected items, and unselect all selected items	Use this option to select all the unselected nodes and clear all the previously selected nodes.	

Preview

Parameter	Description	Additional Information
Viewer	Displays the schema in the viewer editor	**Collapse All**: Use this option to collapse all the nodes. **Search**: Use this option to search a text entered in the search box. **Find Previous**: Use this option to navigate to previous search string in the search results **Find Next**: Use this option to navigate to next search string in the search result.
Text	Displays the schema in the text editor	**Save**: Use this option to save the generated schema. **Search**: Use this option to search through the generated schema. **Print**: Use this option to print the generated schema. **Replace**: Use this option to find and replace text in the generated schema. **Copy**: Use this option to copy the selected text in the

		schema. **Text Options**: Use this option to edit window settings, fonts, syntax color. **Error Check**: Use this option to check errors in the forward engineering script. **Git**: Use this option to commit the FE script to a Git repository.

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.

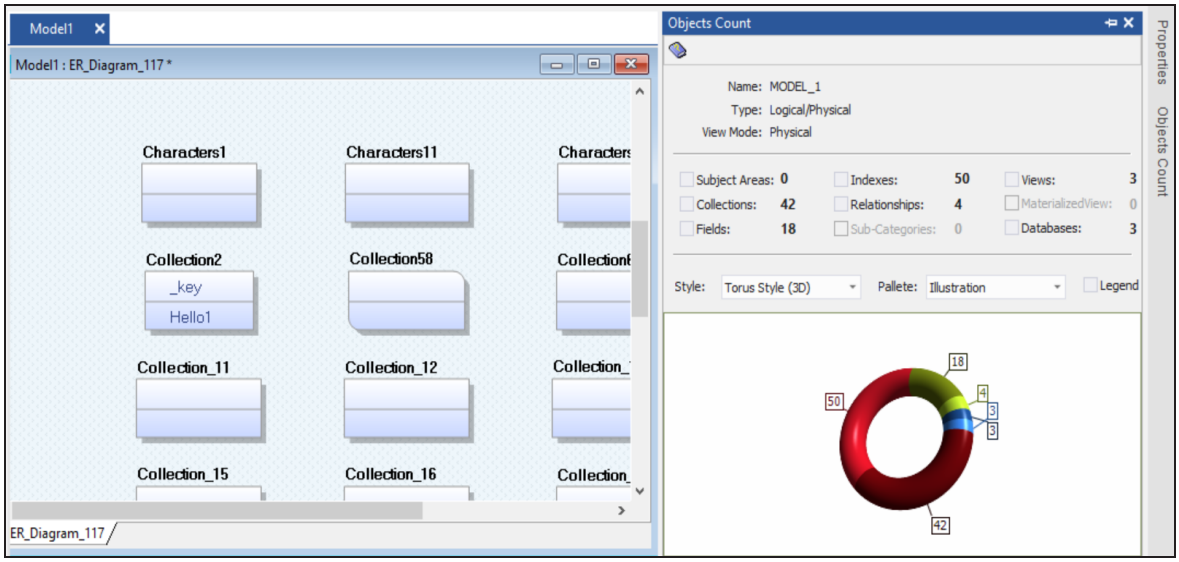
This topic walks you through the steps to compare an Neo4j model with database.

To compare models with database, follow these steps:

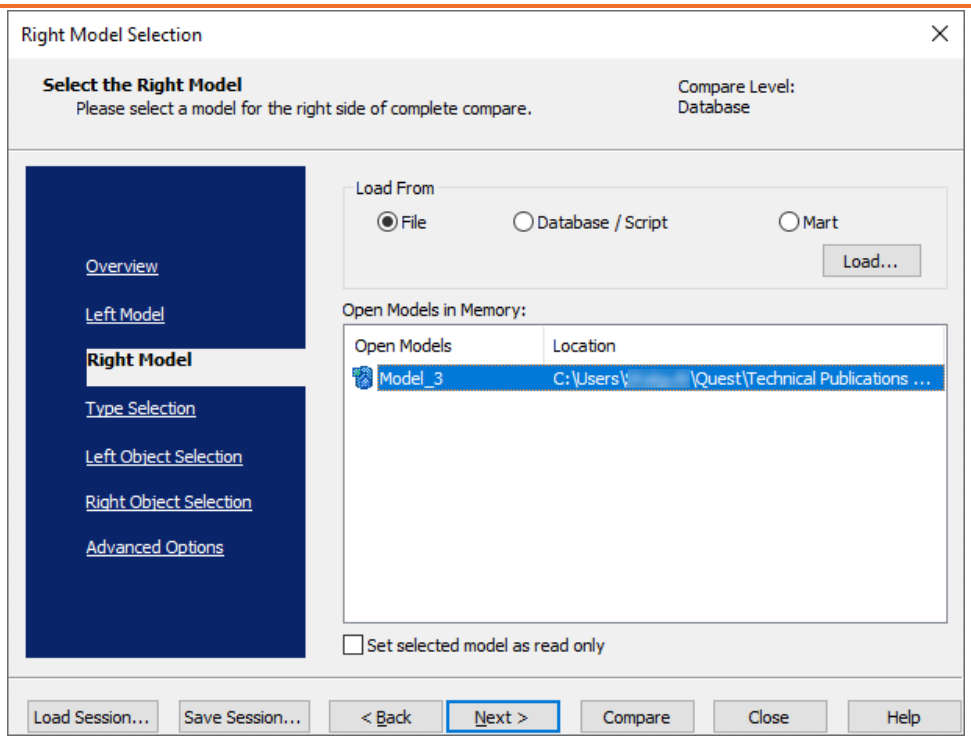
1. Open your Neo4j model.

   Ensure that you are in the Physical mode.

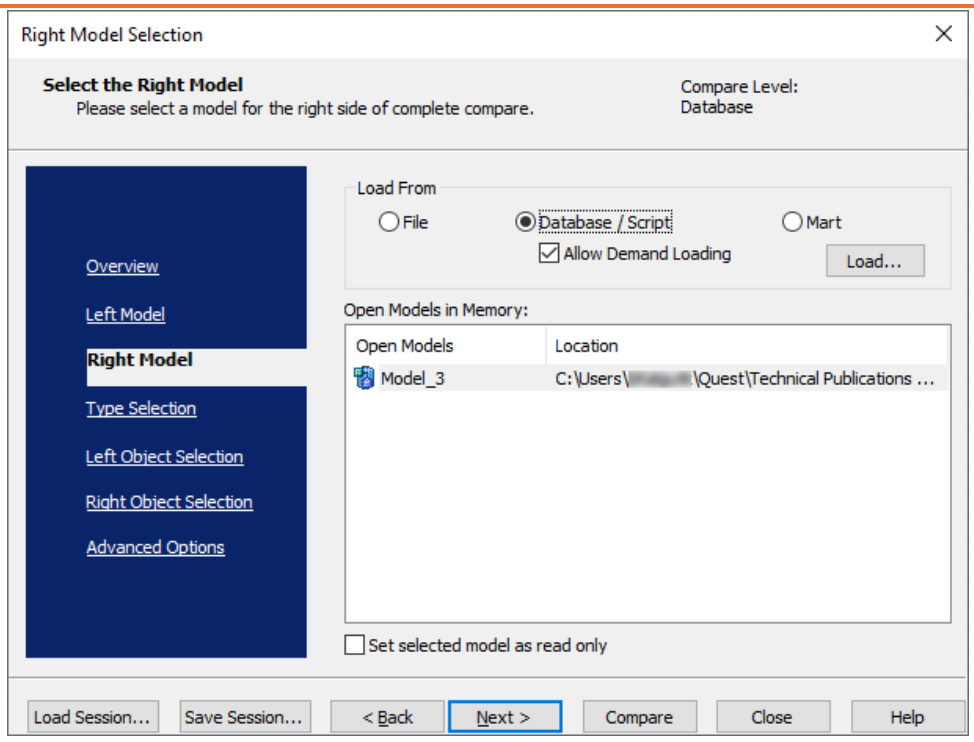
   For example, the following image uses a Neo4j model with three labels and nodes with relationships.



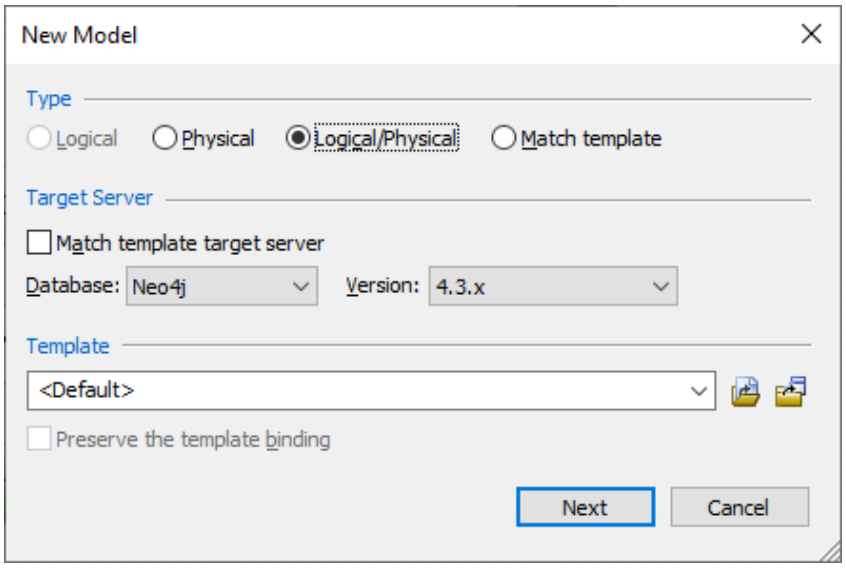
2. Click **Actions** > **Complete Compare**.
   By default, the Complete Compare wizard assigns the open model as the Left Model. Hence, the Right Model tab appears.



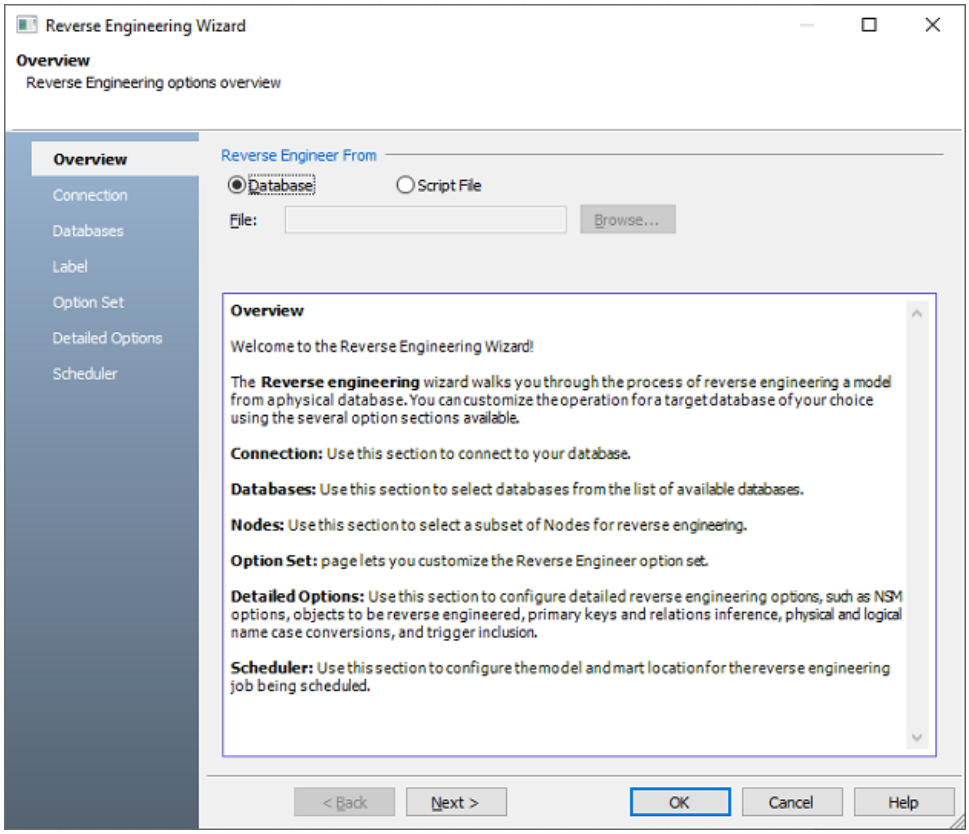
3. Click **Database/Script**.
   By default, the Allow Demand Loading option is selected.



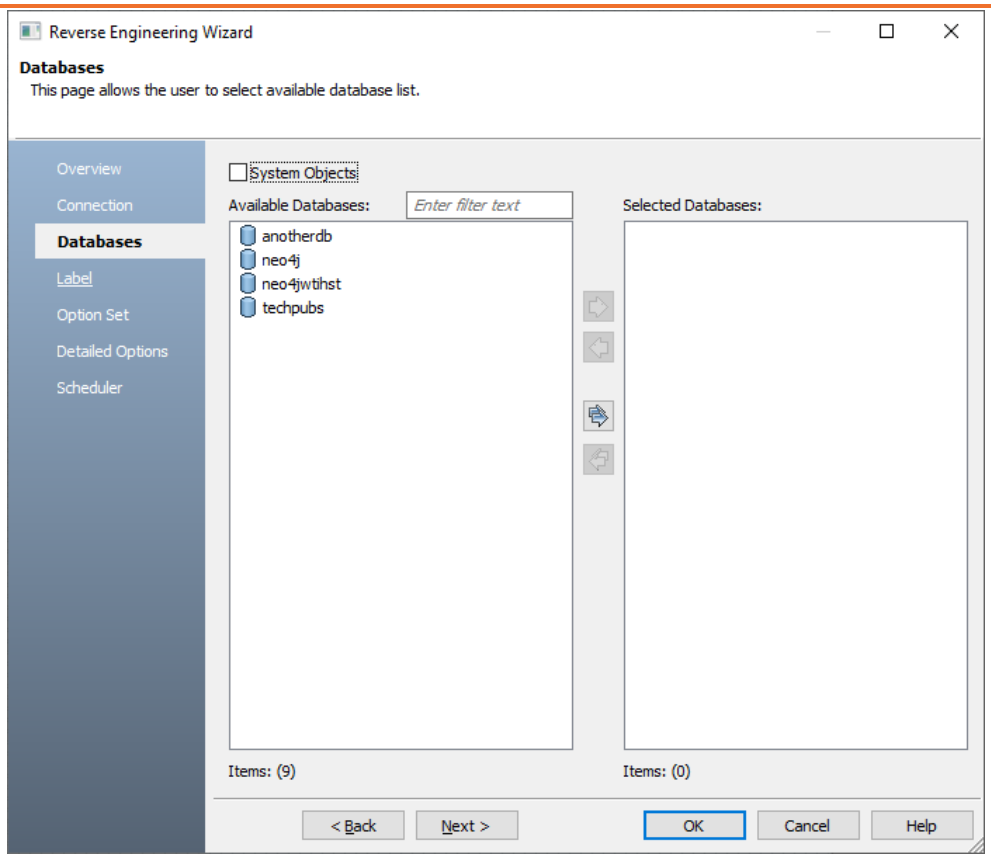
4. Click **Load**.
   The New Model dialog box appears. This starts the reverse engineering process to pull a model from the database to compare.



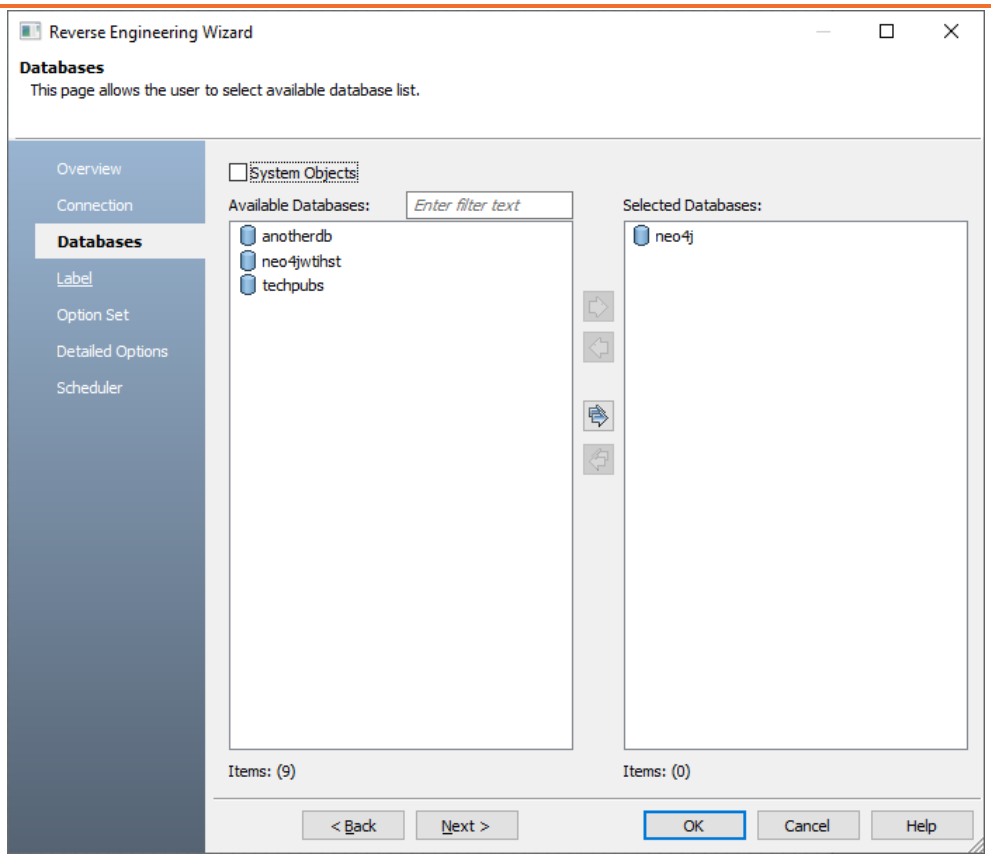
5. Ensure that the Database is set to Neo4j. Then, click **Next**.
   The Reverse Engineering Wizard appears.



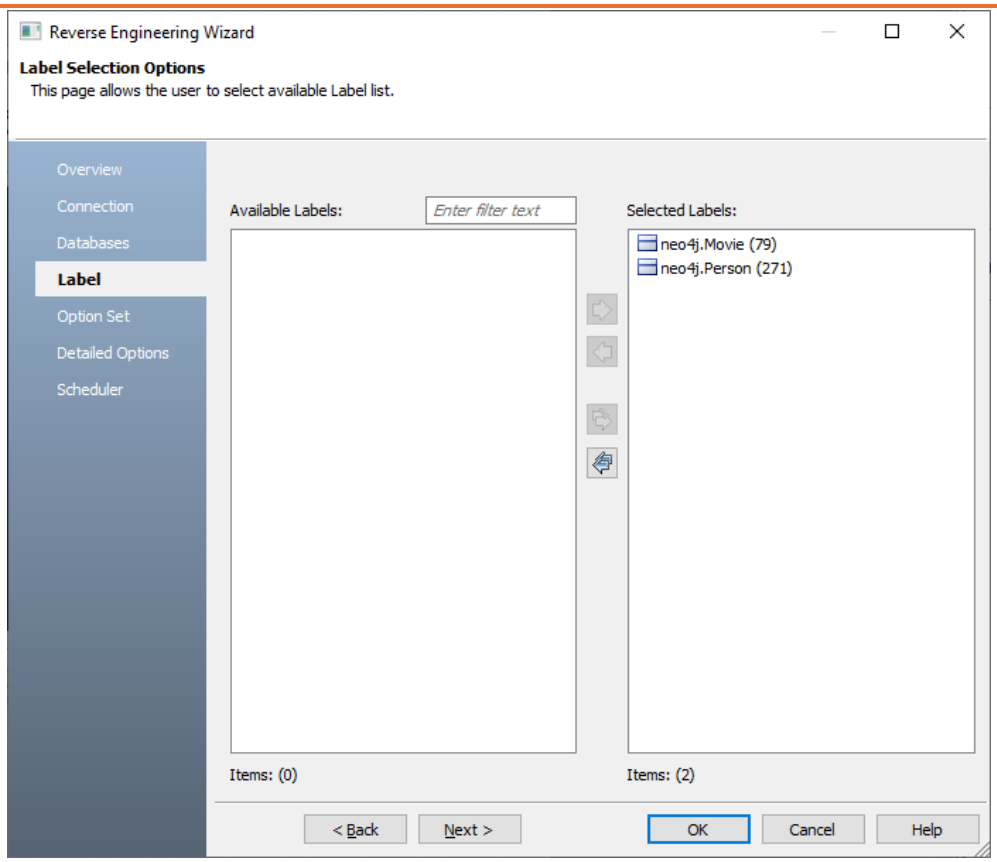
6. Click **Database**. Then, click **Next**.
   The Connection tab appears. Use this tab to connect to the database from which you want to reverse engineer the model.
7. After connection is established, click **Next**.
   The Databases tab appears. It displays a list of available databases.



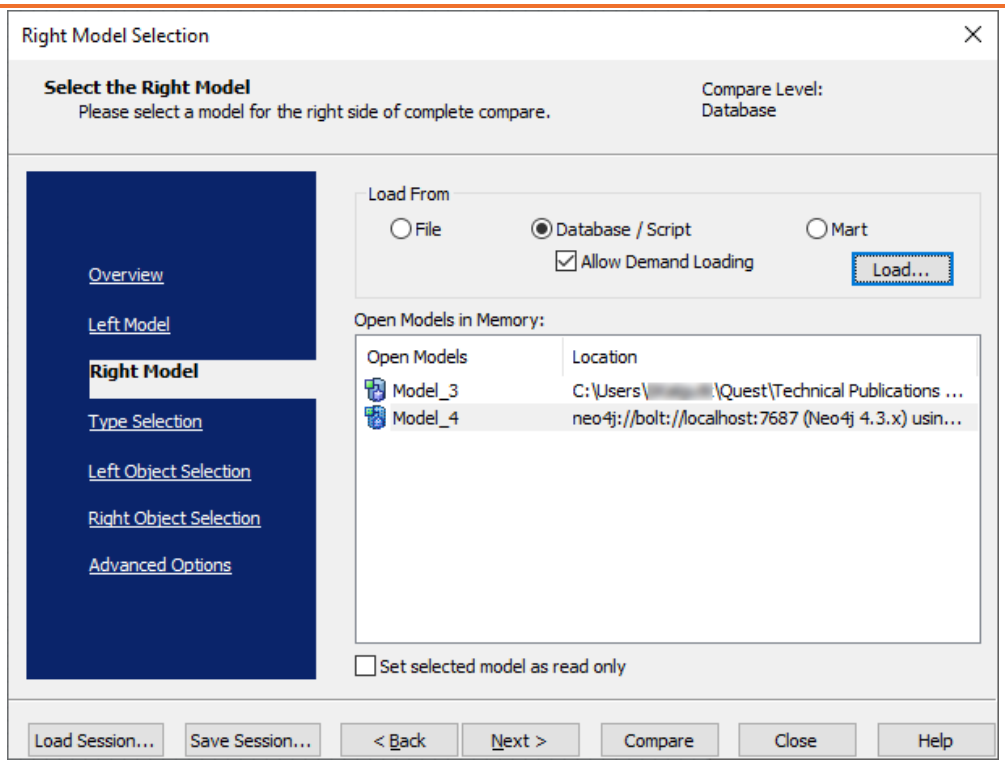
8. Under **Available Databases**, select the databases that you want to reverse engineer. Then, click  .
   This moves the selected databases under Selected Databases.



9. Click **Next** and on the Label tab, click  .
   This selects all the available labels.

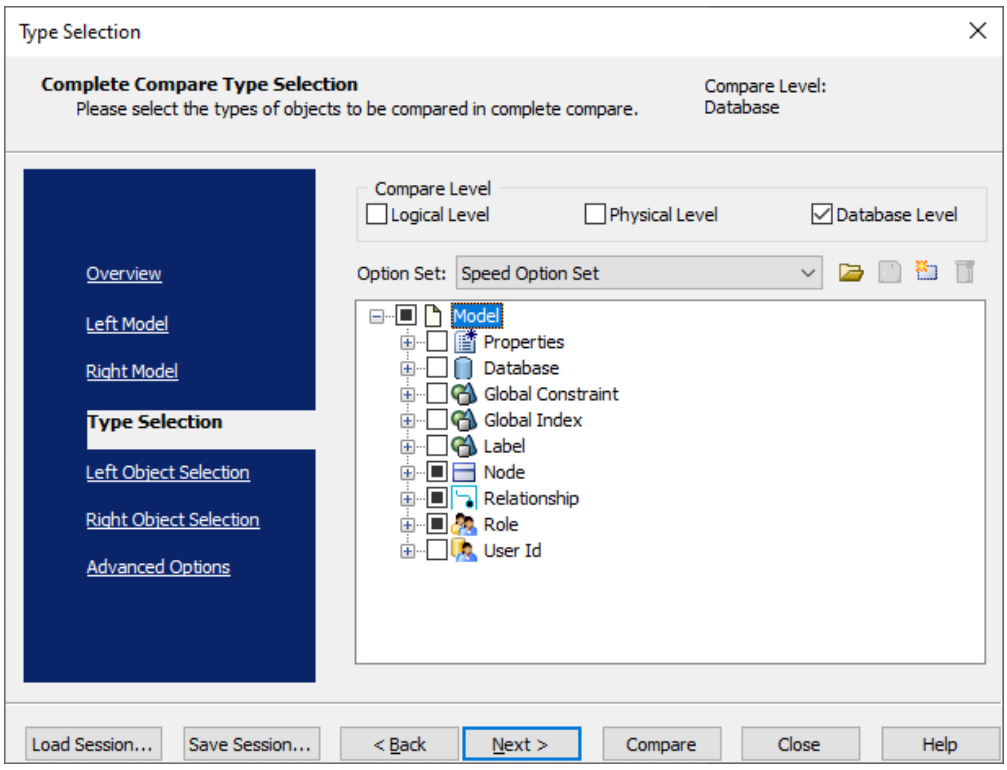


10. Click **Next** and in the Option Set tab, keep the default configuration.
11. Click **Next** and in the Detail Options tab, keep the default configuration.
12. Click **OK**.
    The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse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



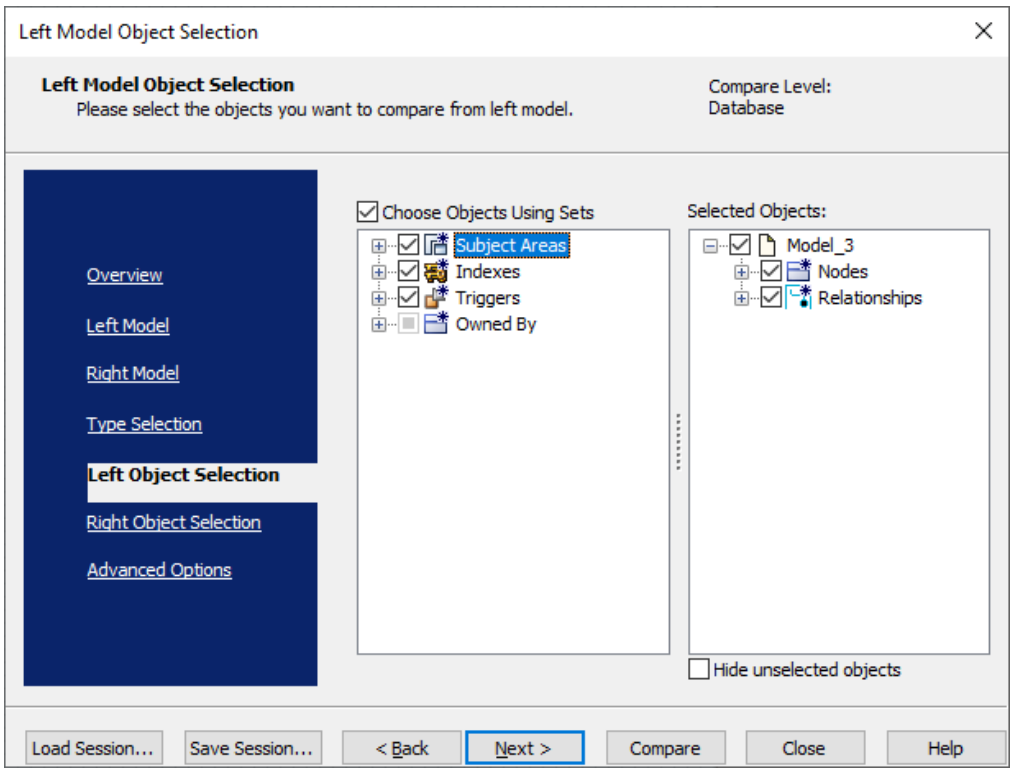
13. Click **Next** and in the Type Selection tab, select the appropriate options.

For example, the following image shows the default options.



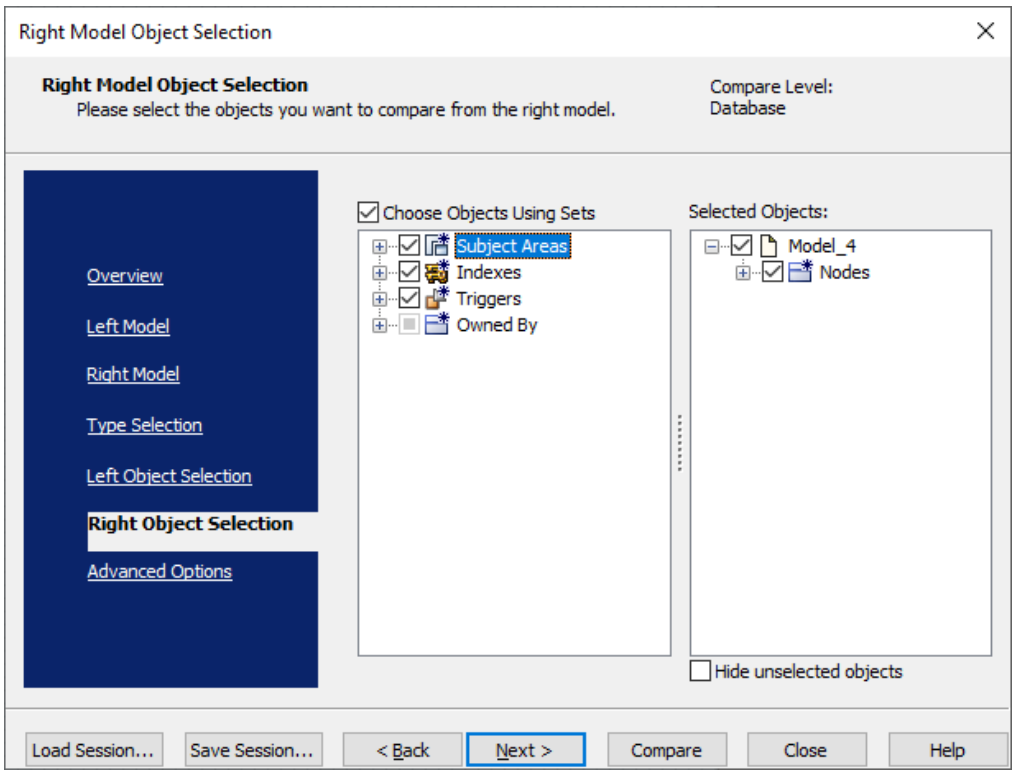
14. Click **Next** and in the Left Object Selection tab, select the appropriate options.

For example, the following image shows the default options.

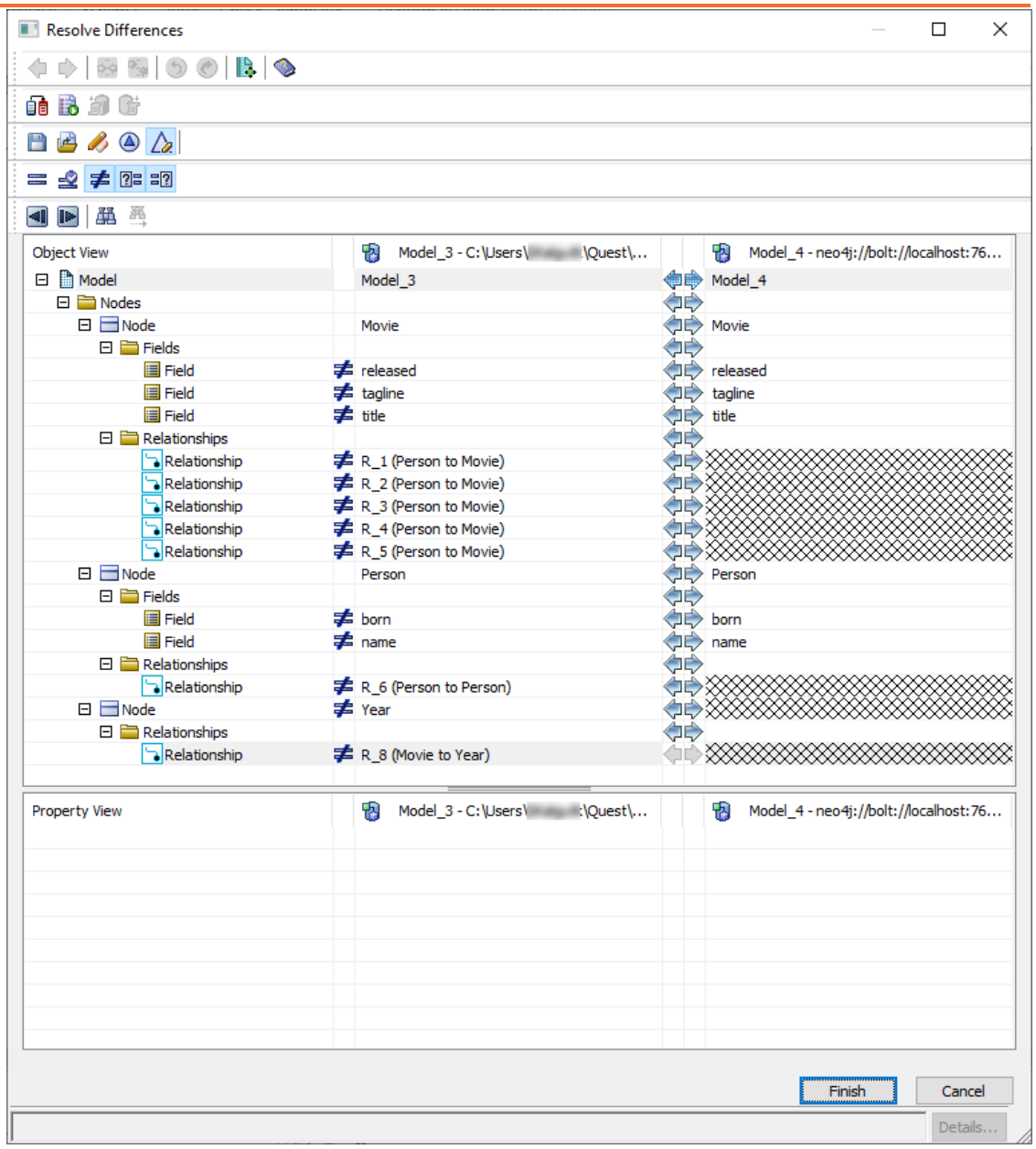


15. Click **Next** and in the Right Object Selection tab, select the appropriate options.

For example, the following image shows the default options.



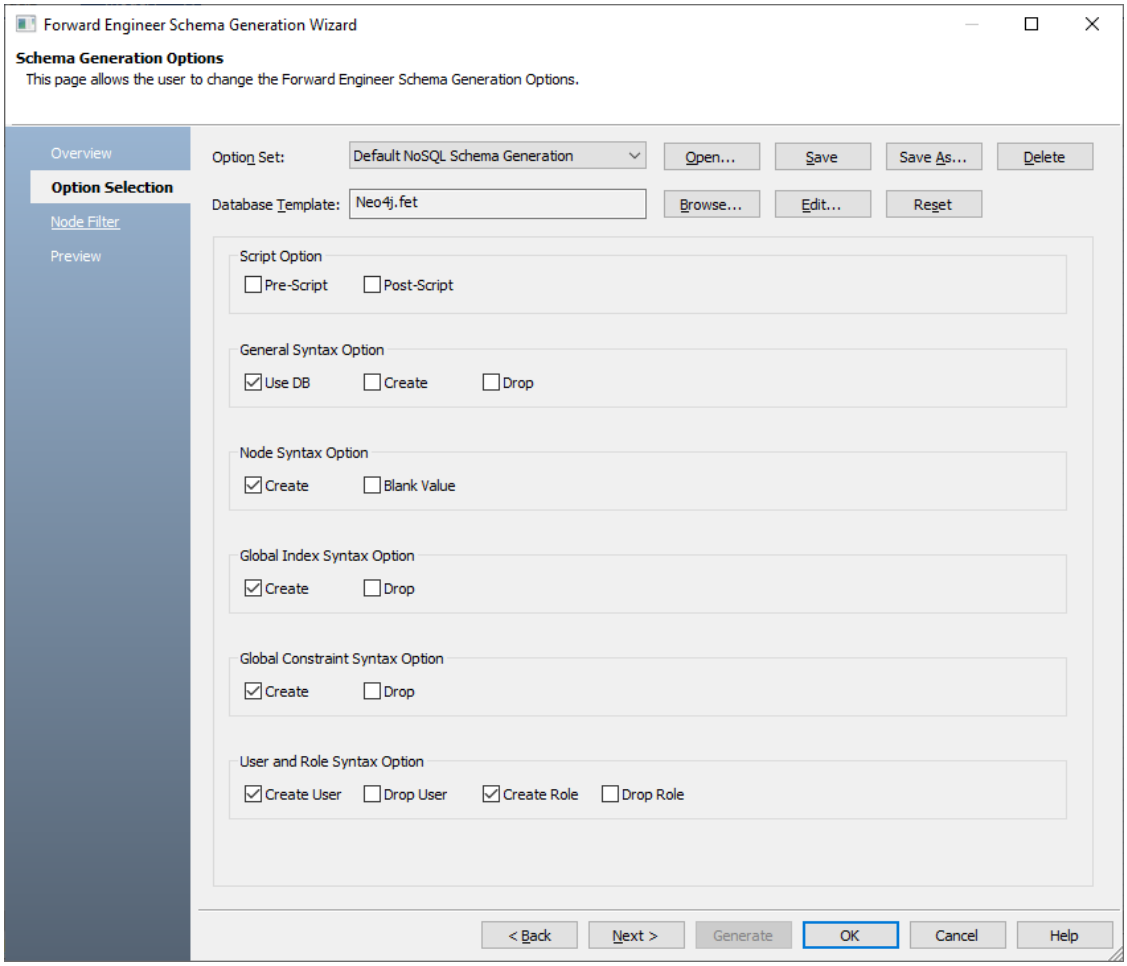
16. Click **Compare**.
    The comparison process runs, and the Resolve Differences dialog box appears. It displays the differences between your model and database.
    For example, the following image shows that the Year node is available in your model but not in the database.



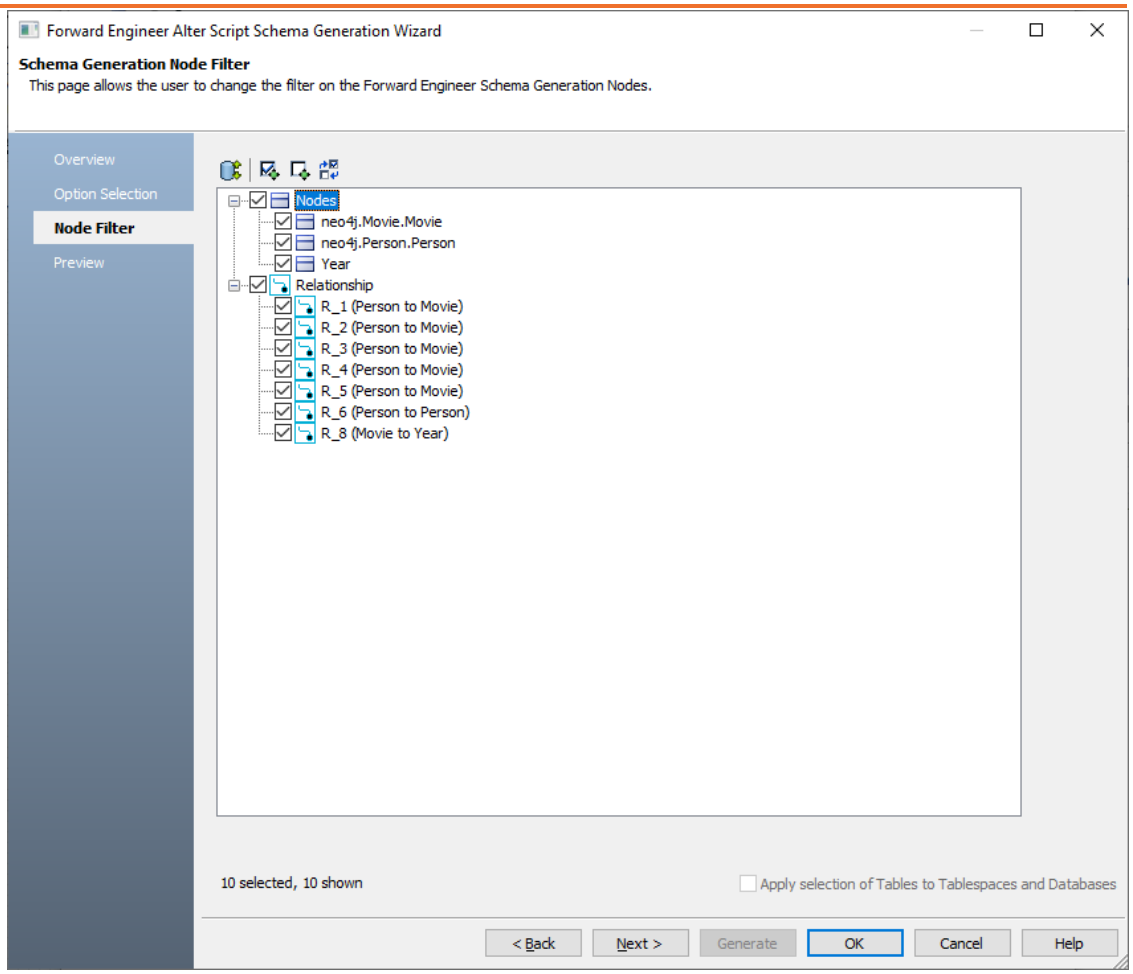
Select the Year node and click . This will move the Year node to the right model (from the database). Similarly, resolve other differences.

17. As differences were moved to the right model, click .
    This opens the forward engineering wizard.

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



19. Click **Node Filter** and select or verify the nodes 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 Neo4j 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**.

Migrating Relational Models to Neo4j Models

You can convert and migrate your relational models to Neo4j 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 Neo4j model.

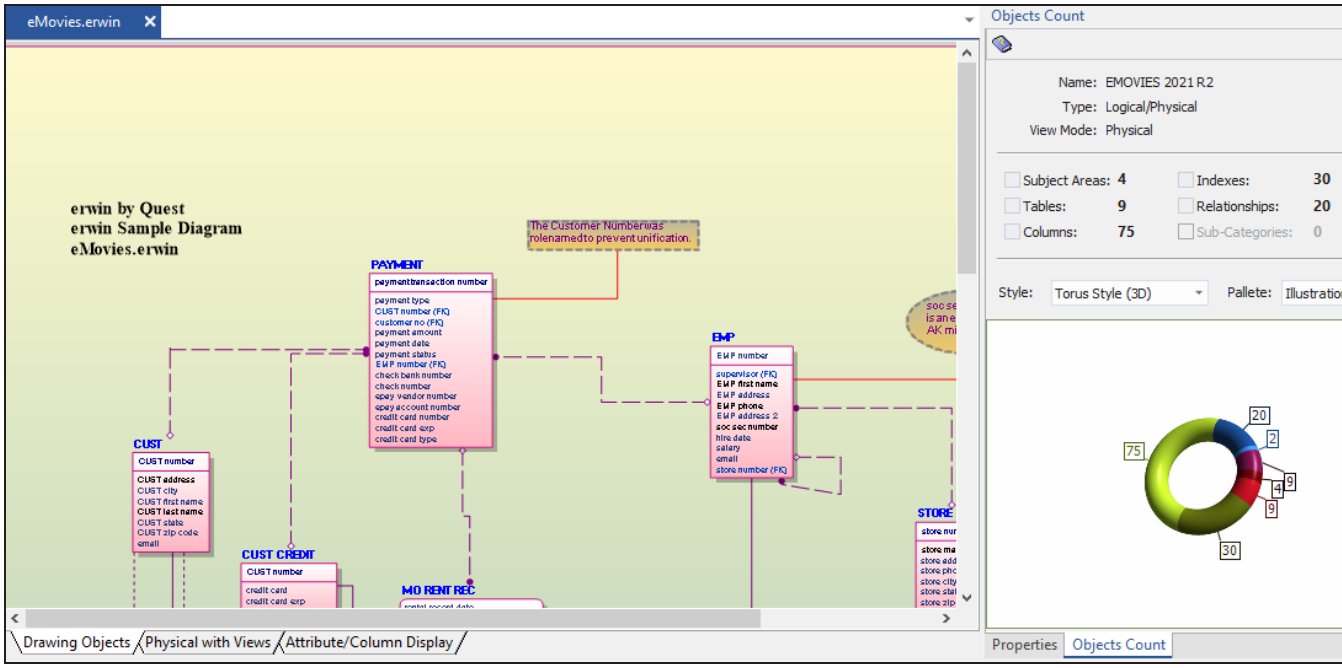
Migration by Changing the Target Database

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

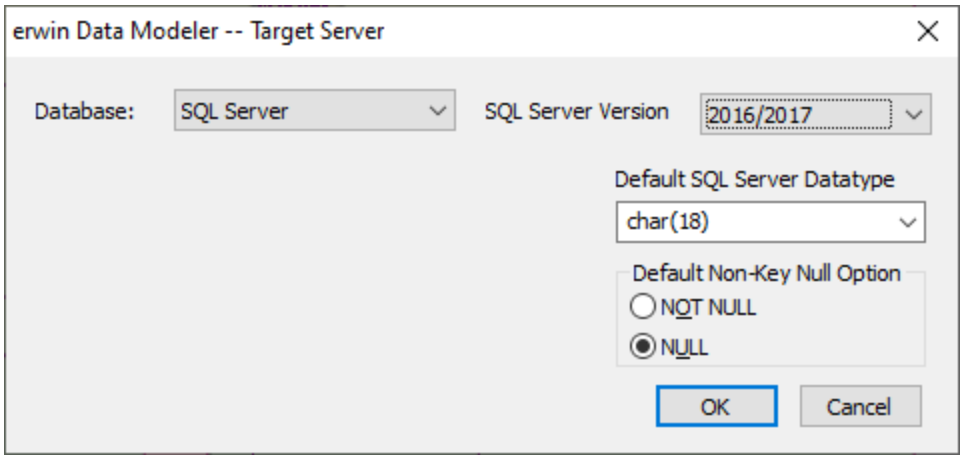
1. Open your relational model.

   Ensure that you are in the Physical mode.

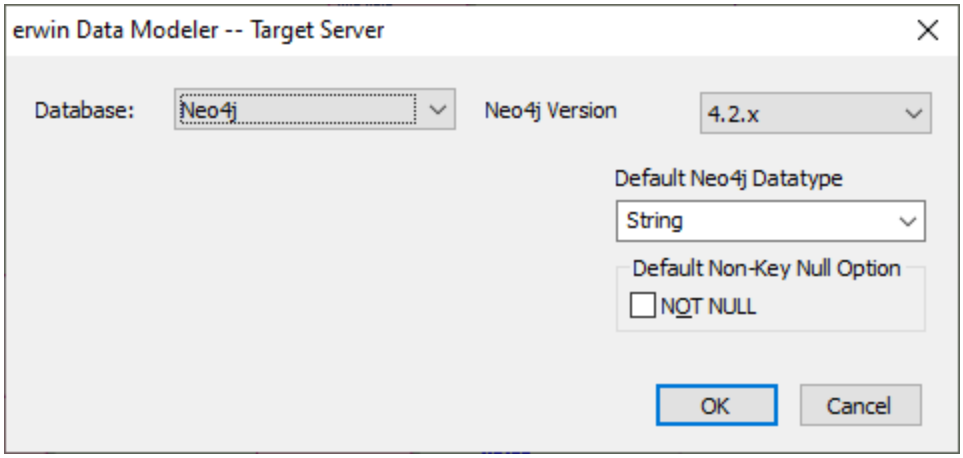
   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.



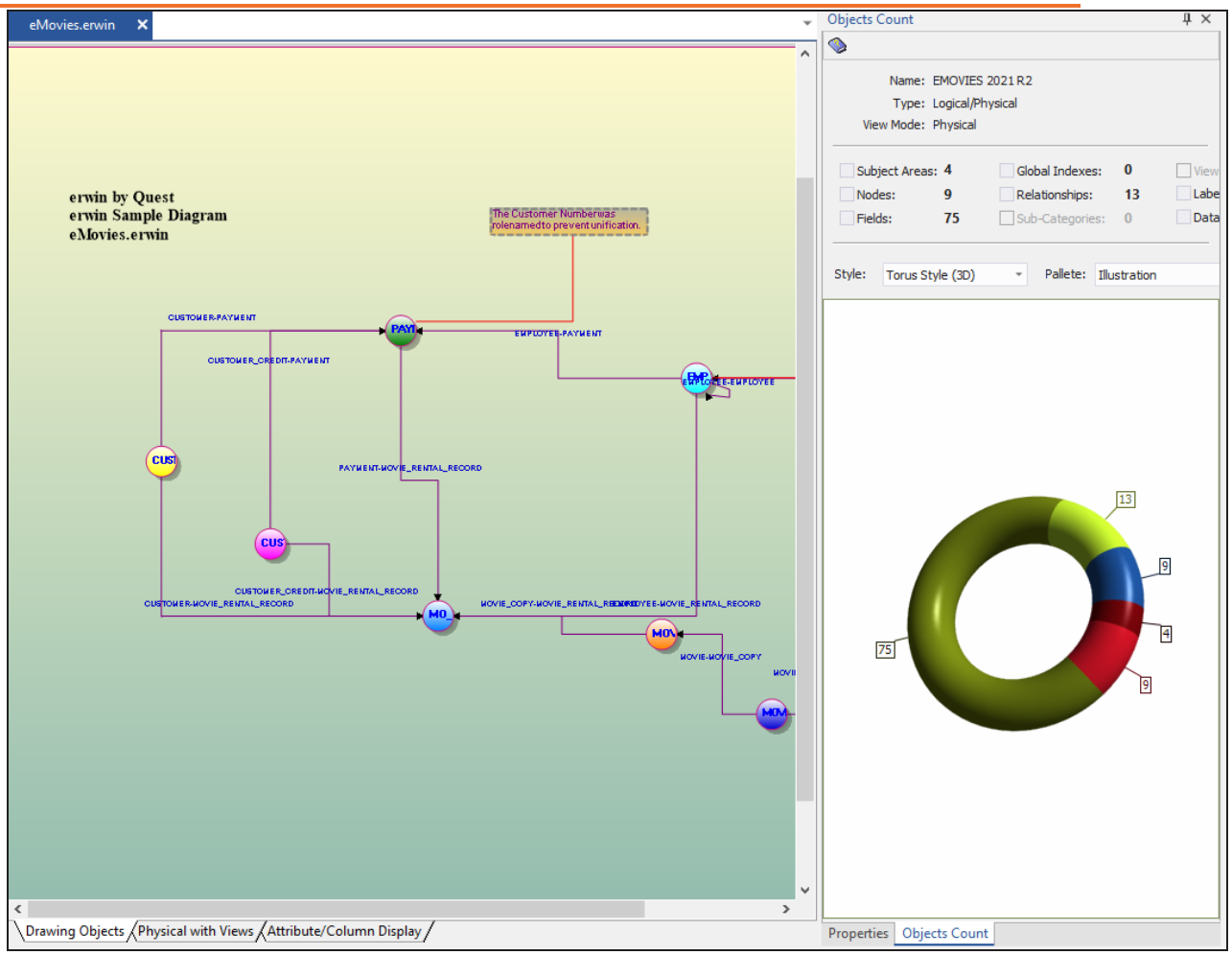
2. On the ribbon, click **Actions** > **Target Database** or on the status bar, click the database name.
   The erwin Data Modeler -- Target Server screen appears.



3. In the **Database** drop-down list, select Neo4j.



4. Click **OK**.
   Once the conversion is complete, the existing model is migrated to a Neo4j model.



In the **Objects Count** pane, note that instead of tables and columns, we now have nodes, labels, and fields. The migration process converts tables, columns, and relationships to the NoSQL format according to the database that you select.

Migration by Deriving a Model

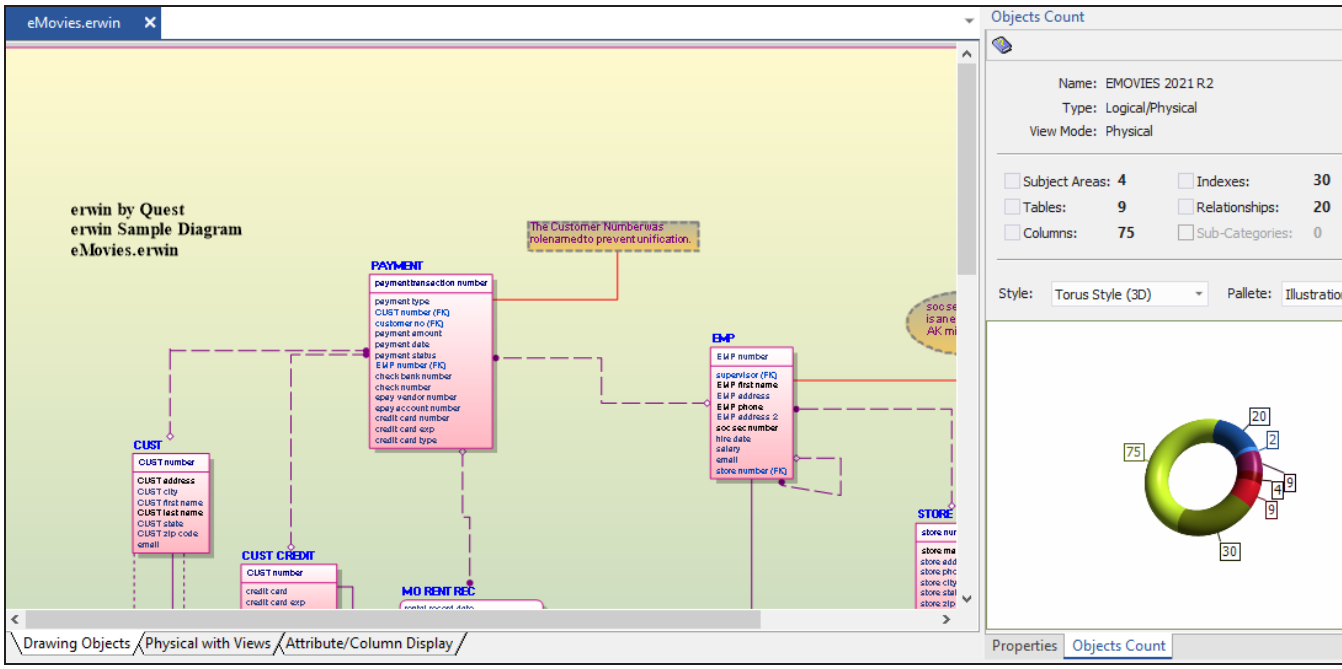
To migrate by deriving a model, follow these steps:

1. Open your relational model.

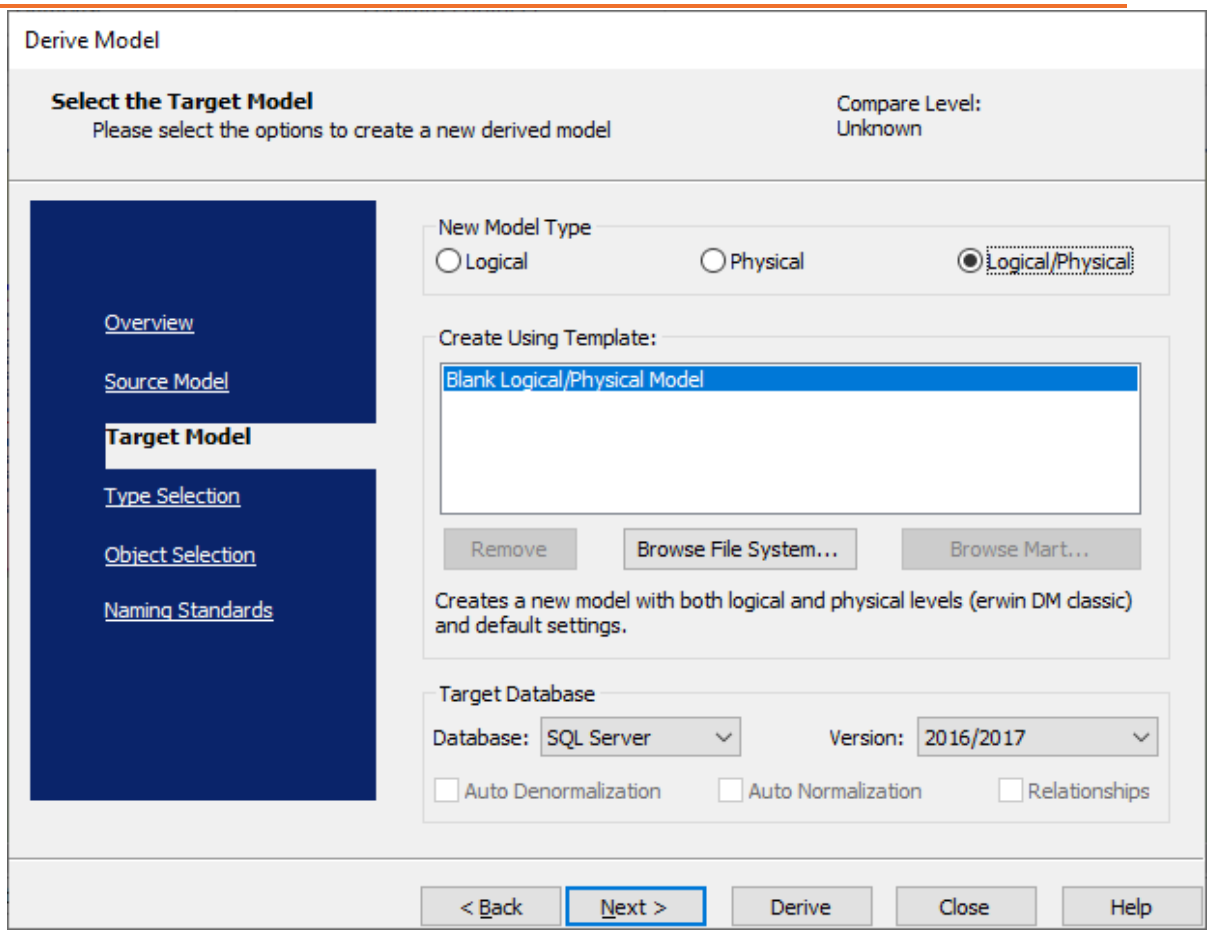
   Ensure that you are in the Physical mode.

   For example, the following image uses the sample eMovies.erwin model. In the

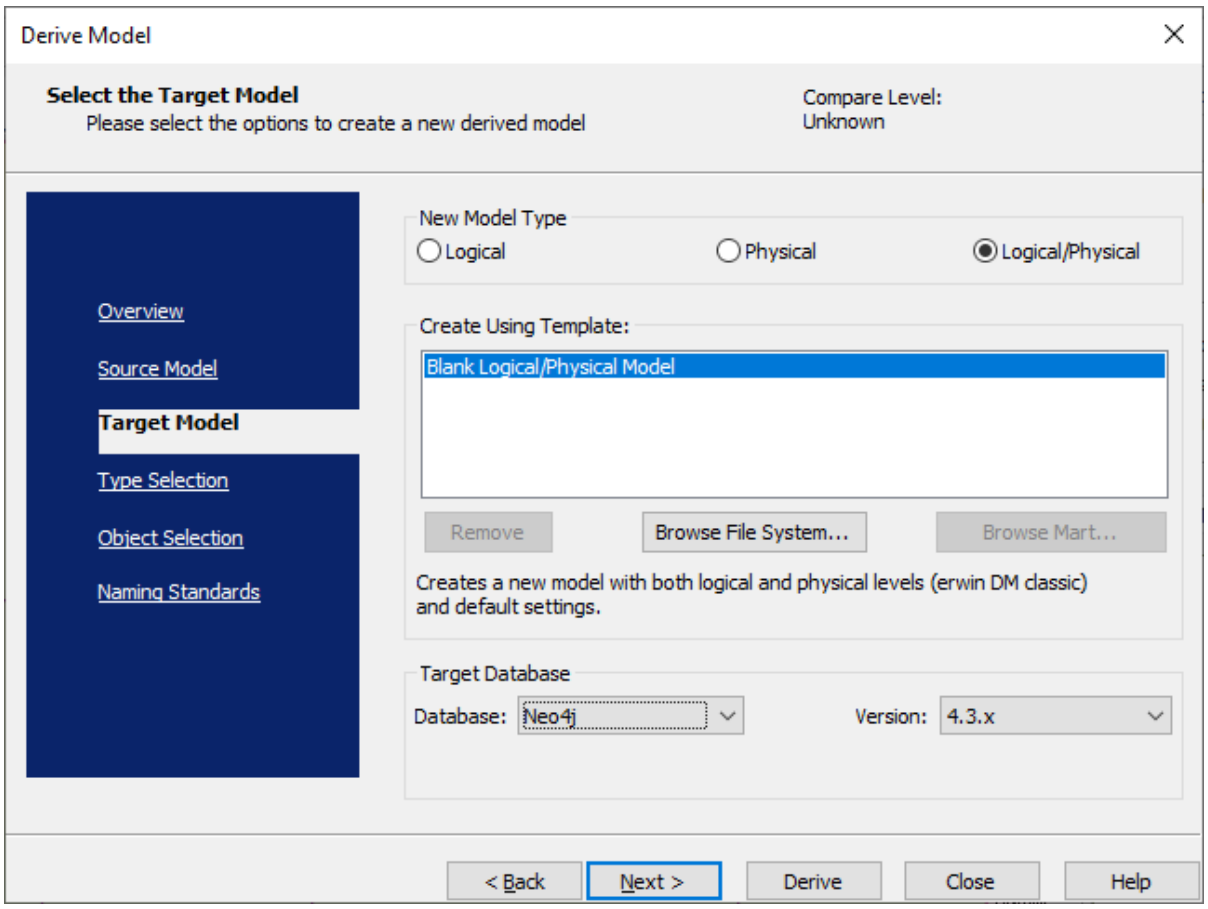
Objects Count pane, note the number of tables, columns, and relationships.



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.


Derive Model
Select the Target Model
Please select the options to create a new derived model
Compare Level:
Unknown
Overview
Source Model
Target Model
Type Selection
Object Selection
Naming Standards
New Model Type
Logical
Physical
Logical/Physical
Create Using Template:
Blank Logical/Physical Model
Remove
Browse File System...
Browse Mart...
Creates a new model with both logical and physical levels (erwin DM classic) and default settings.
Target Database
Database:
SQL Server
Version:
2016/2017
Auto Denormalization
Auto Normalization
Relationships
< Back
Next >
Derive
Close
Help

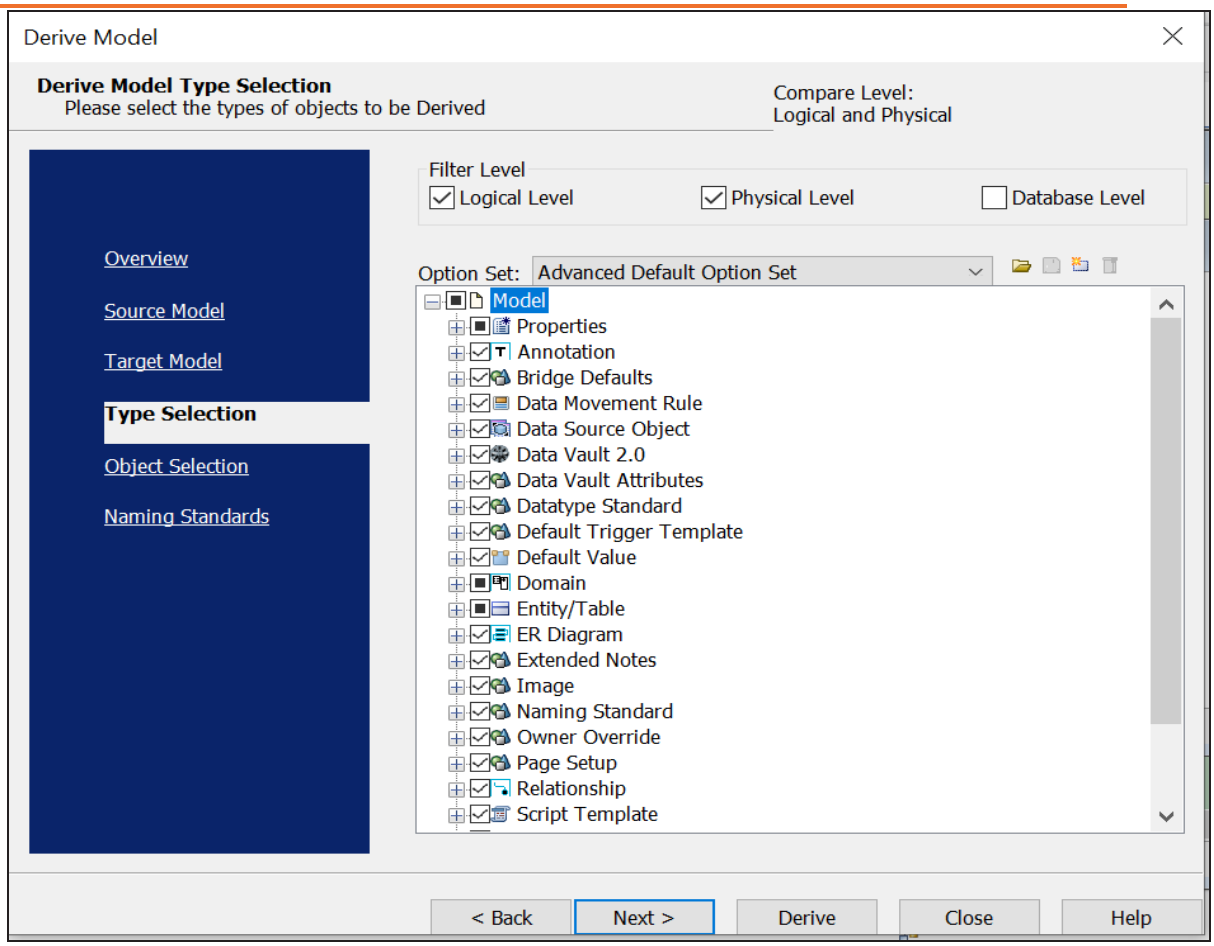
3. In the **Database** drop-down list, select **Neo4j**.



4. Click **Next**.

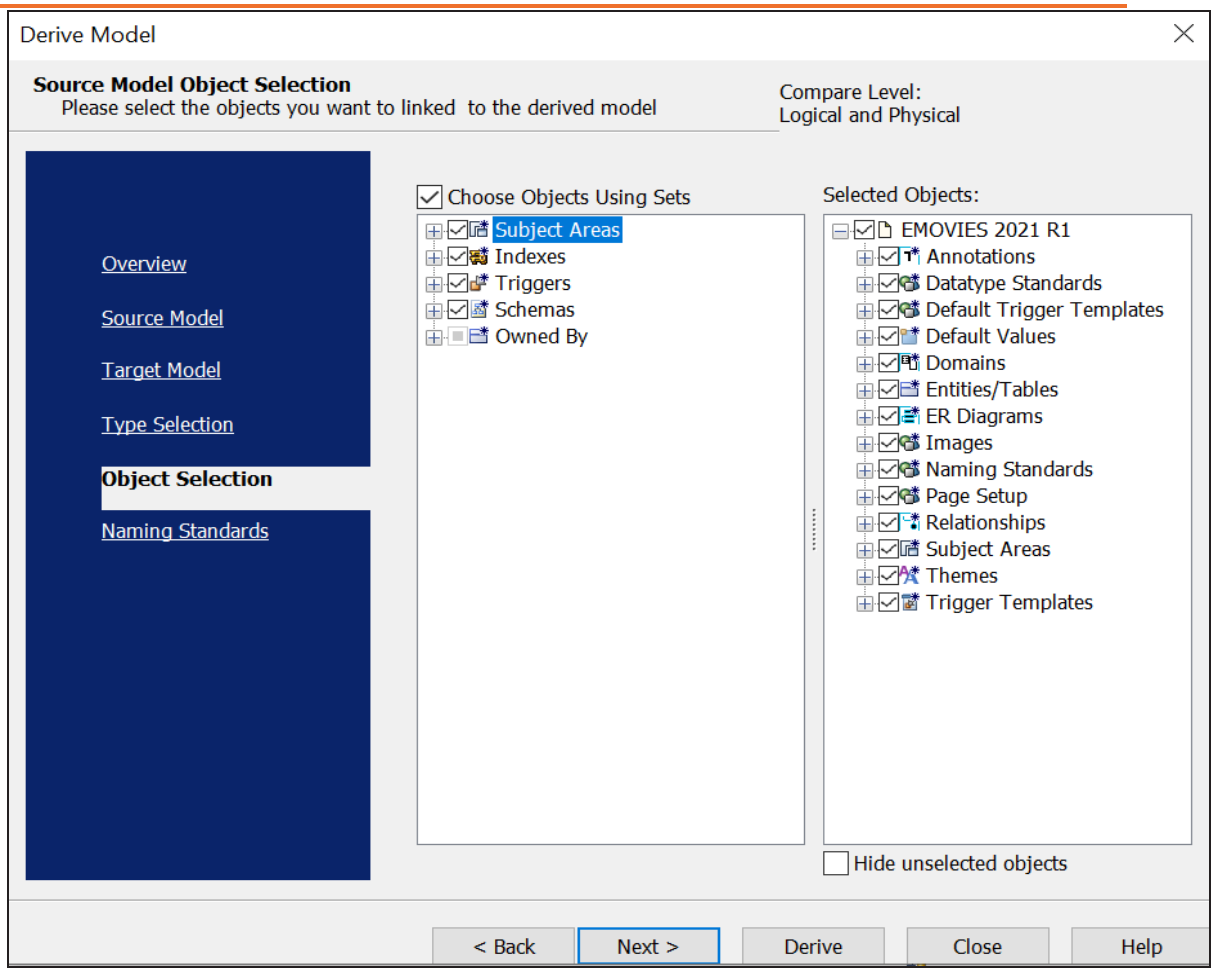
> If the Type Resolution screen appears, click **Finish**.

The Type Selection tab appears.



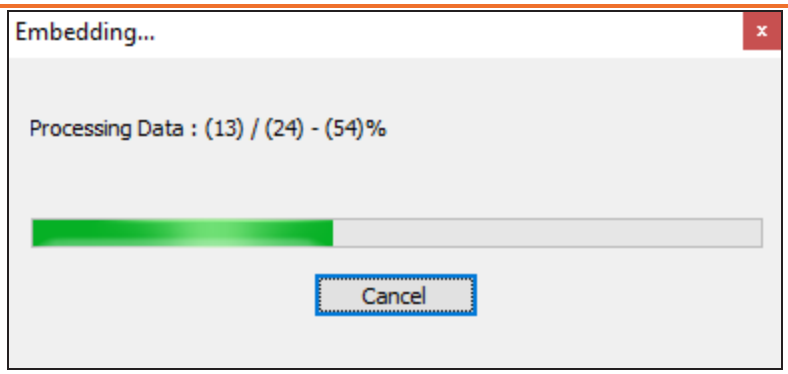
5. Select the types of objects that you want to derive into the target Neo4j model.

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

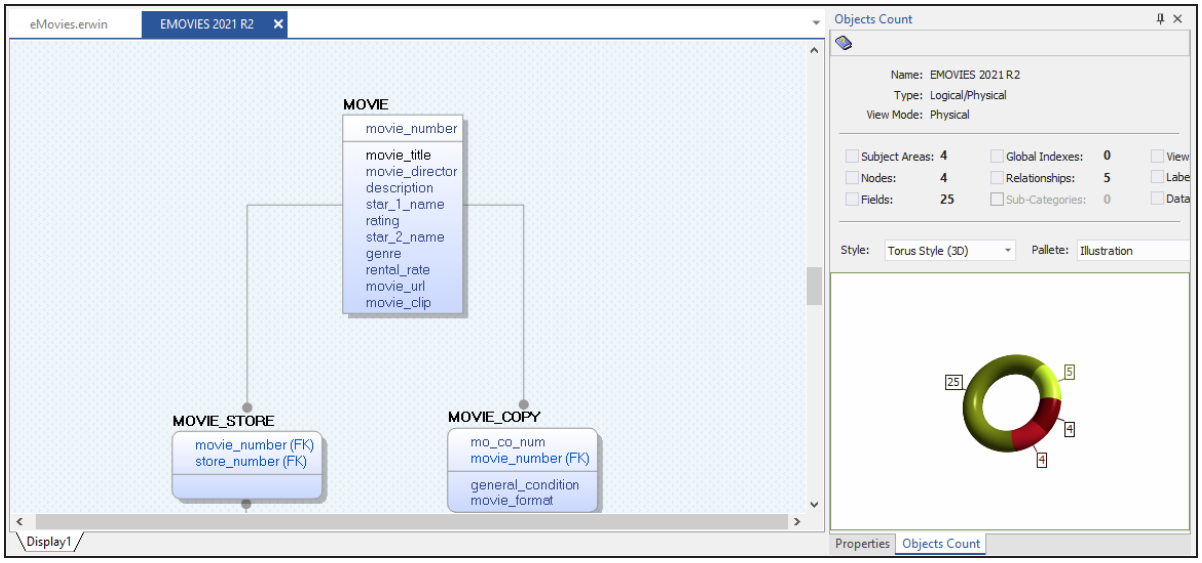


7. Select the objects that you want to derive into the target Neo4j model.

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

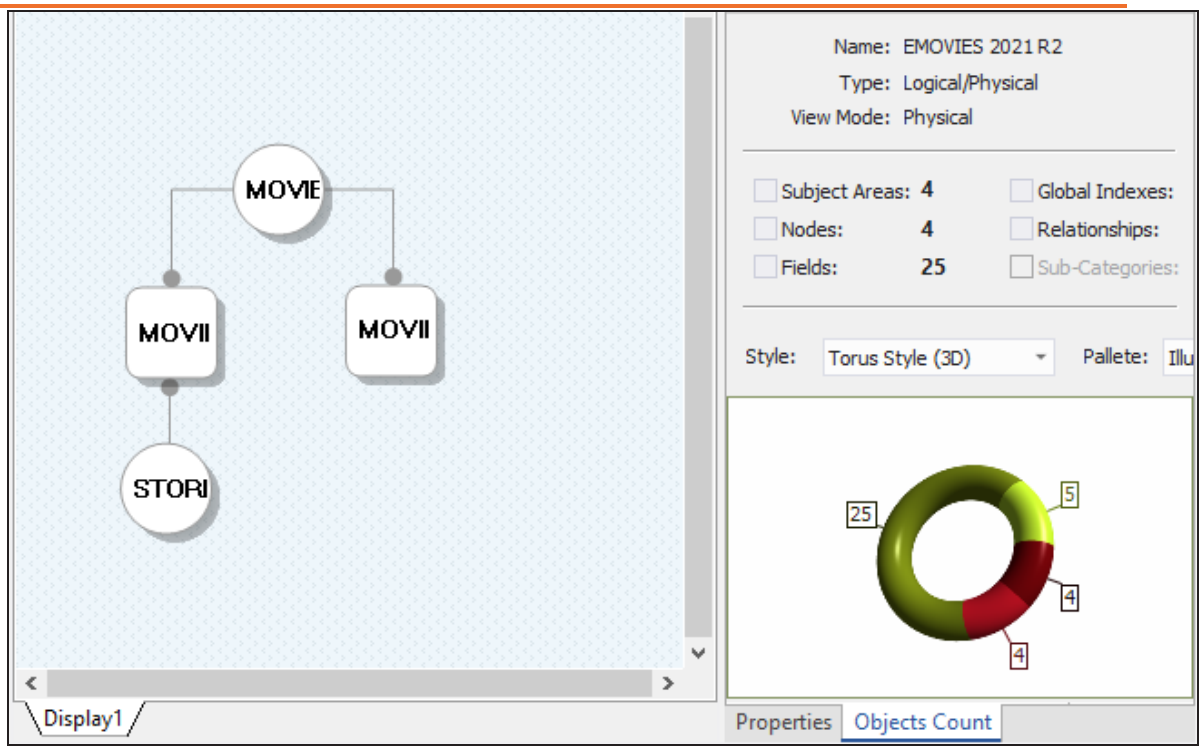


Once the conversion is complete, the existing model in migrated to a Neo4j database.



In the **Objects Count** pane, note that instead of tables and columns, we now have nodes and fields. The migration process converts tables, columns, and relationships to the NoSQL format according to the database that you select.

The derived model has table-like representation. To convert it to graph-like representation, on the ribbon, go to **View** > **Display Level** group. Then, click . This converts the model diagram as follows:


MOVIE
MOVII
MOVII
STORI
Display1
Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical
Subject Areas: 4
Nodes: 4
Fields: 25
Global Indexes:
Relationships:
Sub-Categories:
Style: Torus Style (3D)
Pallete: Illu
25
5
4
4
Properties
Objects Count

Parquet Support

erwin Data Modeler (DM) now supports Parquet 2.x as a target database. This implementation supports the following objects:

- Record
  - Field

Following are the supported data types:

- ARRAY
- BINARY
- BOOLEAN
- BYTE ARRAY
- DOUBLE
- FIXED LEN BYTE ARRAY
- FLOAT
- GROUP
- INT 32
- INT 64
- INT 96
- UNKNOWN

Parquet implementation supports all erwin DM features and functions. The following sections walk you through these features:

- Reverse engineering models from script
- Forward engineering models to the file format
- Comparing changes using Complete Compare
- Migrating relational models to Parquet models

Reverse Engineering Models

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

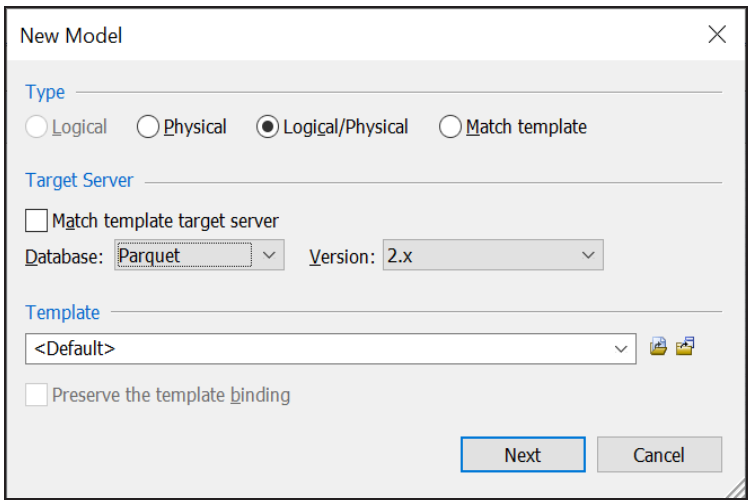
This topic walks you through the steps to reverse engineer a Parquet model. For detailed description of reverse engineering options, refer to the Reverse Engineering Options topic.

To reverse engineer the Parquet model:

1. In erwin Data Modeler (DM), click **Actions** > **Reverse Engineer**.

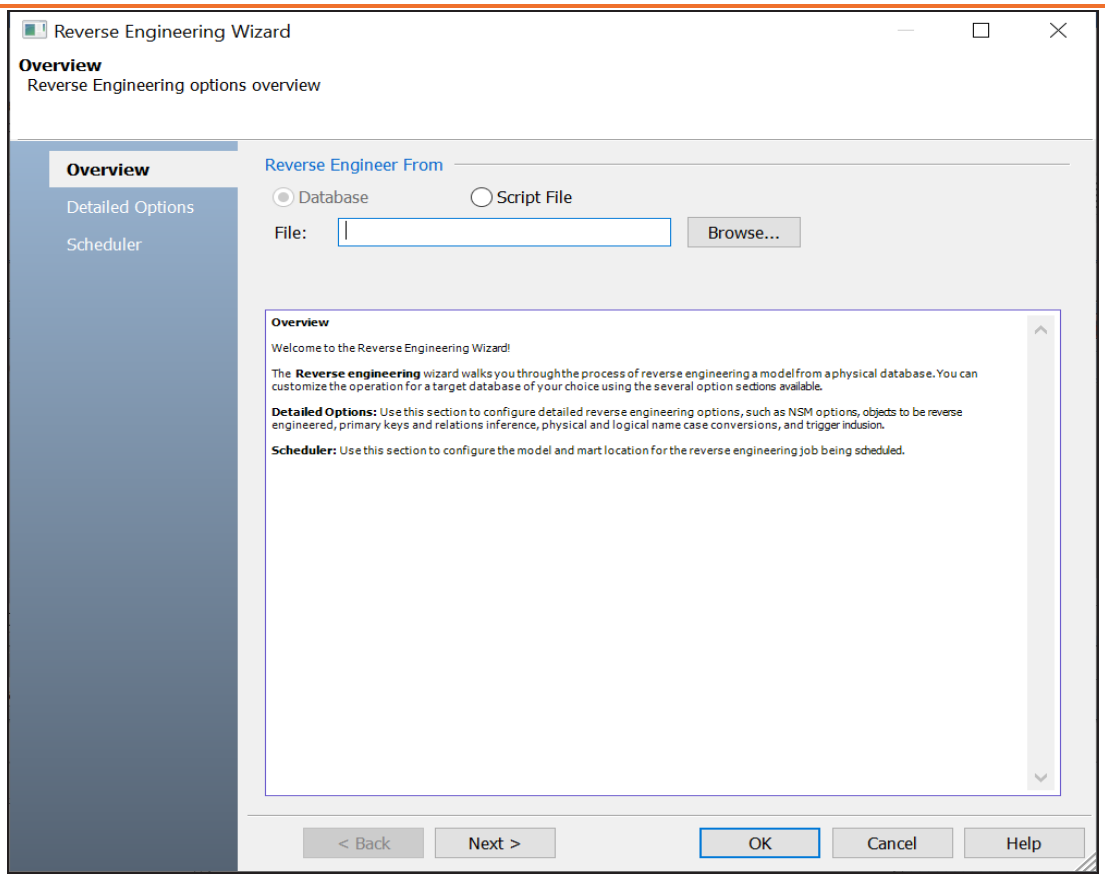
   The New Model screen appears.

2. Click **Logical/Physical** and set **Database** to **Parquet**.



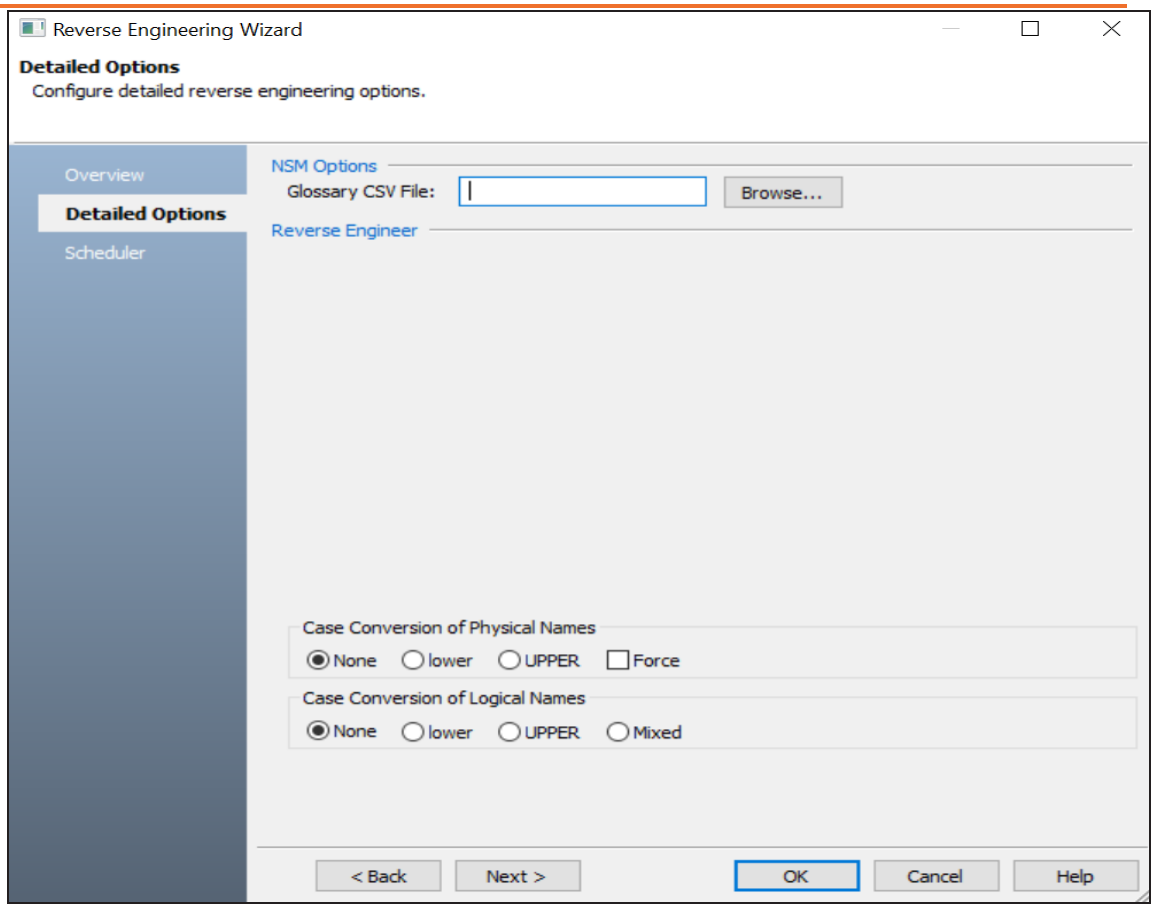
3. Click **Next**.

   The Reverse Engineering Wizard appears.



4. Click **Script File**.
5. Click **Browse** to select the required JSON file.
6. Click **Next**.

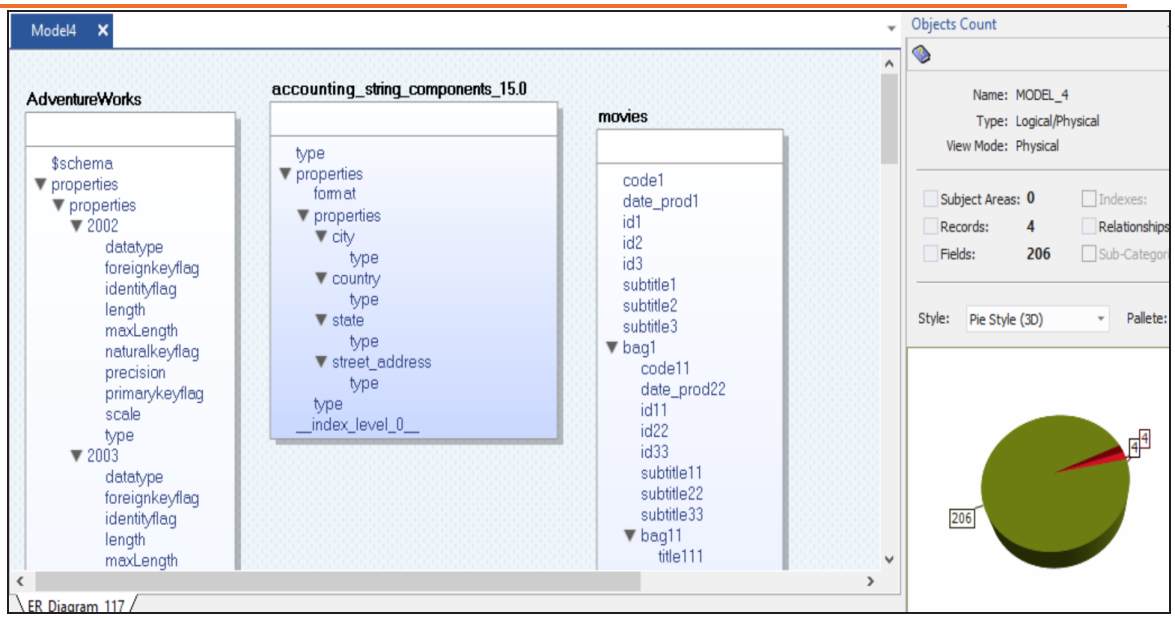
   The Detailed Options tab appears.



7. Set up appropriate options based on your requirement.
8. 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. For example, in the following model, four records and 206 fields are created. In the account_string_components_15.0 record, the city field is nested inside the properties field, which is nested under another properties field.



Reverse Engineering Options for Parquet

Following are the reverse engineering options for Parquet.

Overview

Field	Description	Additional Information
Reverse Engineer From	Specifies whether you want to reverse engineer from a script or database	The Database option is not available for Parquet.
File	Specifies the script file location	

Detailed Options

Parameter	Description	Additional Information
NSM Options	Specifies the naming standard glossary file in the .CSV format	

Case Conversion of Physical Names	Specifies how the case conversion of physical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Force**: Indicates whether the physical name property of all the logical/physical models is overridden. If this option is enabled, the logical/physical link is broken between the logical and physical name. If this option is not enabled, all logical and physical names are set to the same value after the process completes.
Case Conversion of Logical Names	Specifies how the case conversion of logical names is handled	**None**: Indicates that the case in the script file is preserved **lower**: Indicates that the names are converted to lower case **UPPER**: Indicates that the names are converted to upper case **Mixed**: Indicates that the mixed-case logical names are preserved

Scheduler

Parameter	Description	Additional Information
Model	Specifies the location and name of the reverse engineered model	For example: C:\Scheduler\<Model Name>.erwin When you schedule a job on a remote server, ensure the model path is same for remote and local server.
Mart Folder	Specifies the location or library in your mart where the reverse engineered model is saved	To use this option, ensure that you are connected to a mart. For more information, refer to the Connecting to Mart topic.
Complete Compare	Specifies whether the Complete Compare (CC) process should run while reverse engineering	
Output File	Specifies the location of the	

	CC output file generated	
File	Specifies that the target model location is on the local system	
Mart	Specifies that the target model location is in the mart	
Using Latest Version	Specifies whether the target model is the latest version of the model in the mart	This option is available only when Mart is selected.
Save To Mart	Specifies whether the reverse engineered model is saved to the mart	This option is available only when Using Latest Version is selected.
Target Model	Specifies the location of the target model for CC	
Option Set	Specifies the option set that is used for CC	**Advanced Default Option Set**: Indicates that all erwin DM metadata is included. CC works slowest with this option. **Speed Option Set**: Indicates that only the essential metadata is included. CC works the fastest with this option set. **Standard Default Option Set**: Indicates that standard metadata is included. CC works fast with this option set compared to the Advanced option set.

Forward Engineering Models

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

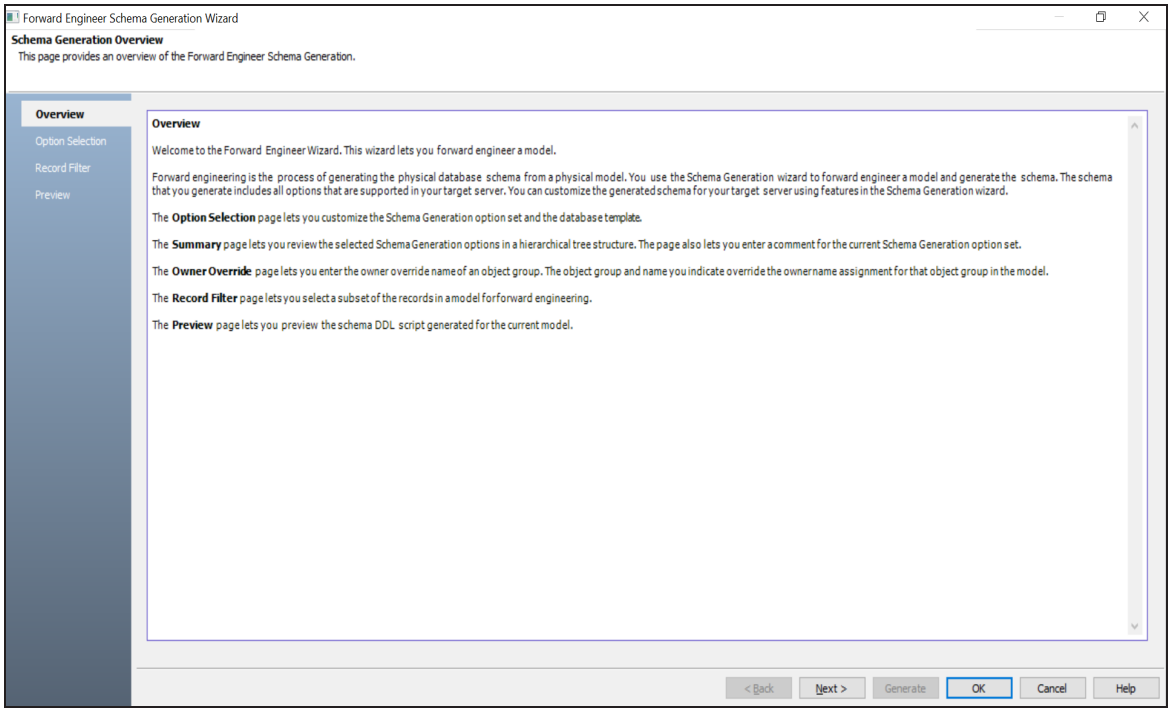
This topic walks you through the steps to forward engineer a Parquet model. For detailed of forward engineering options, refer to the Forward Engineering Options topic.

To forward engineer a Parquet model:

1. Open your Parquet model.

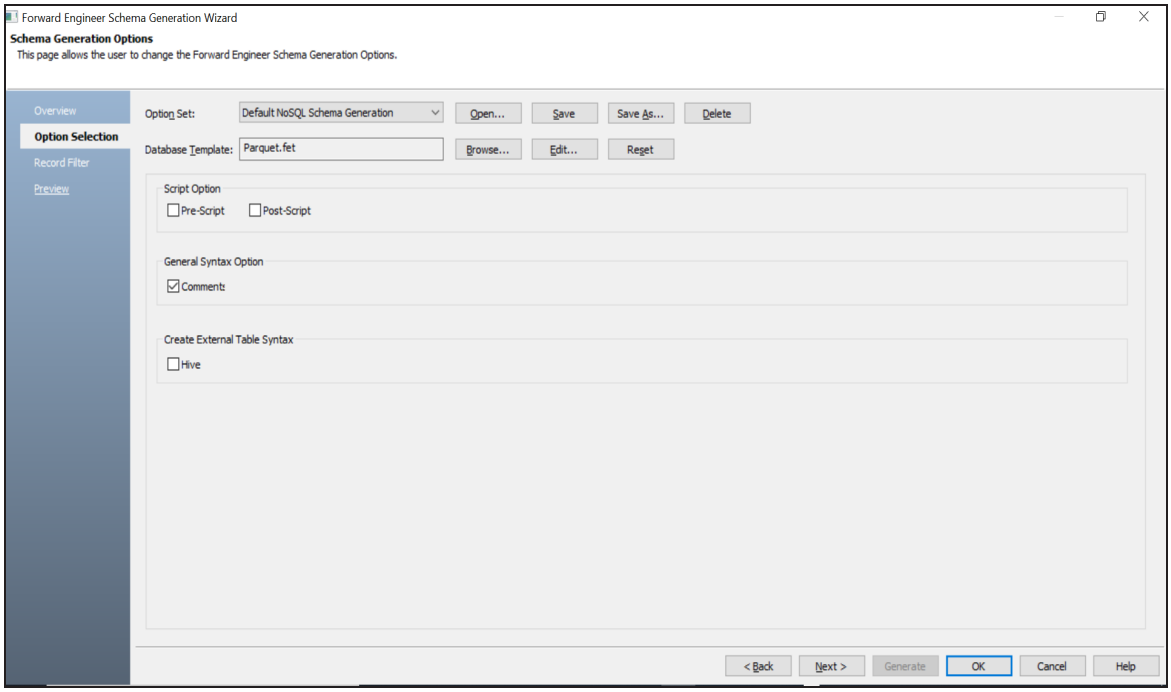
   Ensure that you are in the Physical mode.

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

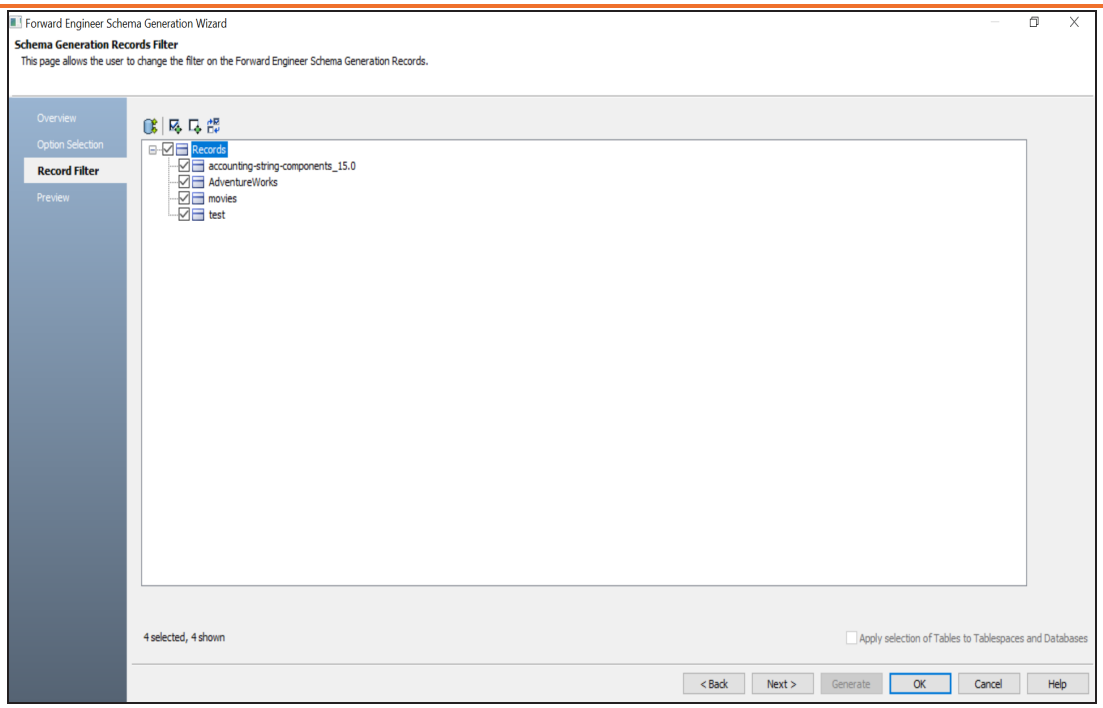


3. Click **Option Selection**.
   The Option Selection tab displays the default option set. Select appropriate syntax

options.

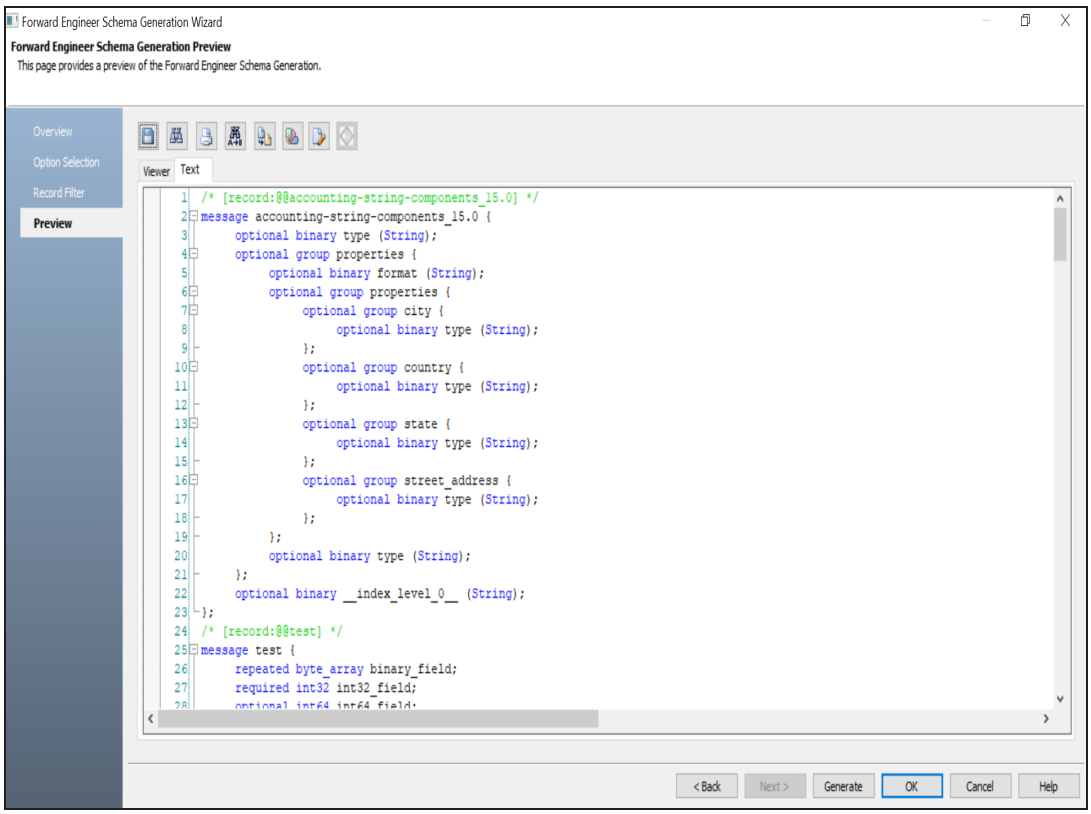


4. Click **Next**.
   The Record Filter tab appears. It displays a list of records available in your model.



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

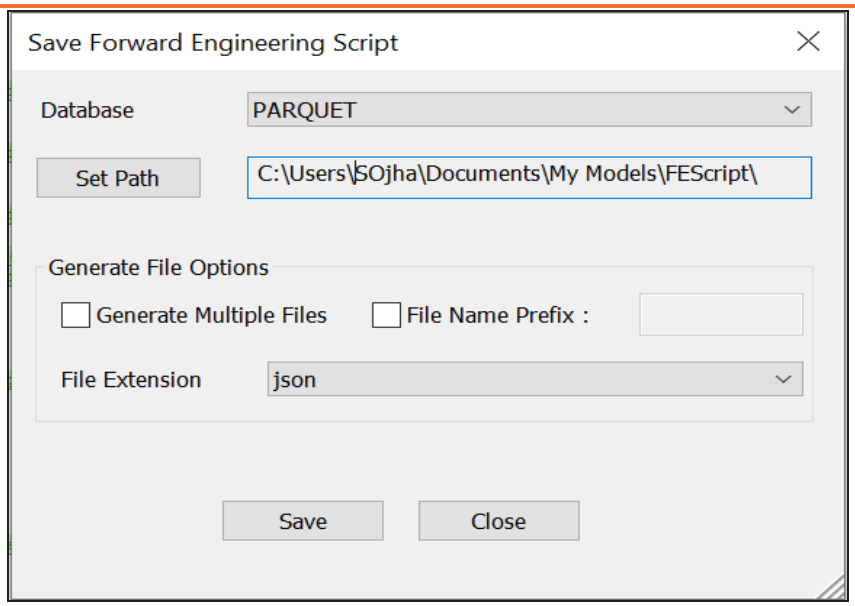
6. Click **Preview** to view the schema script.



Use the following options:

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

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



8. Use the following options:
   - **Set Path**: Use this option to set the location to save the generated forward engineering script file.
   - **Generate Multiple Files**: By default, a single forward engineering script file is created. Use this option to save the script into multiple files, grouped-by objects.
   - **File Name Prefix**: Use 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 in the JSON format at the configured location. You can open it in any text editor and verify.

Forward Engineering Options for Parquet

Following are the forward engineering options for Parquet.

Option Selection

Parameter	Description	Additional Information
Option Set	Specifies the option set template for forward engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save a configured option set. **Save As**: Use this option to save an option set either in the model or in the XML format at some external location. **Delete**: Use this option to delete an option set.
Database Template	Specifies the database template for controlling schema generation	**Browse**: Use this option to browse and select a database template. **Edit**: Use this option to edit a template in the Template Editor. **Reset**: Use this option to reset the Database Template option.
Script Option	Specifies the script option for the schema generation	**Pre-Script**: Indicates whether pre-scripts attached to the schema are executed **Post-Script**: Indicates whether the post-scripts attached to the schema are executed
Comments	Indicates whether comments are included in the schema	
Hive	Indicates whether the external table syntax for Hive is executed	

Record Filter

Parameter	Description	Additional Information
Records	Specifies the selected records for the schema generation	
Display either Logical Names or Physical Names		**Logical Names**: Indicates that only logical names of the records are included in the generated schema **Physical Names**: Indicates that only physical names of the records are included in the generated schema **Physical Names, show owner**: Indicates that physical names and owners of the records are included in the generated schema **Physical Names, show owner using User**: Indicates that the physical names and owners of the records are included in the generated schema. Owners of the records are displayed using User.
Select all of the items in the list	Use this option to select all the records in the list.	
Unselect all of the items in the list	Use this option to unselect all the records.	
Select all unselected items, and unselect all selected items	Use this option to select all the unselected records and unselect all the previously selected records.	

Preview

Parameter	Description	Additional Information
Viewer	Displays the schema in	**Collapse All**: Use this option to collapse all the nodes.

	the viewer editor	**Search**: Use this option to search a text entered in the search box. **Find Previous**: Use this option to navigate to previous search string in the search results **Find Next**: Use this option to navigate to next search string in the search result.
Text	Displays the schema in the text editor	**Save**: Use this option to save the generated schema. **Search**: Use this option to search through the generated schema. **Print**: Use this option to print the generated schema. **Replace**: Use this option to find and replace text in the generated schema. **Copy**: Use this option to copy the selected text in the schema. **Text Options**: Use this option to edit window settings, fonts, and syntax color. **Git**: Use this option to commit the FE script to a Git repository.

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.

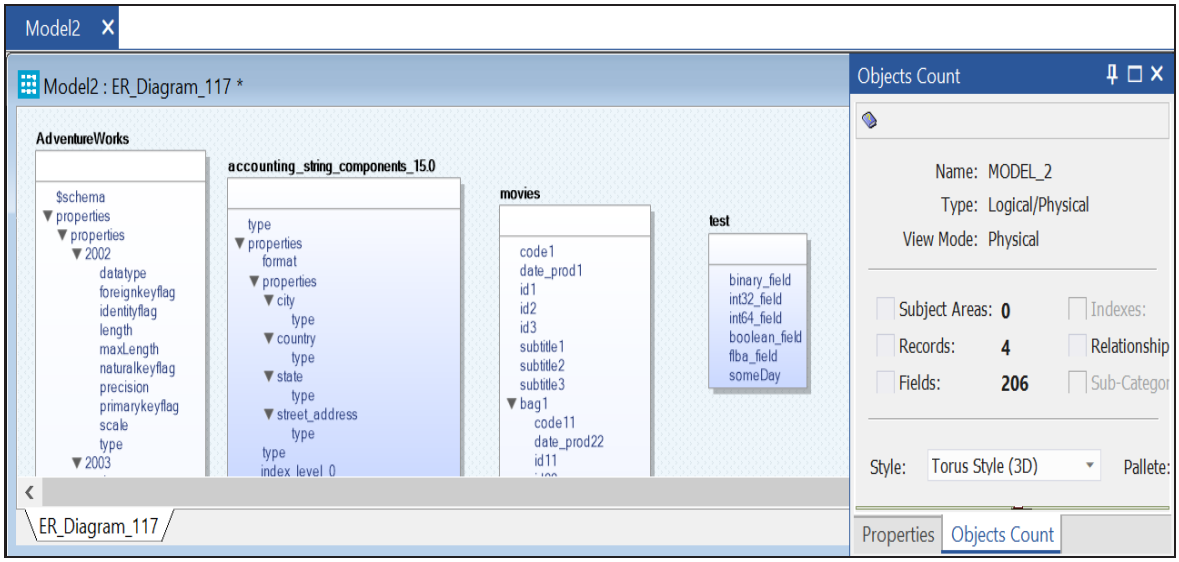
This topic walks you through the steps to compare a Parquet model with script.

To compare models with script:

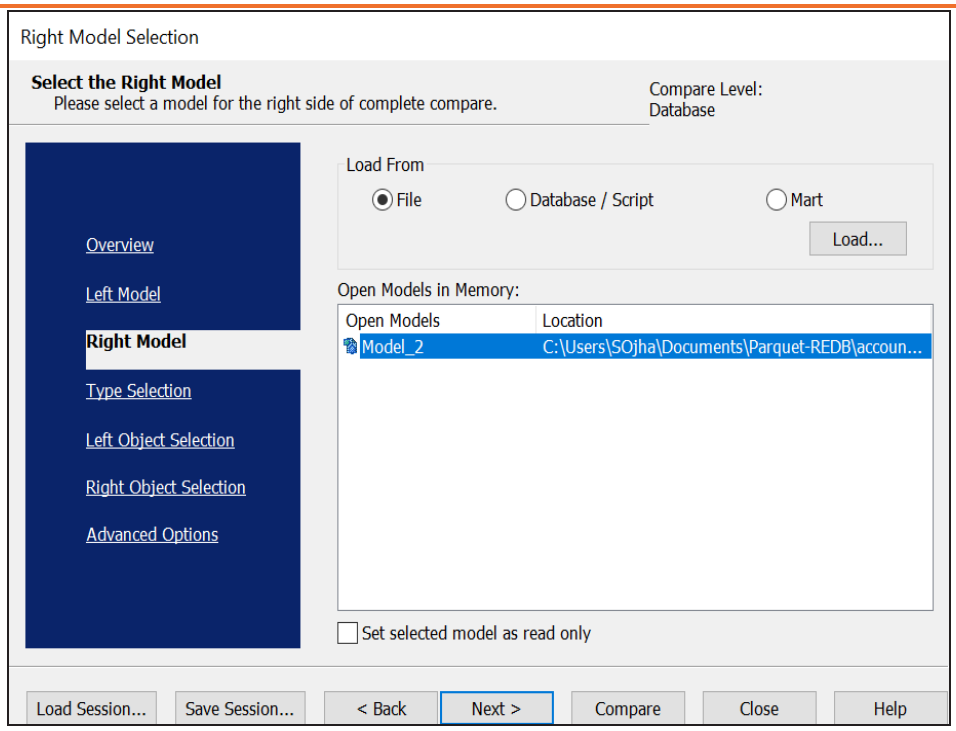
1. Open your Parquet model.

   Ensure that you are in the Physical mode.

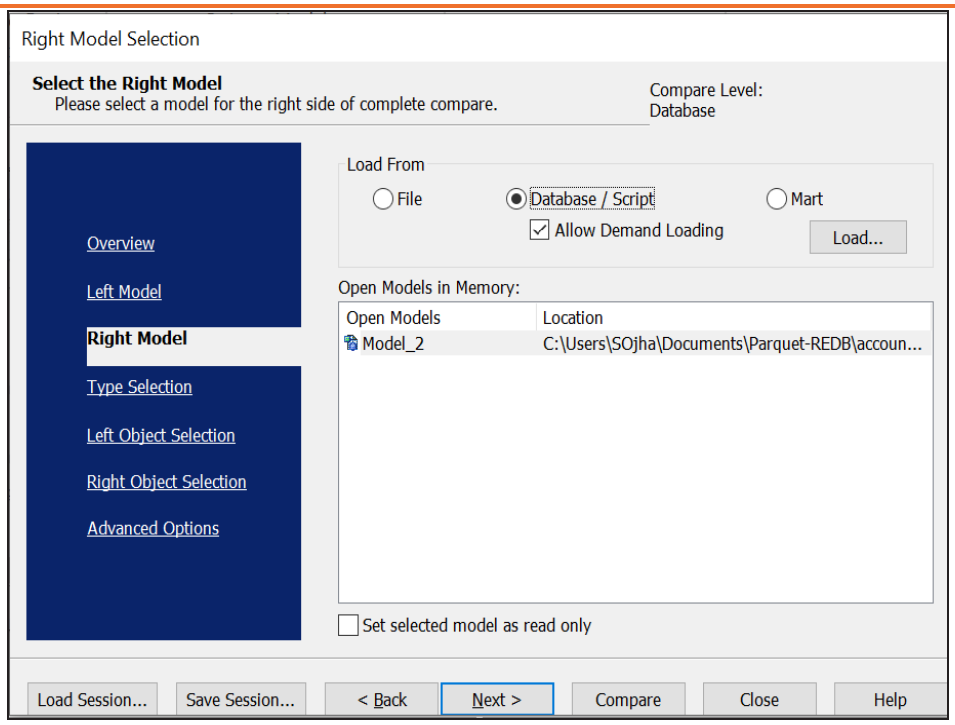
   For example, the following image uses a Parquet model with four records.



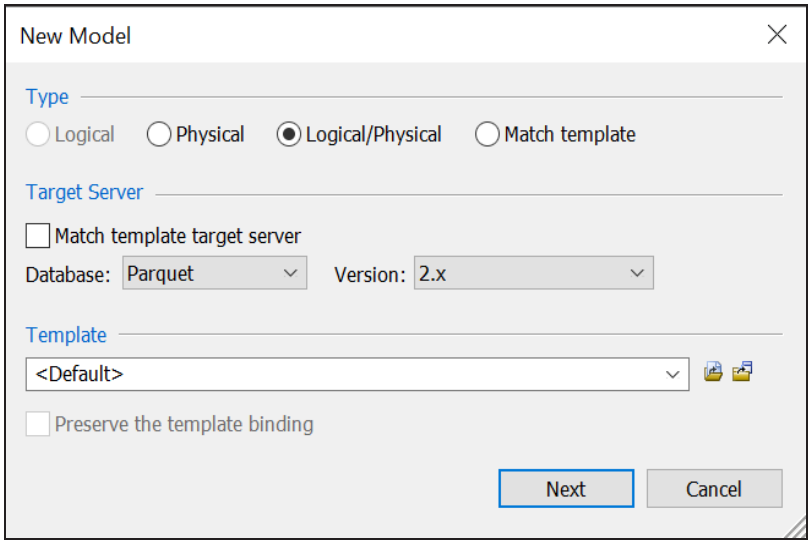
2. Click **Actions** > **Complete Compare**.
   By default, the Complete Compare wizard assigns the open model as the Left Model. Hence, the Right Model tab appears.



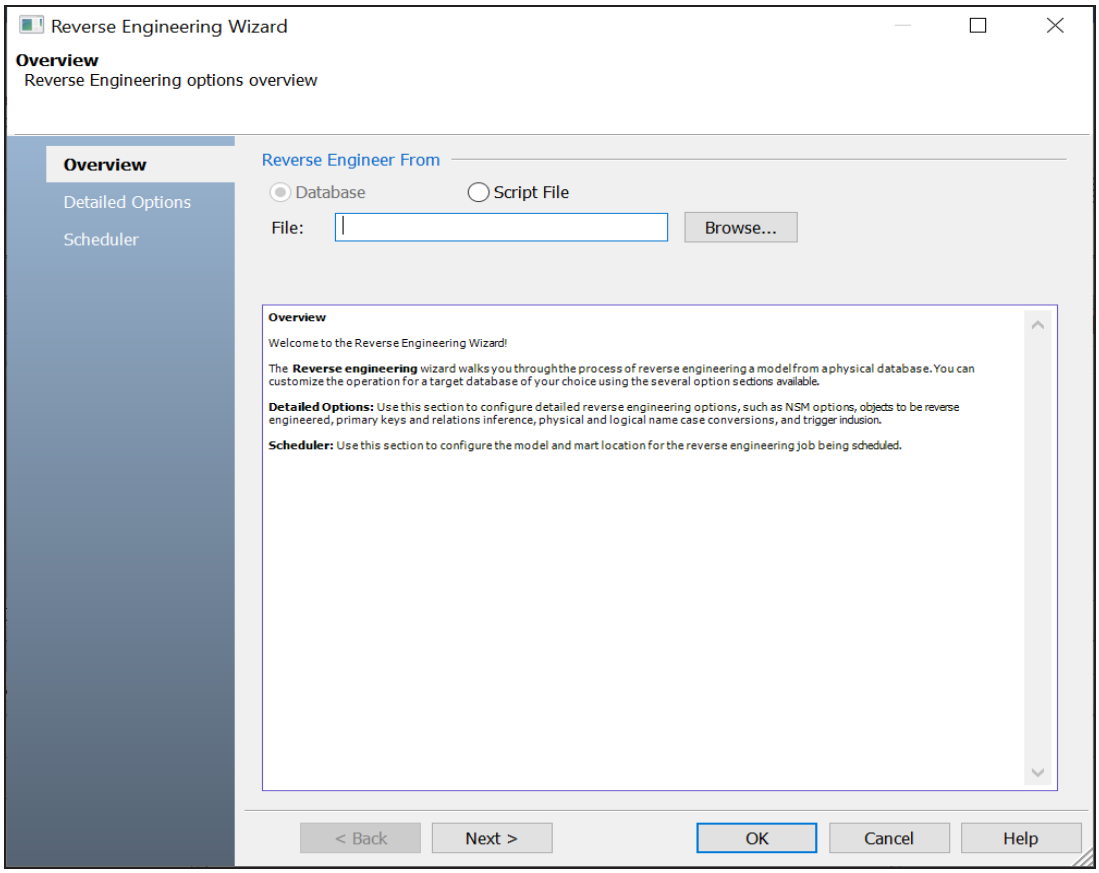
3. Click **Database/Script**.
   By default, the Allow Demand Loading option is selected.



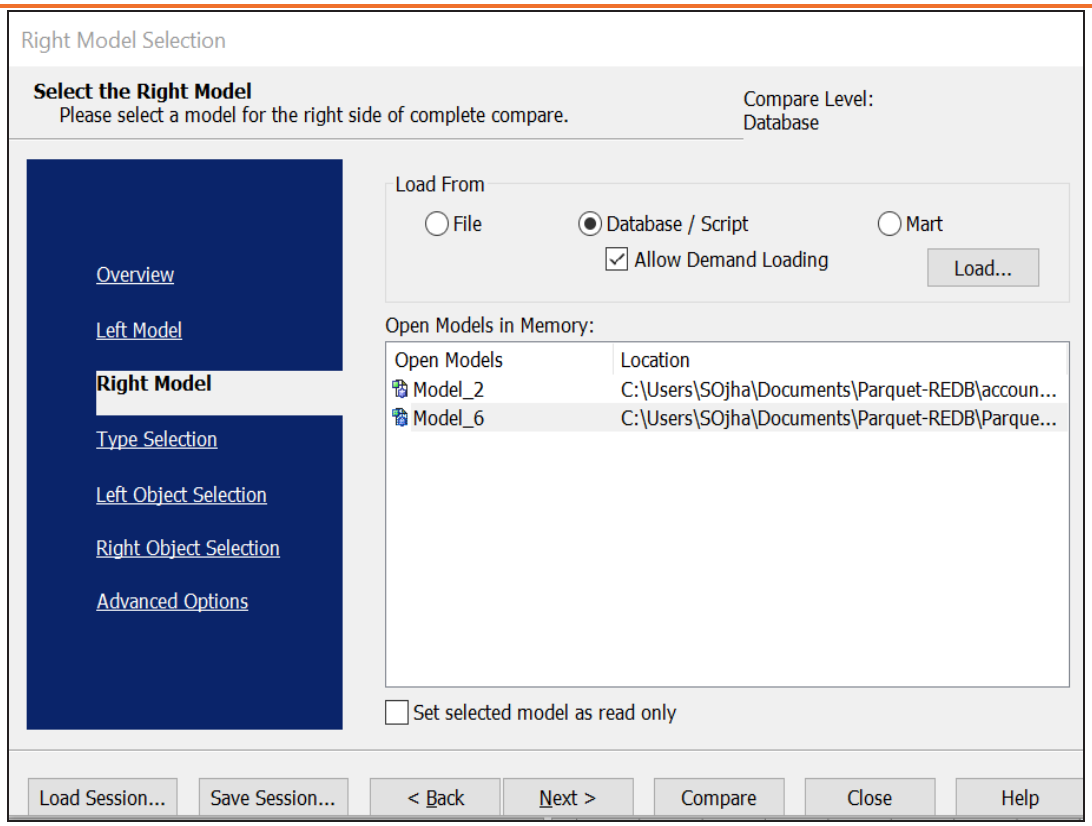
4. Click **Load**.
   The New Model dialog box appears. This starts the reverse engineering process to pull a model from the script to compare.



5. Ensure that the Database is set to Parquet. Then, click **Next**.
   The Reverse Engineer Process Wizard appears.

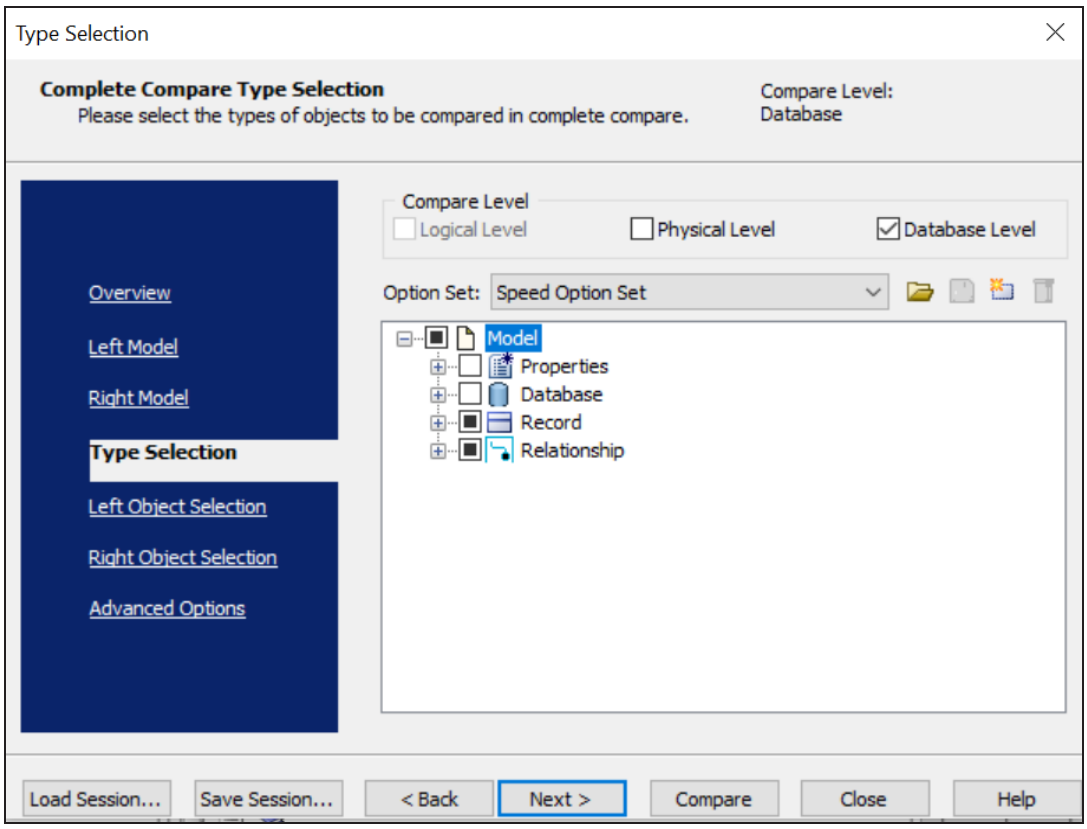


6. Click **Script File**. Then, browse and select a script file.
7. Click **OK**.
   The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse engineered.



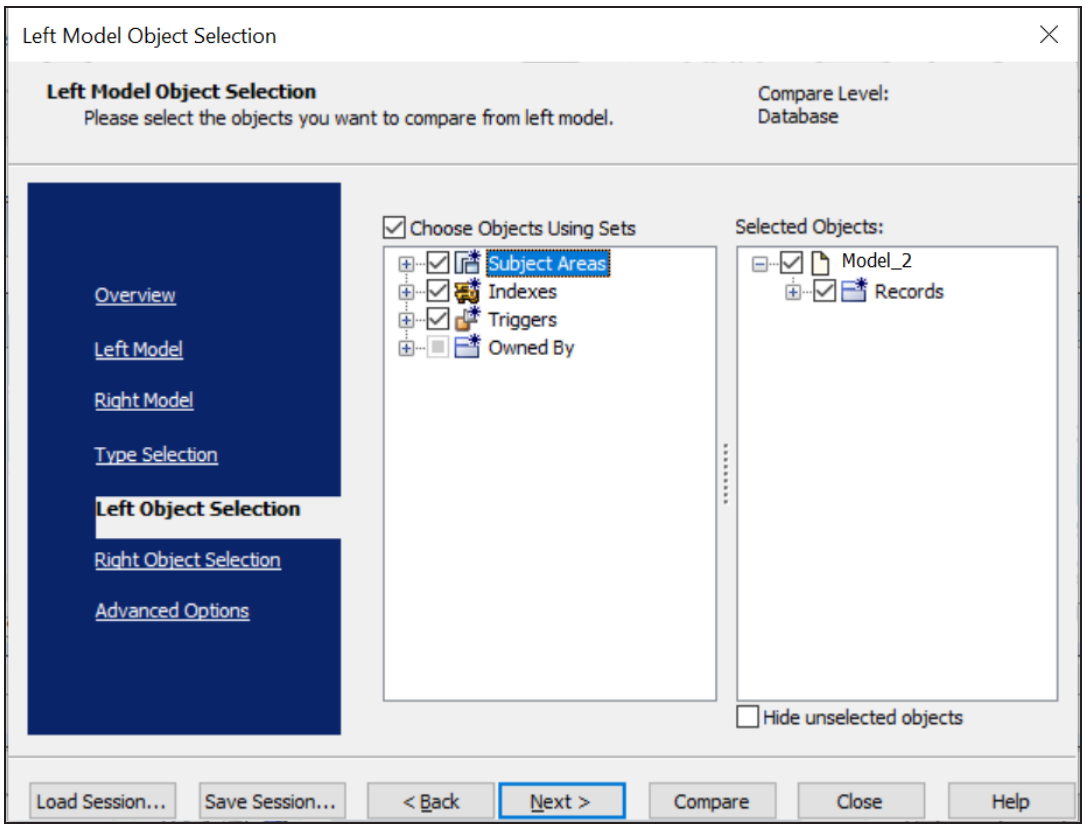
8. Click **Next** and on the Type Selection tab, select the appropriate options.

For example, the following image shows the default options.



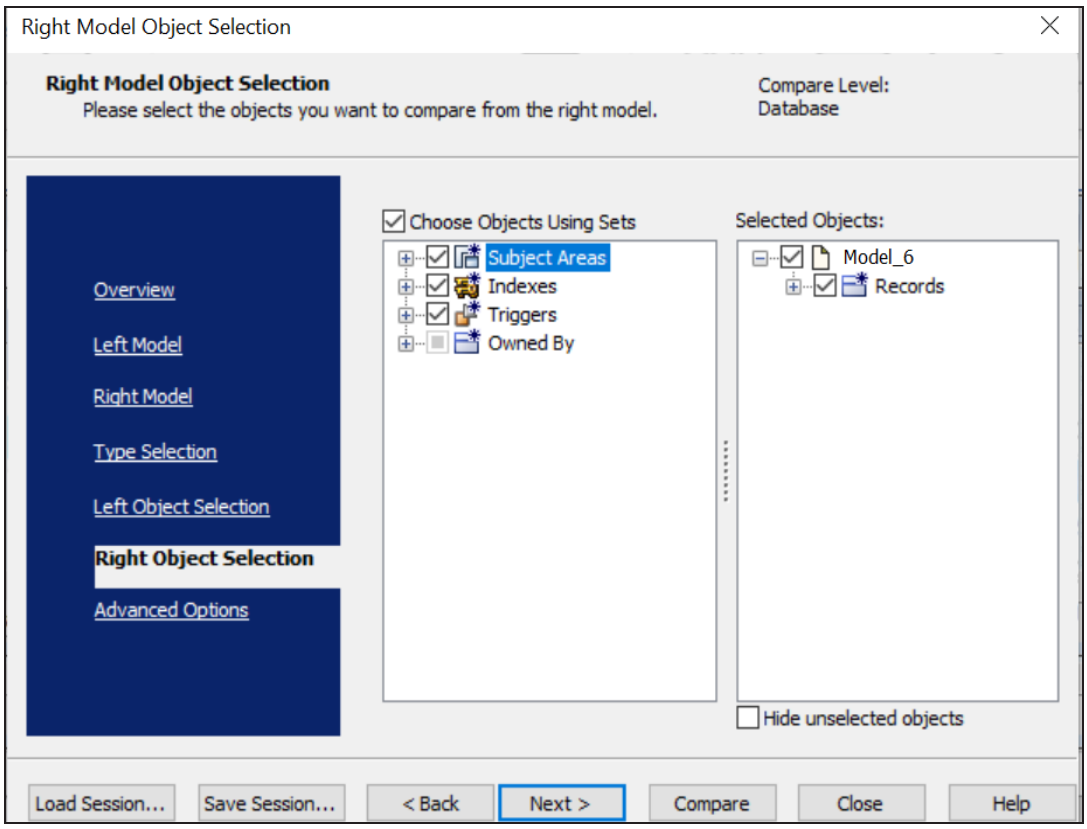
9. Click **Next** and on the Left Object Selection tab, select the appropriate options.

For example, the following image shows the default options.

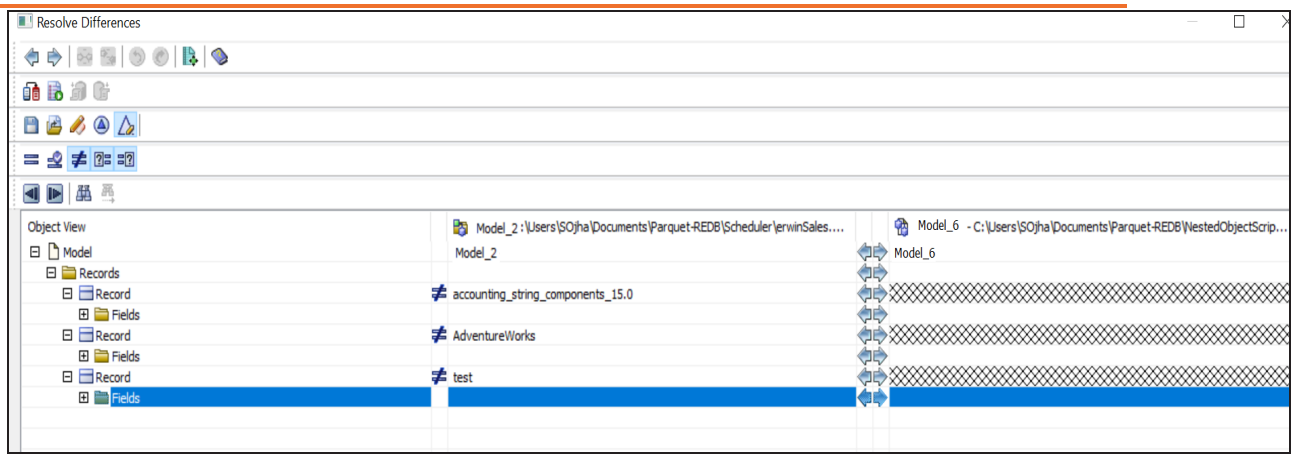


10. Click **Next** and on the Right Object Selection tab, select the appropriate options.

For example, the following image shows the default options.



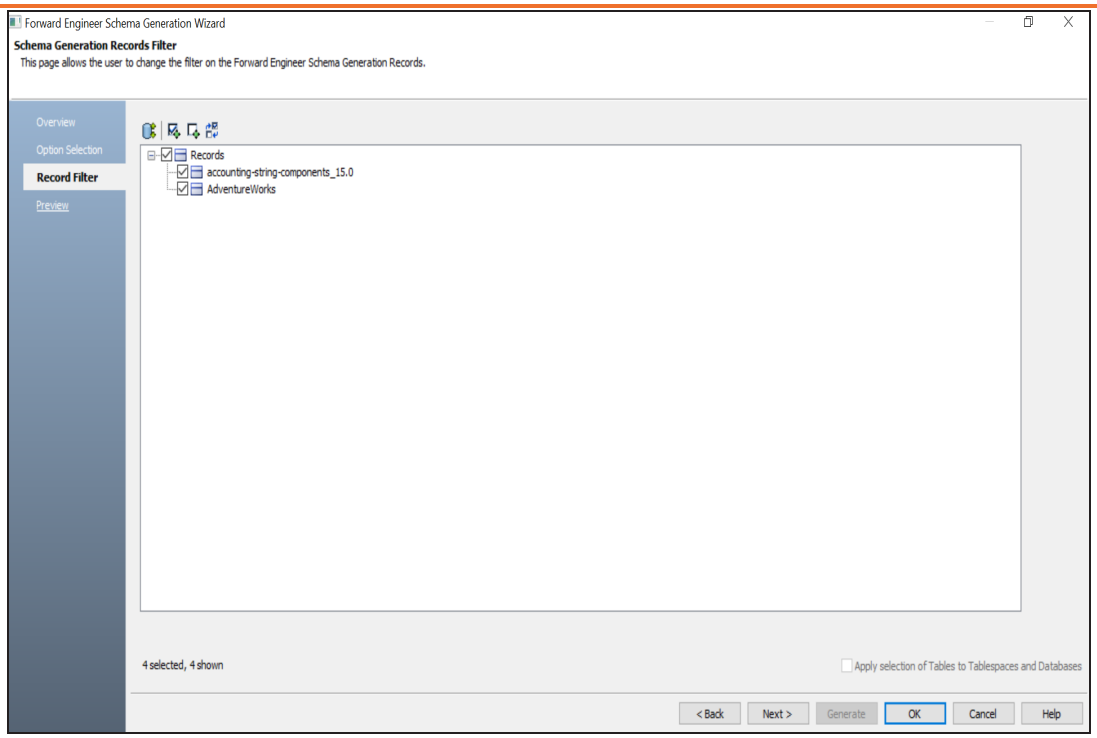
11. Click **Compare**.
    The comparison process runs, and the Resolve Differences dialog box appears. It displays the differences between your model and script.
    For example, the following image shows that the AdventureWorks record is available in your model but not in the script.



Select the AdventureWorks record and click . This will move the AdventureWorks record to the right model. Similarly, resolve other differences.

12. As differences were moved to the right model, click .
    This opens the Forward Engineering Alter Script Schema Generation Wizard.

13. Click **Record Filter** and select or verify the records to be included on the forward engineering script.



14. Click **Preview** to view and verify the alter script.
15. Click **Generate**.
    The forward engineering process starts. The script generates your physical database schema. You can verify the newly generated schema and save it.
16. Click **OK**. Then click **Finish**.
    This closes the Resolve Differences dialog box and displays the Complete Compare wizard.
17. Click **Close**.

Migrating Relational Models to Parquet Models

You can convert and migrate your relational models to Parquet 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 Parquet model.

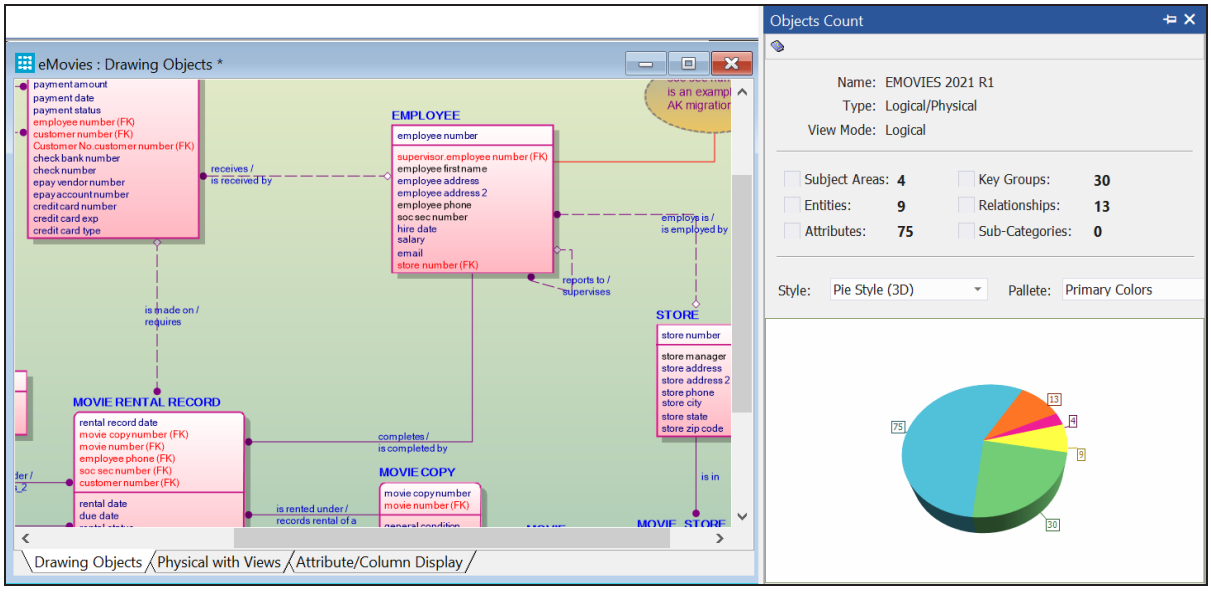
Migration by Changing the Target Database

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

1. Open your relational model in erwin Data Modeler (DM).

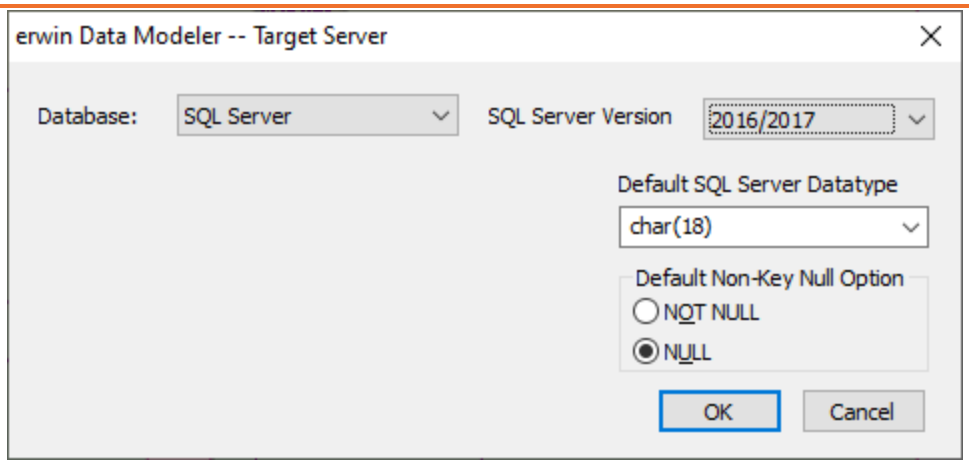
   Ensure that you are in the Physical mode.

   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.

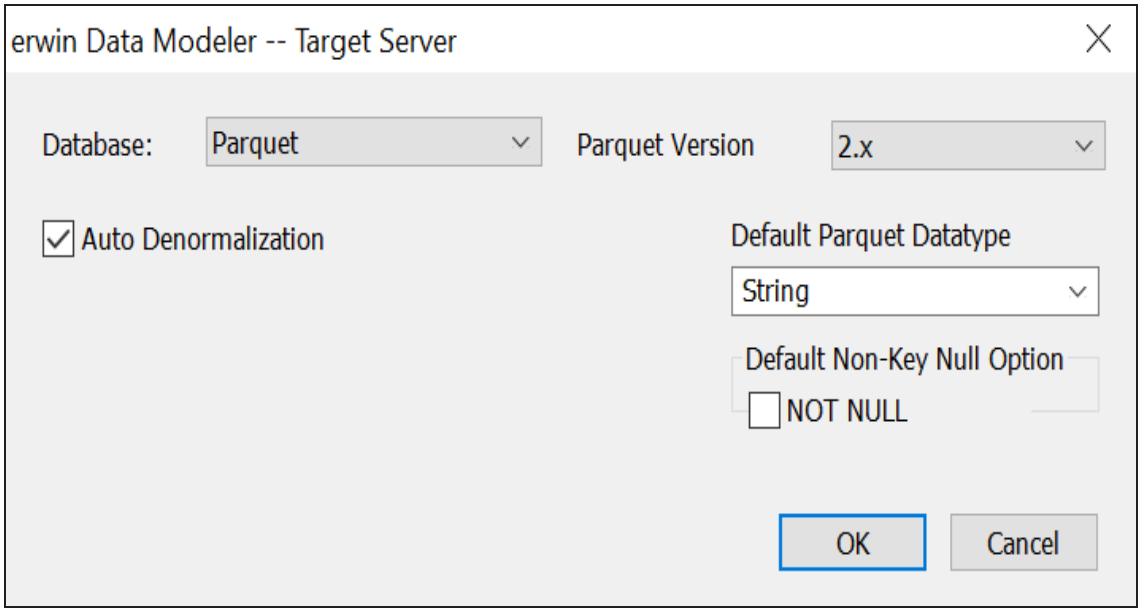


2. On the ribbon, click **Actions** > **Target Database** or on the status bar, click the database name.

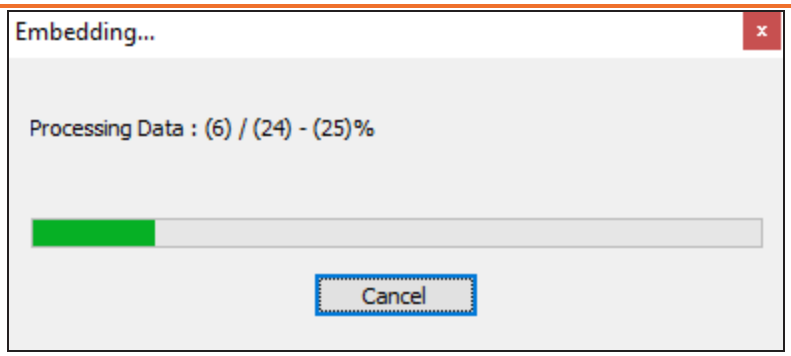
   The erwin Data Modeler -- Target Server screen appears.



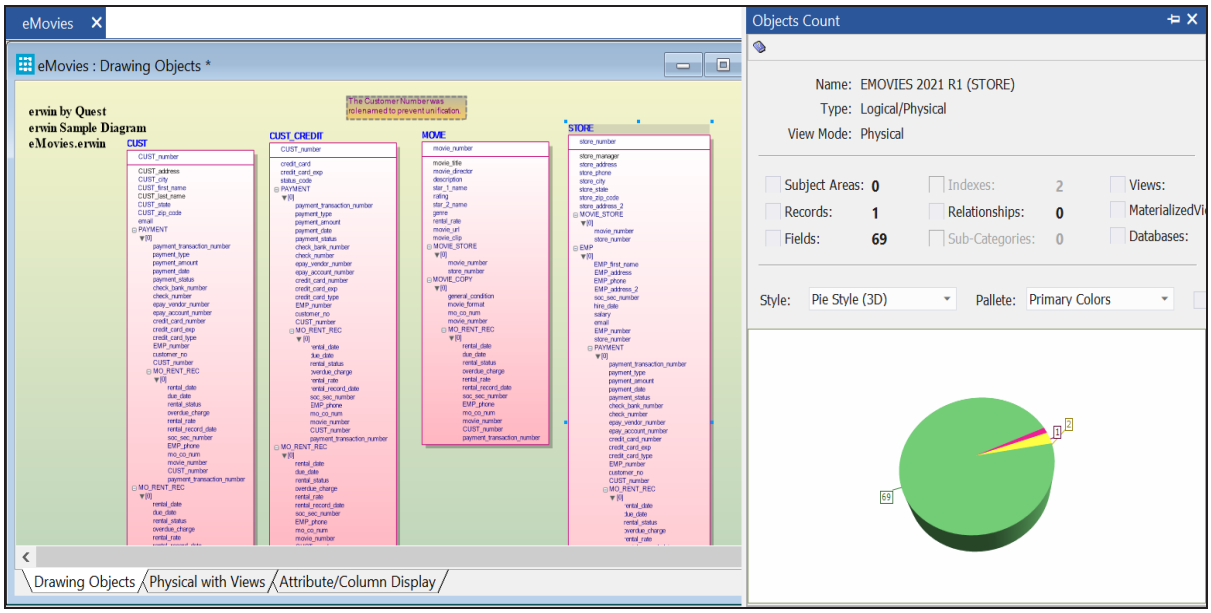
3. In the **Database** drop-down list, select MongoDB.
   By default, the Auto Denormalization check box is selected. Keep it selected.



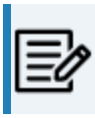
4. Click **OK**.
   The conversion process starts.



Once the conversion is complete, the existing model is migrated to a Parquet model.



In the **Objects Count** pane, note that instead of tables and columns, we now have fields and records. 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.



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

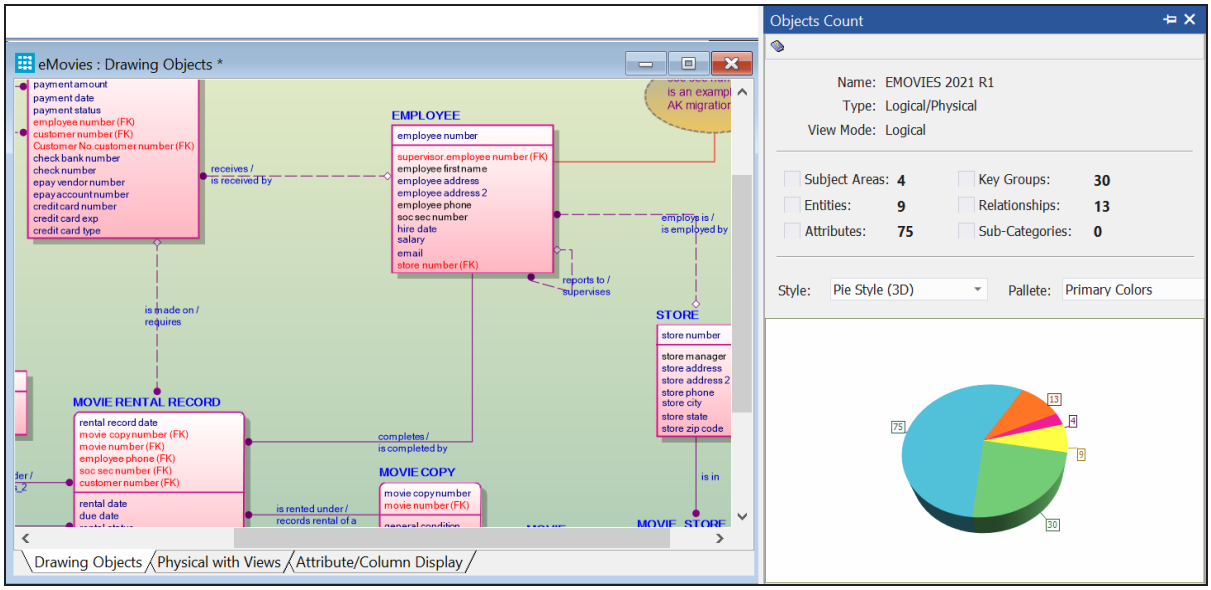
Migration by Deriving a Model

To migrate by deriving a model, follow these steps:

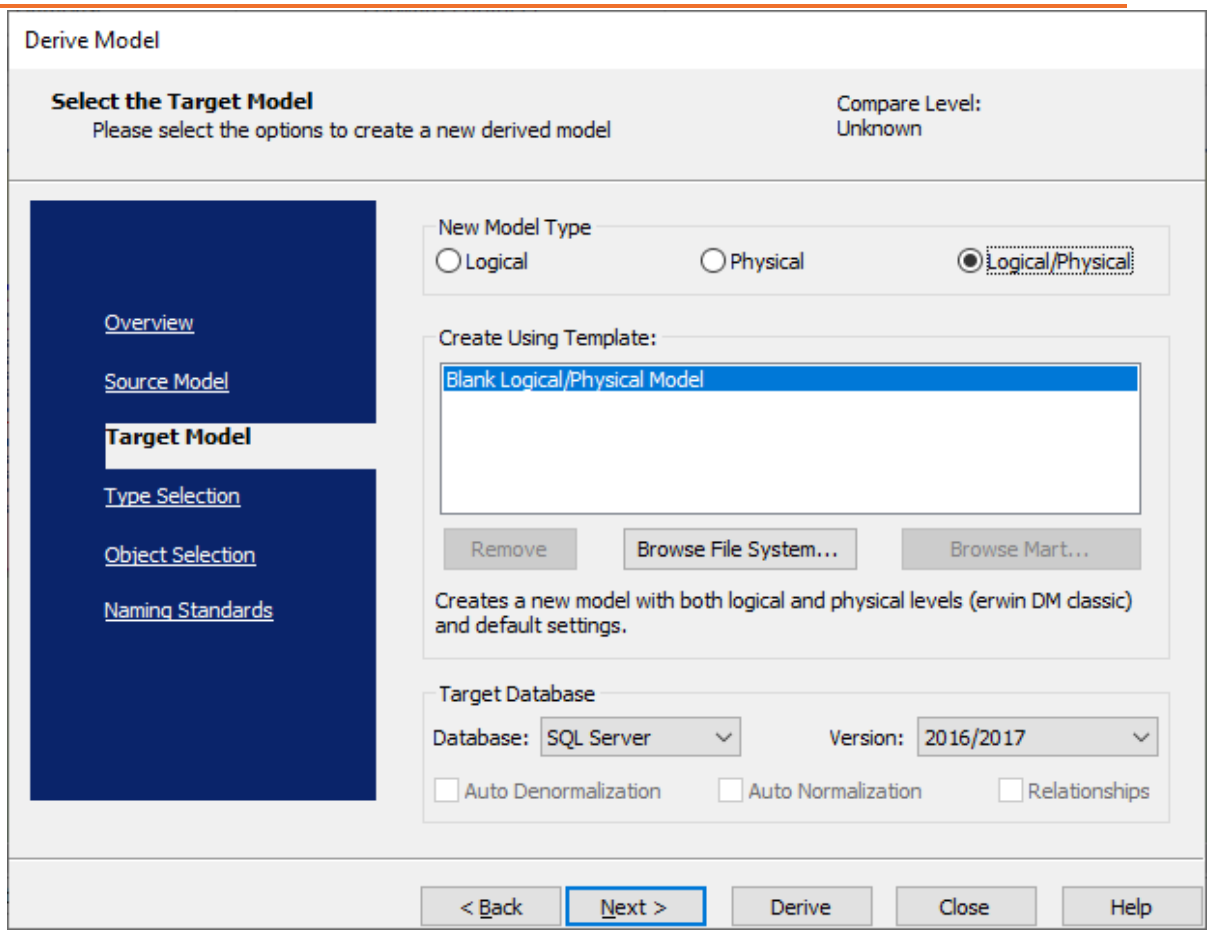
1. Open your relational model in erwin Data Modeler (DM).

   Ensure that you are in the Physical mode.

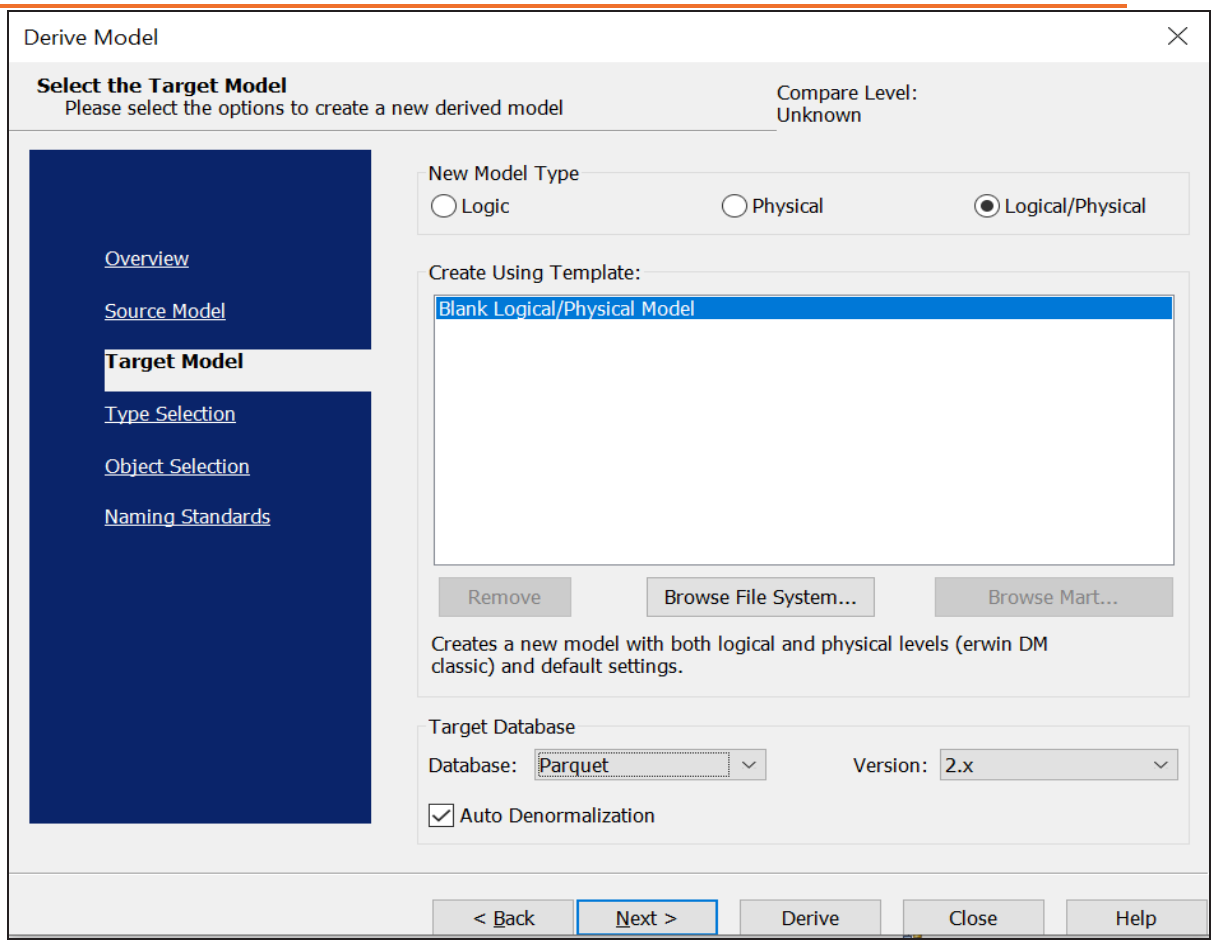
   For example, the following image uses the sample eMovies.erwin model. In the **Objects Count** pane, note the number of tables, columns, and relationships.



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.



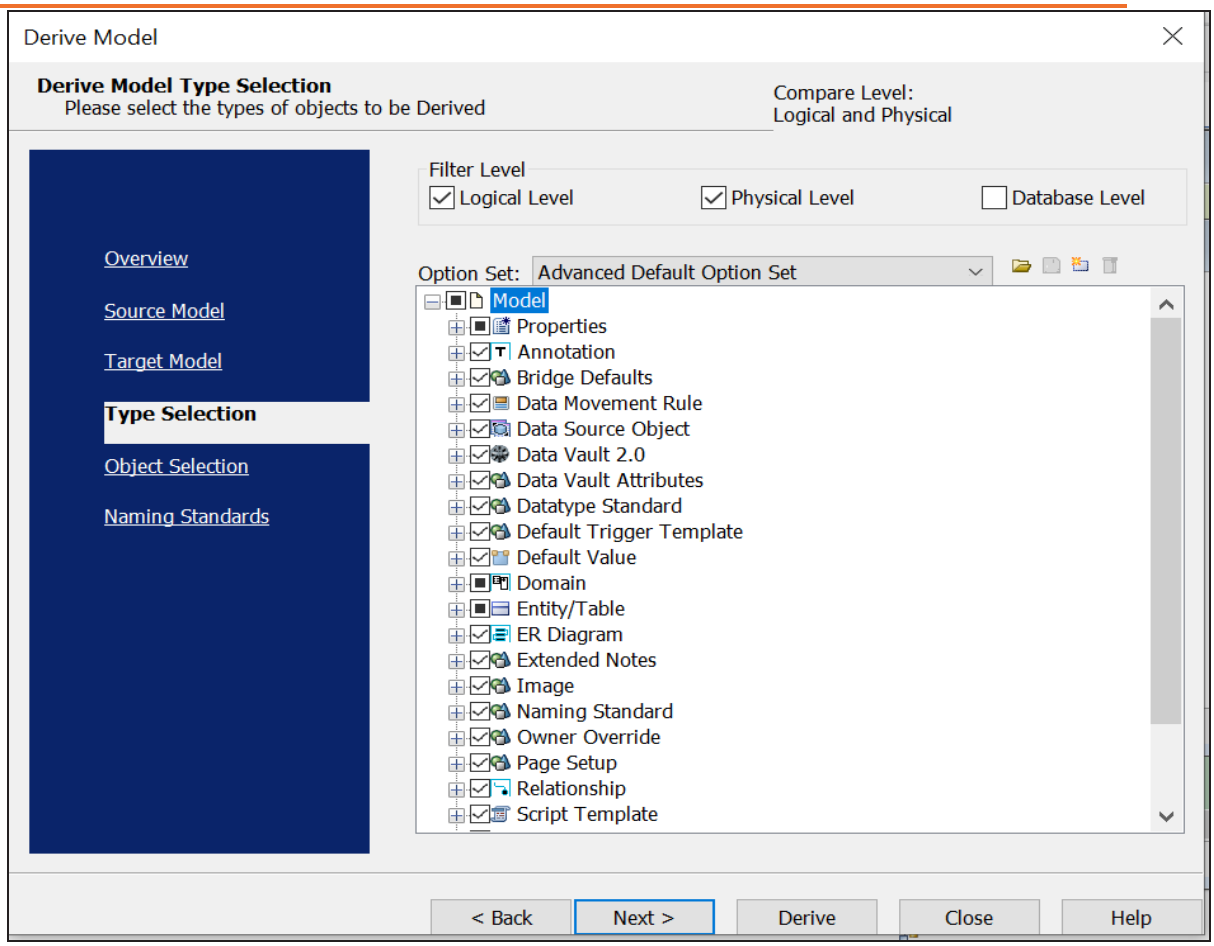
3. In the **Database** drop-down list, select **Parquet**.
   By default, the Auto Denormalization check box is selected. Keep it selected.



4. Click **Next**.

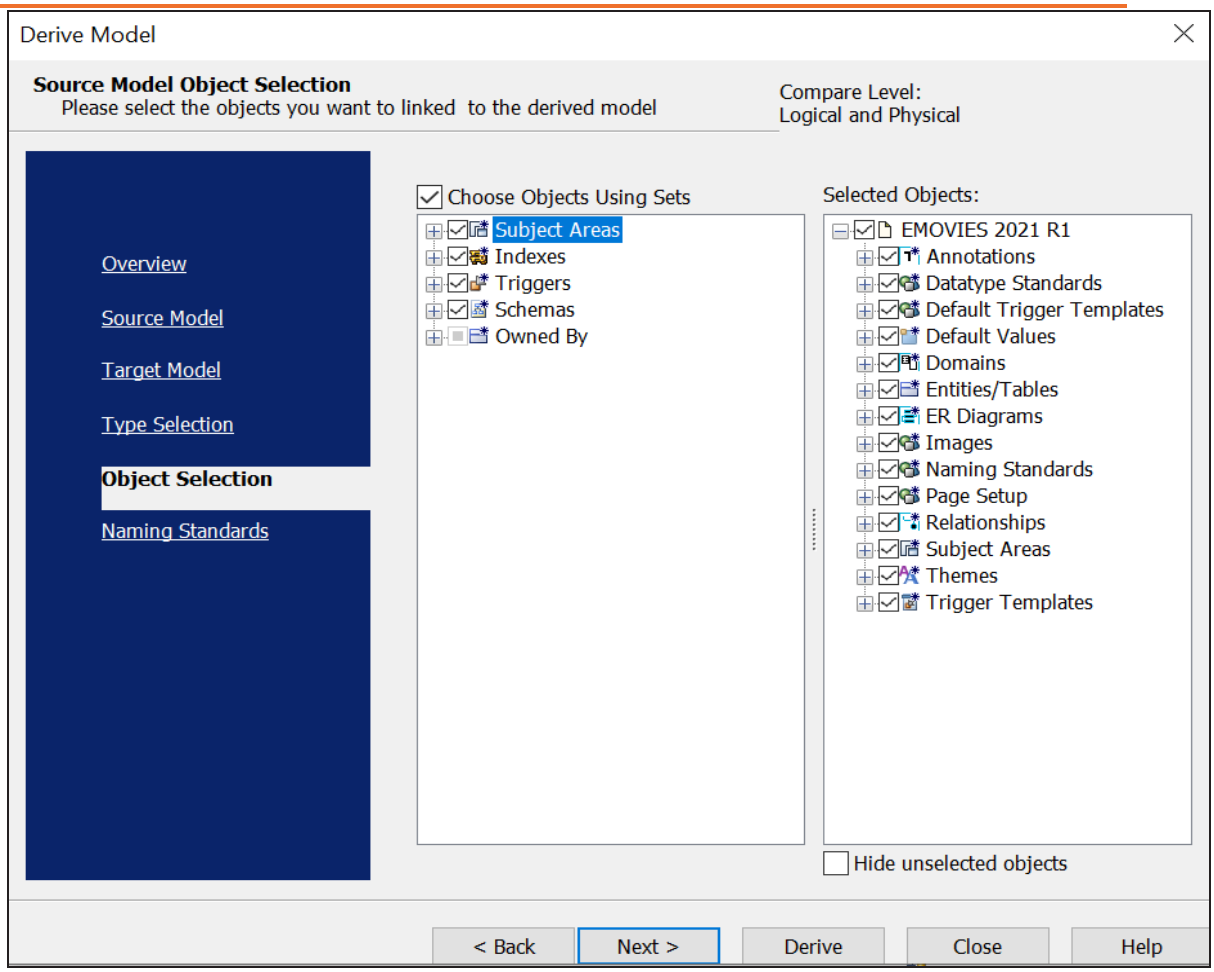

If the Type Resolution screen appears, click **Finish**.

The Type Selection section appears.

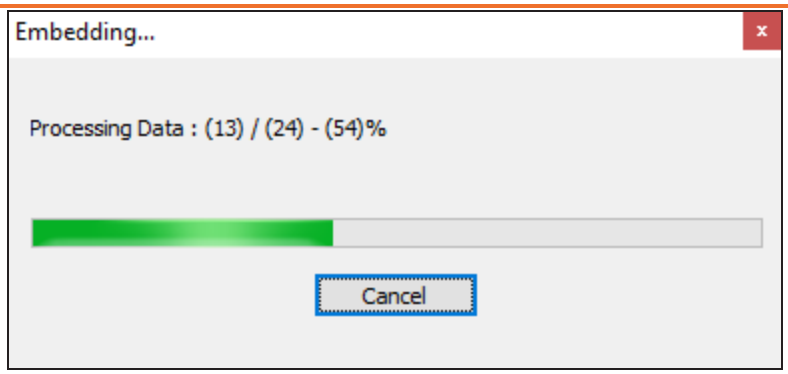


5. Select the types of objects that you want to derive into the target Parquet model.

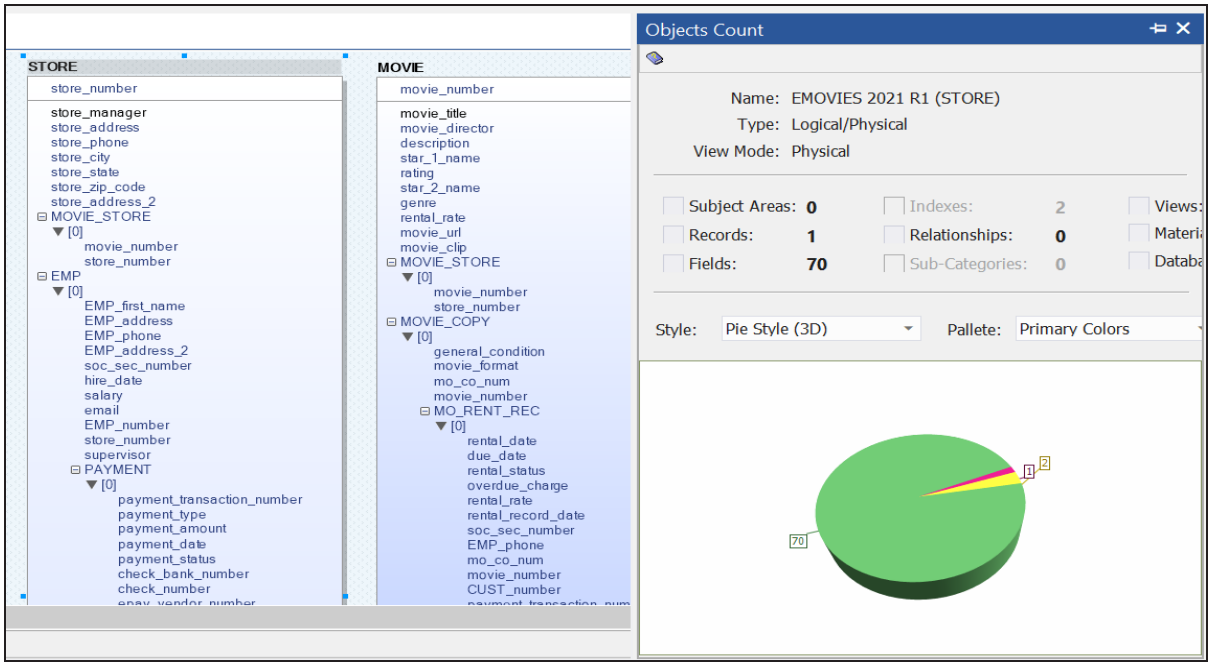
6. Click **Next**.
   The Object Selection section appears. Based on the object types you selected in step 5, it displays a list of objects.



7. Select the objects that you want to derive into the target Parquet 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 records 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.

Couchbase Support

erwin Data Modeler (DM) now supports Couchbase 7.x as a target database. Apart from the existing objects this implementation supports the following new objects:

- Collection
- Function
- Scope

Central Scheduler

Starting erwin Data Modeler 12.0, you can now use erwin DM Scheduler to schedule reverse engineering jobs centrally on a local or a remote instance of erwin Data Modeler. You can configure multiple remote machines as servers and set up jobs to run in parallel on these servers. The Central Scheduler saves time and provides you with an improved performance by distributing reverse engineering jobs across multiple servers.

Apart from this, in case of models on erwin Mart, you can now run the Complete Compare process as part of reverse engineering. This enables you to compare the reverse engineering result with the model in your mart. In case of differences, you can save the updates as the latest version of your model in the mart.

The features introduced in this release are:

- Scheduling Remote Jobs
- Running Complete Compare
- Productivity and UI Enhancements

Scheduling Remote Jobs

You can schedule reverse engineering jobs on a remote server using the scheduler. Before scheduling a remote job, ensure that you configure your remote server.

Configuring Remote Server

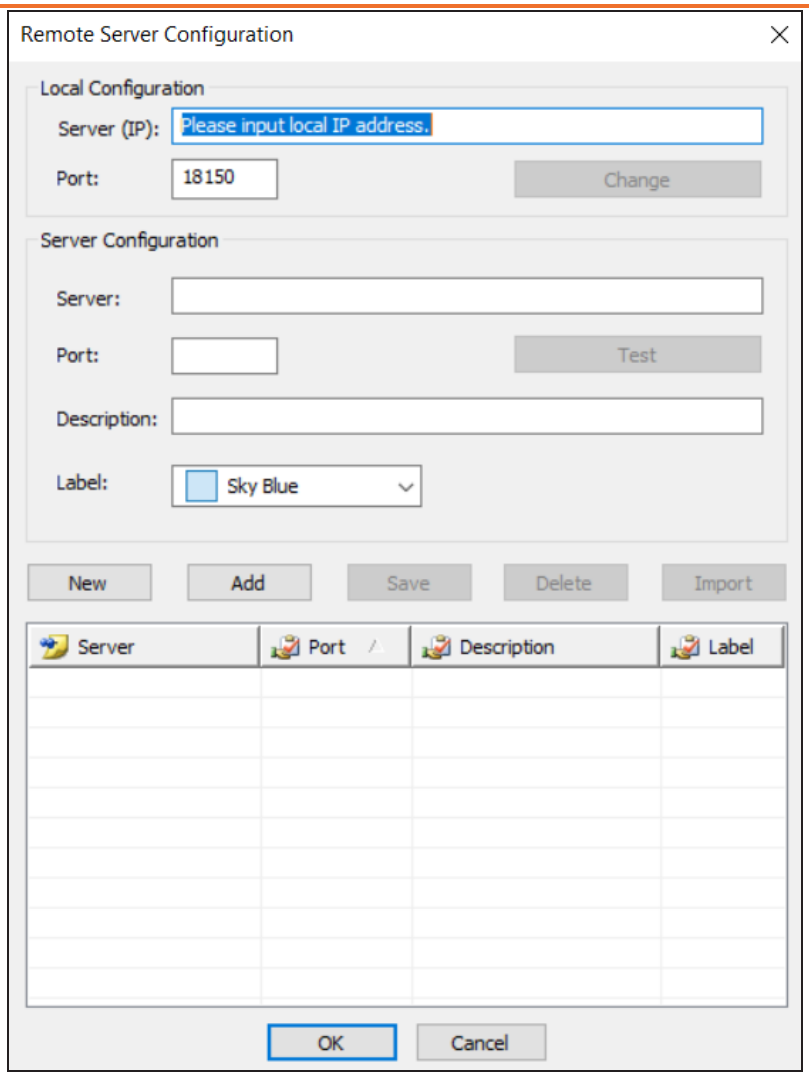
You can use erwin DM Scheduler to schedule reverse engineering (RE) jobs on a remote instance of erwin Data Modeler. You can configure multiple remote machines as servers and set up jobs to run in parallel on these servers. This saves time and provides you with an improved performance by distributing reverse engineering jobs across multiple servers.

To use remote servers to run RE jobs, ensure that:

- the remote server is running
- the remote server configuration is set up on your local Scheduler
- the local server configuration is set up on your remote Scheduler

To set up remote server configuration, follow these steps:

1. On the ribbon, go to **Remote** > **Remote Server Configuration**.
   The Remote Server Configuration screen appears.

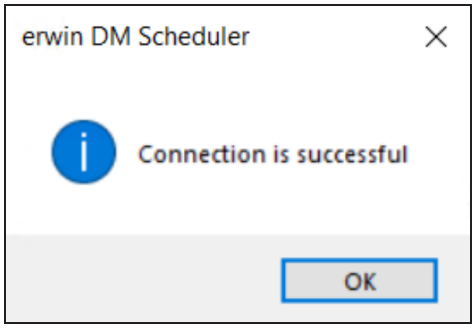


2. Configure the remote server based on your requirement. Refer to the following table for field descriptions.

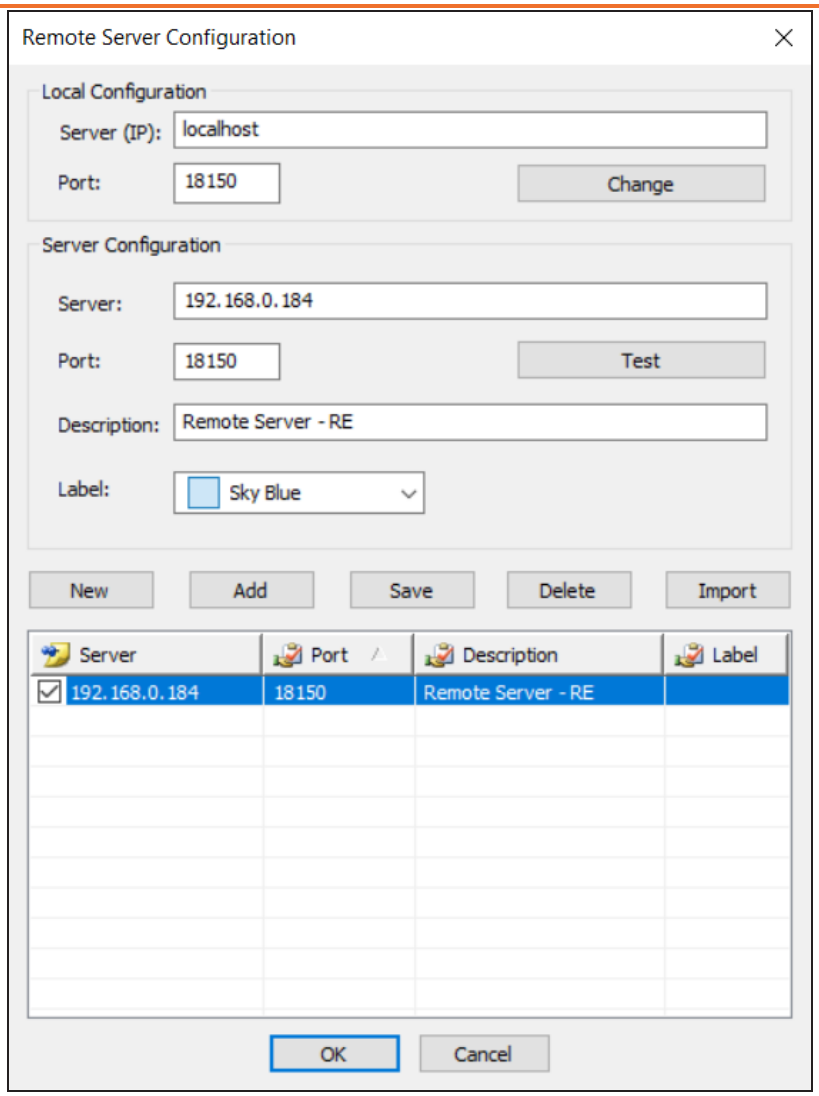
Section	Option	Description
Local Configuration	Server (IP)	Specifies the IP address of the local host.
	Port	Specifies the service port number for the remote scheduler. This field displays the default port number. Click **Change** to update the port number.

Section	Option	Description
Server Configuration	Server	Specifies the IP address of the remote server.
	Port	Specifies the port number for remote server. Ensure that the remote server is running before testing the connection.
	Description	Specifies the description for the remote server.
	Label	Specifies the label color to categorize the server configuration.

3. Once you have added remote server configuration, click **Test**.
   The erwin DM Scheduler screen appears on a successful connection.



4. Click **Add**.
   The remote server configuration is added to the list of remote servers.



Once you have set up a remote server configuration, use one of the following options:

- **New**: Use this option to set up another remote server. Selecting this option resets any information entered on the screen.
- **Save**: Use this option to save any changes to selected remote server configuration.
- **Delete**: Use this option to delete any selected remote server configurations.

- **Import**: Use this option to import an existing remote server configuration. Select a server and click **Import**. This imports the This option is available when server information is configured under Server Configuration section.

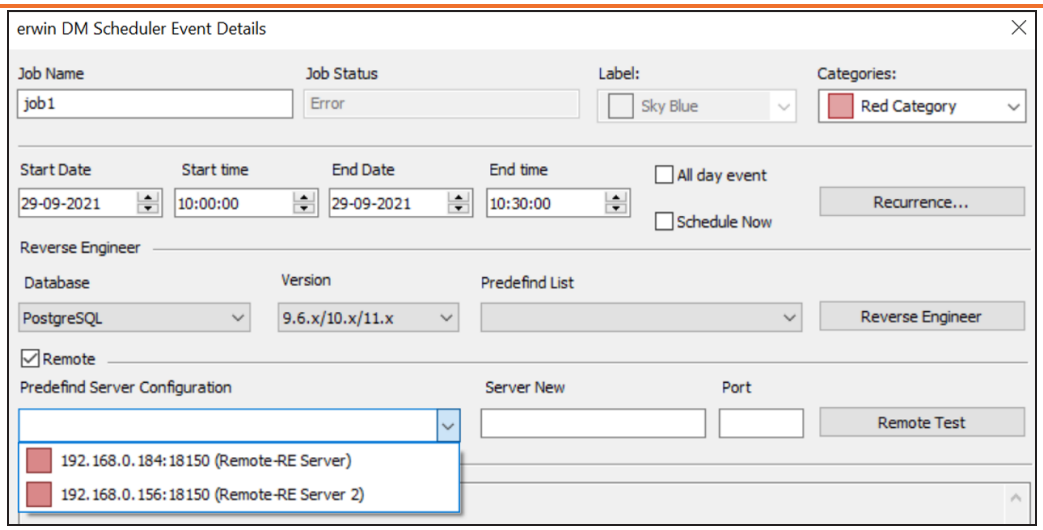
  The import replaces the existing server configuration with the latest configuration.

5. Once you have set up a remote server configuration, use one of the following options:
   - **New**: Use this option to set up another remote server. Selecting this option resets any information entered on the screen.
   - **Save**: Use this option to save any changes to selected remote server configuration.
   - **Delete**: Use this option to delete any selected remote server configurations.
   - **Import**: Use this option to import an existing remote server configuration. Select a server and click **Import**. This imports the remote server and replaces the existing remote server configurations.

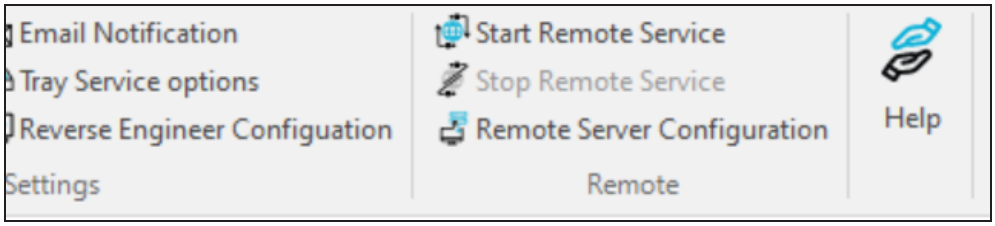
     The import replaces the existing server configuration with the latest configuration.

   Click **OK**.
   Remote server configuration is saved and is available in the Predefined Server Configuration list on the erwin DM Scheduler Event Details screen.



Once you have set up remote servers, to use them, on the ribbon, in the Remote group, click either of the following options:

- **Start Remote Services**: Use this option start a remote service.
- **Stop Remote Services**: Use this option to stop a remote service.



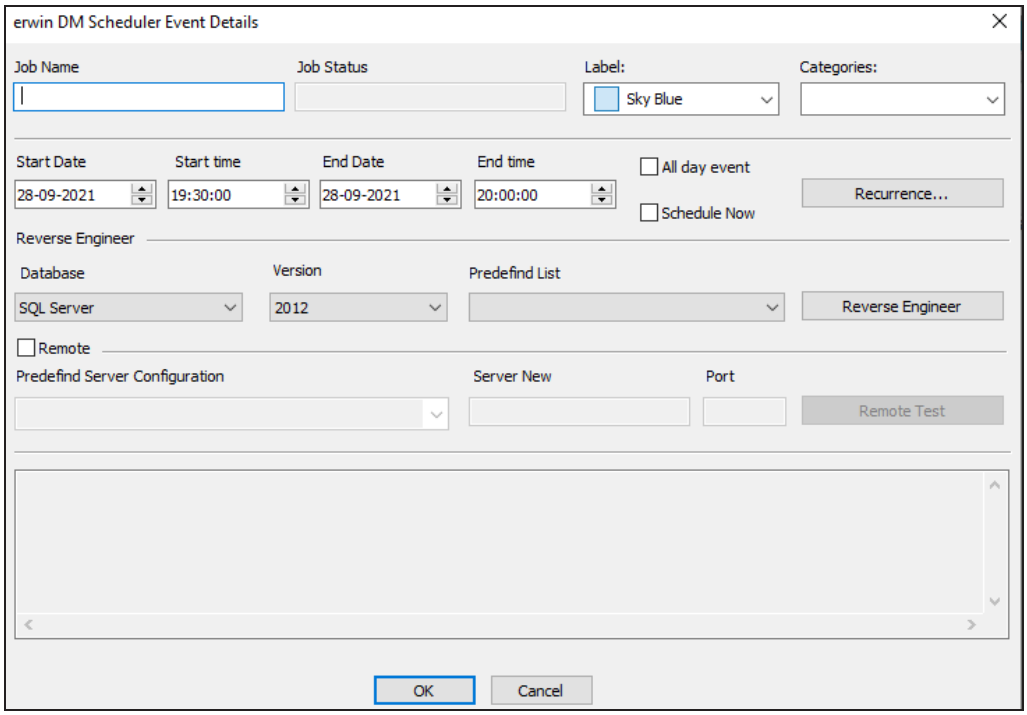
Scheduling Remote Jobs

Before scheduling a remote job, ensure that you start scheduler services and remote services on both, local and the remote instances of erwin DM Scheduler:

- To start a service, on the ribbon, under the Home tab, click **Start Service** option in the Services group.
- To start a remote service, on the ribbon, under the Home tab, click **Start Remote Service** option in the Remote group.

To schedule remote reverse engineering (RE) jobs, do the following:

1. Create an event in one of the following ways:
   - On the ribbon, under the Home tab, click **New**.
   - In the Calendar view, double-click a time slot under the day of your choice.
   - In the Calendar view, right-click a time slot under the day of your choice and click **Add new event**.
2. The erwin DM Scheduler Event Details page appears.



3. On the erwin DM Scheduler Event Details page, configure the following options.

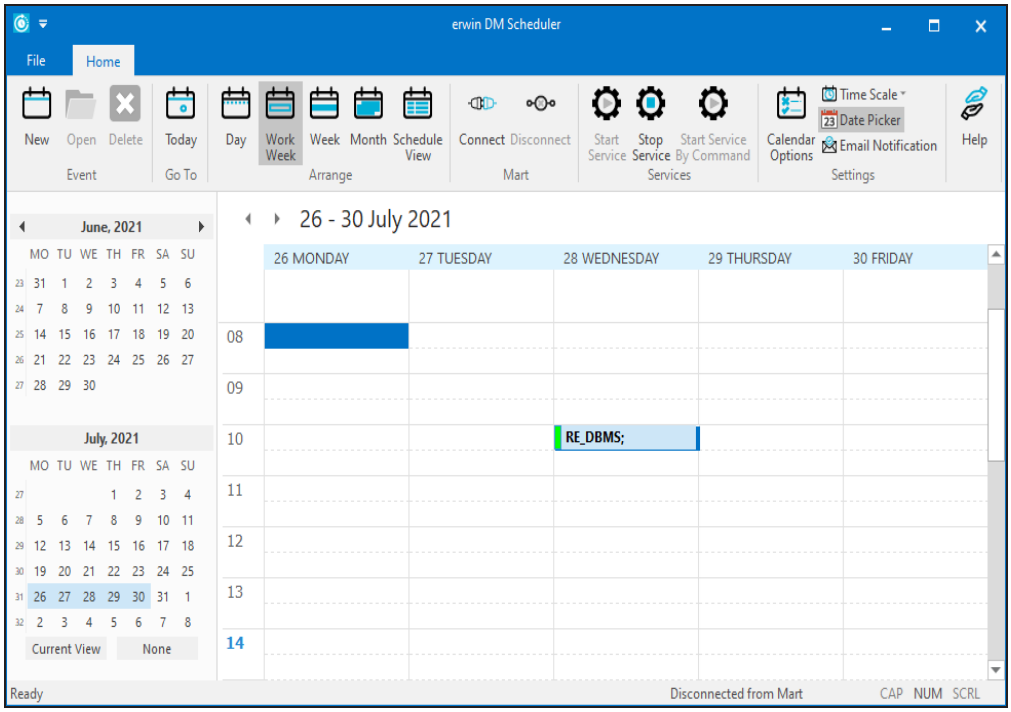
Option	Description	Additional Information
Job Name	Specifies the name of the job	
Job Status	Displays the status of the job	
Label	Specifies the color of the job label	

Option	Description	Additional Information
Start Date	Specifies the start date for a job	▪ Jobs are run serially. Hence, schedule a reasonable job duration. Ensure that you consider the DB, its size, and the approximate job duration of the current jobs, and then schedule a new job accordingly. ▪ Also, in case of multiple jobs scheduled at the same time with the Schedule Now option, it randomly selects a job to run. Therefore, it is recommended that you do not schedule multiple jobs to run at the same time.
Start Time	Specifies the start time for a job	
End Date	Specifies the end date for a job	
End Time	Specifies the end time date for a job	
All day event	Indicates whether it is an all day event	Selecting this option disables the Start Time and End Time options.
Schedule Now	Indicates whether to schedule the job now	Selecting this option disables the Start Time, Start Date, End Time, and End Date options and schedule the job now.
Recurrence	Specifies whether to schedule a job on recurrence bases for the jobs that you do repeatedly. This opens the Scheduling Recurrence page.	To schedule a recurrence job, refer to the Setting Recurrence topic.

Option	Description	Additional Information
Database	Specifies the database for reverse engineering	If you set Redshift as the database, ensure that you do the following: 1. On the ODBC Data Source Administrator dialog box, go to the **System DNS** tab. 2. Select the Redshift data source and click **Configure**. The Amazon Redshift ODBC Driver DSN Setup dialog box opens. 3. Under Encrypt Password For, ensure that the **All Users of This Machine** check box is selected.
Version	Specifies database version for reverse engineering	
Predefined List	Displays a list of predefined databases for reverse engineering	
Reverse Engineer	Specifies reverse engineering options for connecting with the selected database. The Reverse Engineering Wizard appears.	On the Reverse Engineering Wizard, click **Connections** to set up database connections. For more information on database specific connection parameters, refer to the Database Connection Parameters topic. You can also configure the reverse engineering options available on the wizard. For more information, refer to the Setting Reverse Engineering Options topic.
Remote	Indicates whether to use a remote server for reverse	

Option	Description	Additional Information
	engineering	
Predefined Server Configuration	Displays the lists of pre-defined remote servers for reverse engineering	
Server New		
Port	Specifies the port number for the remote server	
Remote Test	Click this option to test the remote server connection	

4. Click **OK**.
   Your RE job is scheduled. It runs as configured, and the job status and its event log is displayed.

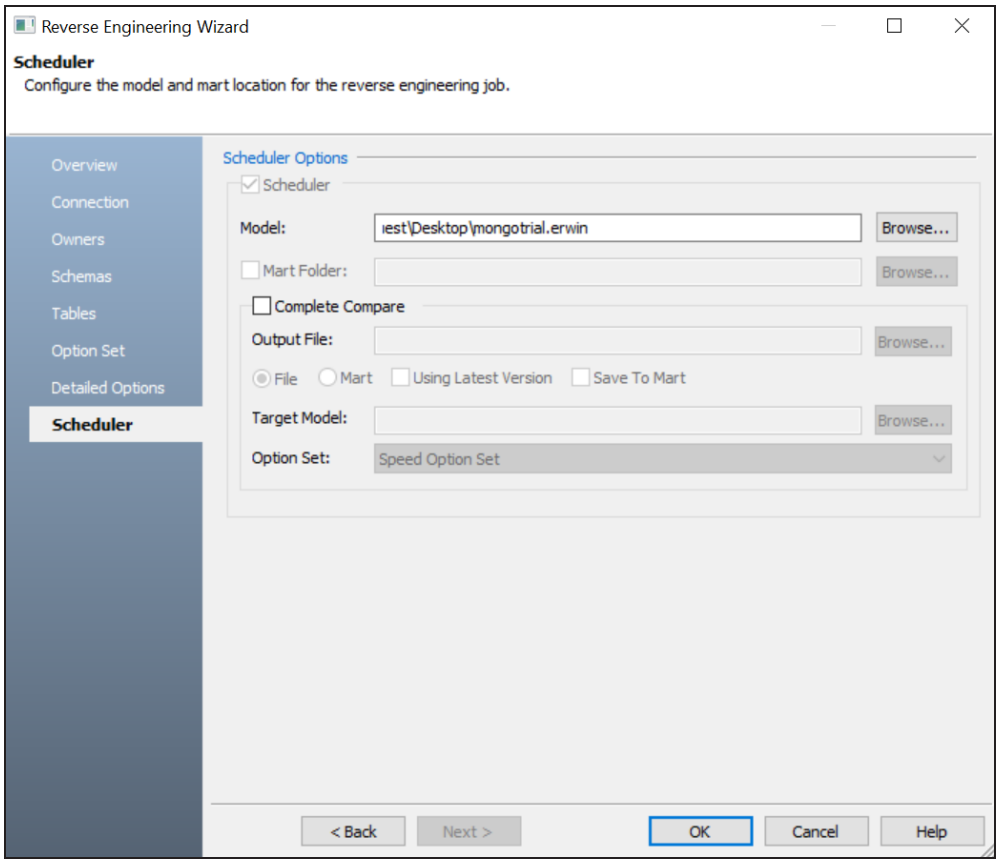


Depending on the settings you make and the job duration that you set, the job tile displays the following information about the job:

- Name
- Status
- Start and end times
- Run time

Running Complete Compare

The Scheduler tab of the Reverse Engineering Wizard now provides options to run the Complete Compare process when you reverse engineering a model to the mart. This enables you to compare the reverse engineering result with the model in your mart. In case of differences, you can save the updates as the latest version of your model in the mart.



Refer to the following table for option description:

Parameter	Description	Additional Information
Mart Folder	Specifies the location/library in your mart where the reverse engineered model should be saved.	To use this option, ensure that you are connected to a mart. For more information, refer to the Connecting to Mart topic.

Complete Compare	Specifies whether the Complete Compare (CC) process should run while reverse engineering	
Output File	Specifies the location of the CC output file generated after the reverse engineering process	
File	Specifies that the target model location is on the local system	
Mart	Specifies that the target model location is in the mart	
Using Latest Version	Specifies whether the target model is the latest version of the model in the mart	This option is available only when Mart is selected.
Save To Mart	Specifies whether the reverse engineered model is saved to the mart	This option is available only when Using Latest Version is selected.
Target Model	Specifies the location of the target model for CC	
Option Set	Specifies the option set that must be used for CC	**Advanced Default Option Set**: Indicates that all erwin DM metadata is included. CC works slowest with this option. **Speed Option Set**: Indicates that only the essential metadata is included. CC works the fastest with this option set. **Standard Default Option Set**: Indicates that standard metadata is included. CC works fast with this option set compared to the Advanced option set.

Productivity and UI Enhancements

Enhancements have been implemented to improve erwin DM Scheduler's productivity and usage experience. These enhancements are:

- Run Multiple Jobs
- Predefined Reverse Engineering Configurations

Run Multiple Jobs

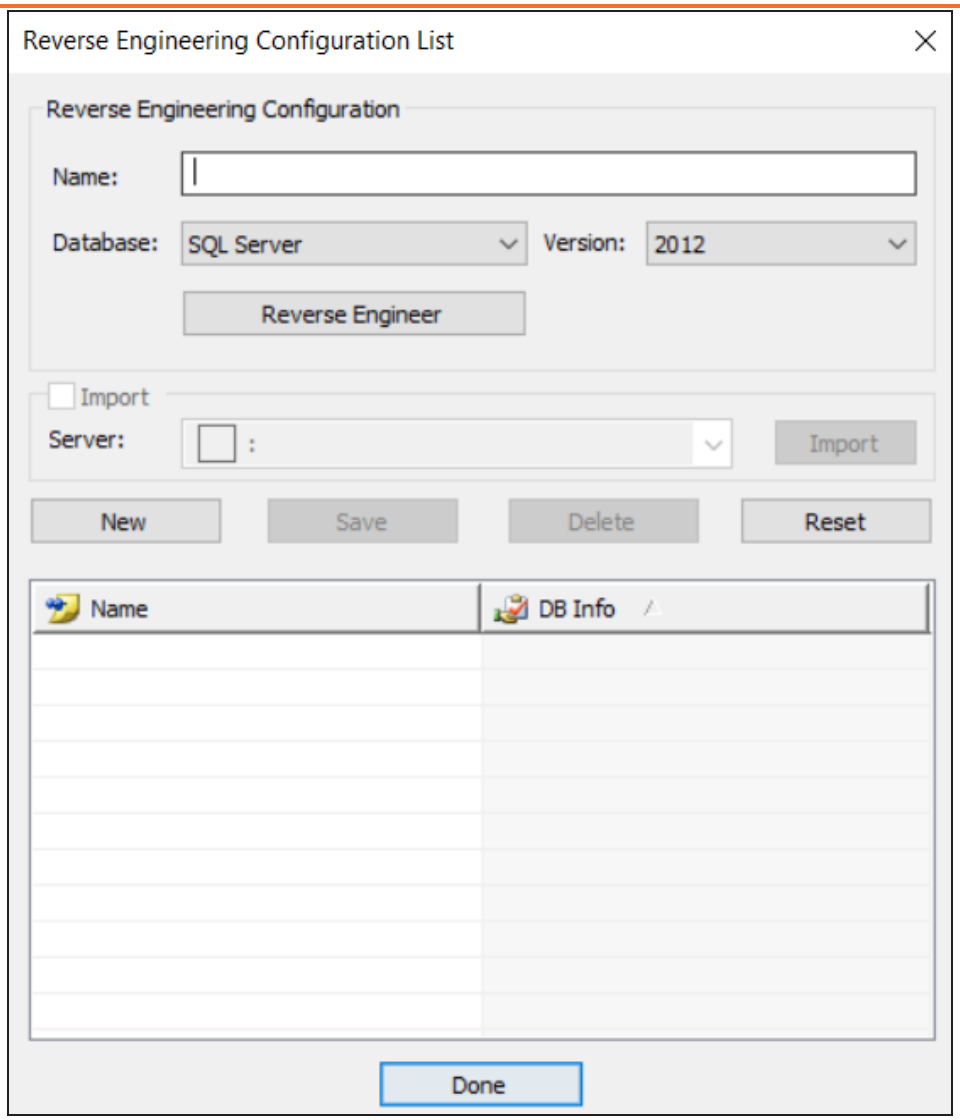
ewin DM Scheduler can now run multiple jobs in parallel on a remote server at the same time.

Predefined Reverse Engineering Configurations

You can create or import database reverse engineering configurations and use that configuration as a predefined configuration for scheduling a job. Access these predefined list on the erwin DM Scheduler Event Details page.

To create a reverse engineering configuration, follow these steps:

1. On the ribbon, in the Settings group, click **Reverse Engineer Configuration**.
   The Reverse Engineer Configuration List appears.



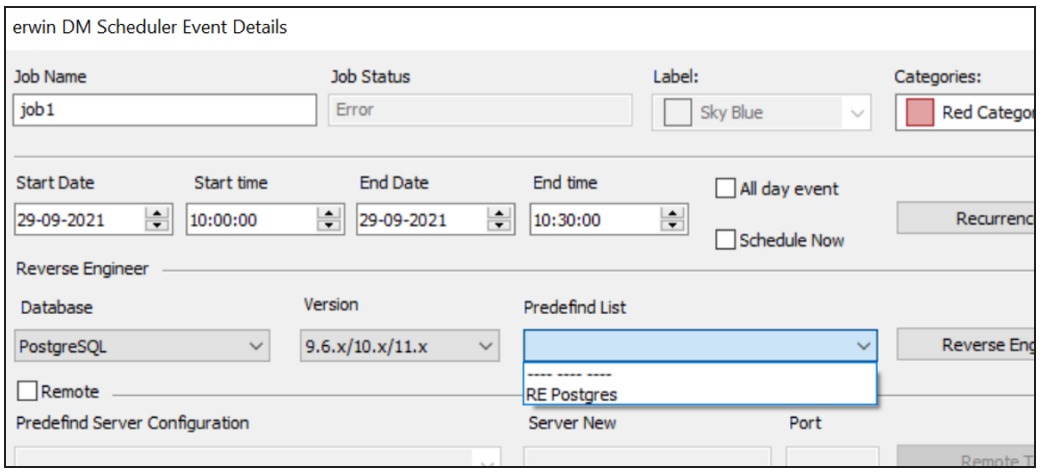
2. On the Reverse Engineer Configuration List, use the following options in the below table to create or import configurations.

Option	Description
Name	Enter a name for the configuration.
Database	Select a database for reverse engineering.
Version	Select a database version for reverse engineering.

Option	Description
Reverse Engineer	Select this option to specify database options for reverse engineering. The Reverse Engineering Wizard appears. On the Reverse Engineering Wizard, click **Connections** to set up database connections. For more information on database specific connection parameters, refer to the Database Connection Parameters topic.  You can also configure the reverse engineering options available on the wizard. For more information, refer to the Setting Reverse Engineering Options topic.
Import	Select this option to import configurations saved on a remote server.
Server	Select a server on the drop-down, then click **Import**. The imported configurations are displayed in the configuration list.

3. Once you have created a configuration, on the Reverse Engineering Configuration List, use one of the following options:
   - **New**: Use this option to create a new reverse engineering configuration. Selecting this option resets the Reverse Engineering Configuration section to add a new one.
   - **Add**: Use this option to add the new configuration. The added configurations are displayed in the configurations list.
   - **Save**: Use this option to save the changes to a selected configuration on the list.
   - **Delete**: Use this option to delete the selected configurations on the list.
   - **Reset**: Use this option to reset the data in the Reverse Engineer Configuration section.
4. Click **Done**.
   The reverse engineering configurations are saved as predefined configurations. When you schedule a job, you can select this configuration under **Predefined List** on

the erwin DM Scheduler Event Details page.



Git Support

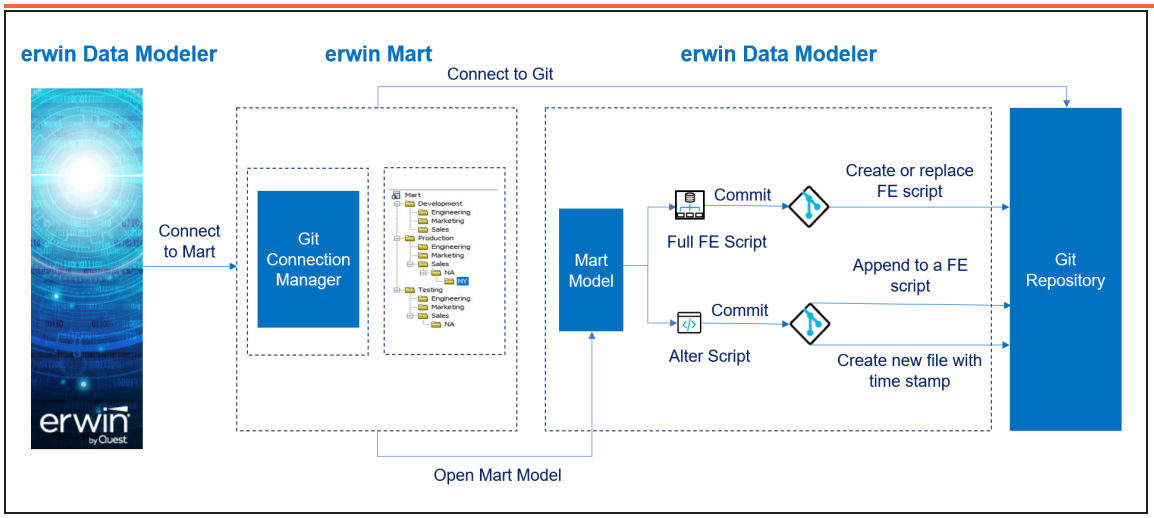
Starting erwin Data Modeler (DM) 12.0, you can connect erwin DM to Git repositories via Mart Server. This enables you to push Forward Engineering (FE) scripts for a Mart Model to GitLab or GitHub. You cannot store FE scripts or DDL on a Mart Server but only erwin models. With Git support you can adopt DevOps principles as you can commit FE scripts in Git repositories. Working with these repositories help you in:

- DevOps adoption
- collaboration with team members
- version control
- workflow management
- data integrity

Pushing FE scripts to a Git repository involves:

1. Connecting erwin DM to Mart Server
2. Connecting erwin DM to a Git repository
3. Opening a Mart Model and committing FE scripts

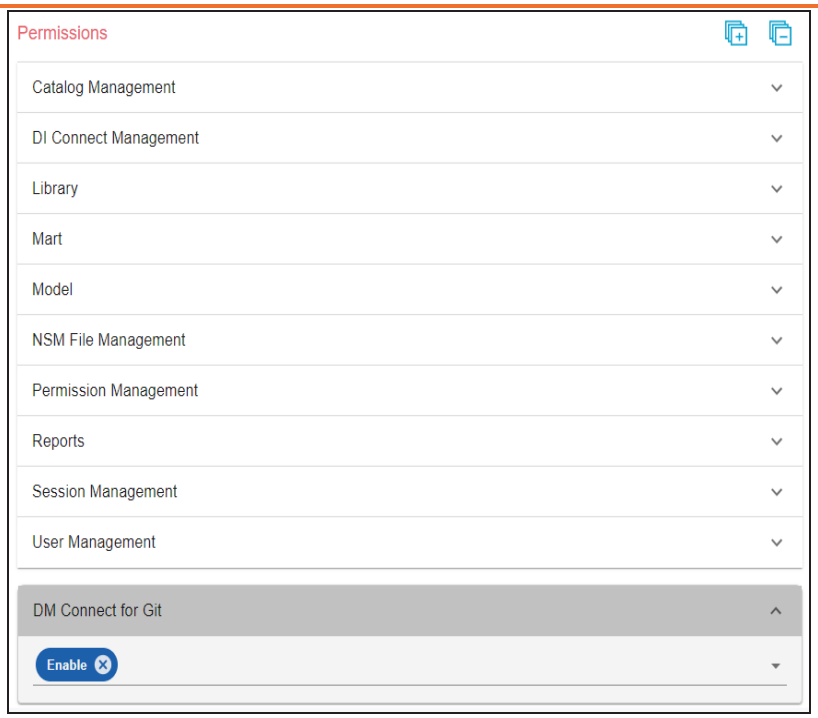
To summarize, following is the workflow to commit FE scripts.



Connecting to Git Repositories

A Git repository may be hosted on GitLab or GitHub. For a successful connection to these repositories, following are the prerequisites:

- **erwin Mart**: Ensure that,
  - erwin DM is connected to erwin Mart Server. For more information on connecting erwin DM to Mart Server, refer to the Connect to Mart topic.
  - the DM Connect for Git permission is enabled for your Mart user profile at the root, Mart level. By default, this permission is enabled for the out-of-box Admin profile.

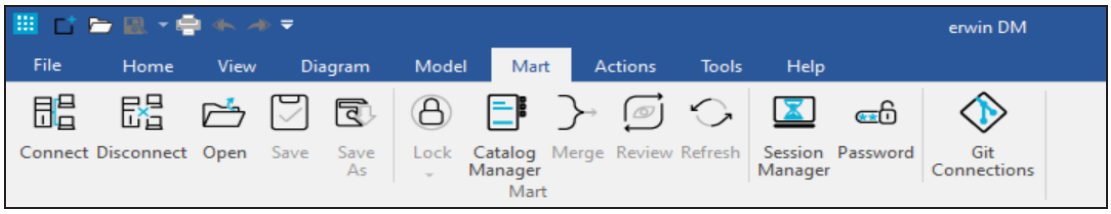


This permission is not available for other out-of-box profiles.

- **Personal Access Token**: Ensure that you have created the required personal access token. To know how to create personal access tokens for GitLab, refer to the GitLab documentation. To know how to create personal access tokens for GitHub, refer to the GitHub documentation.

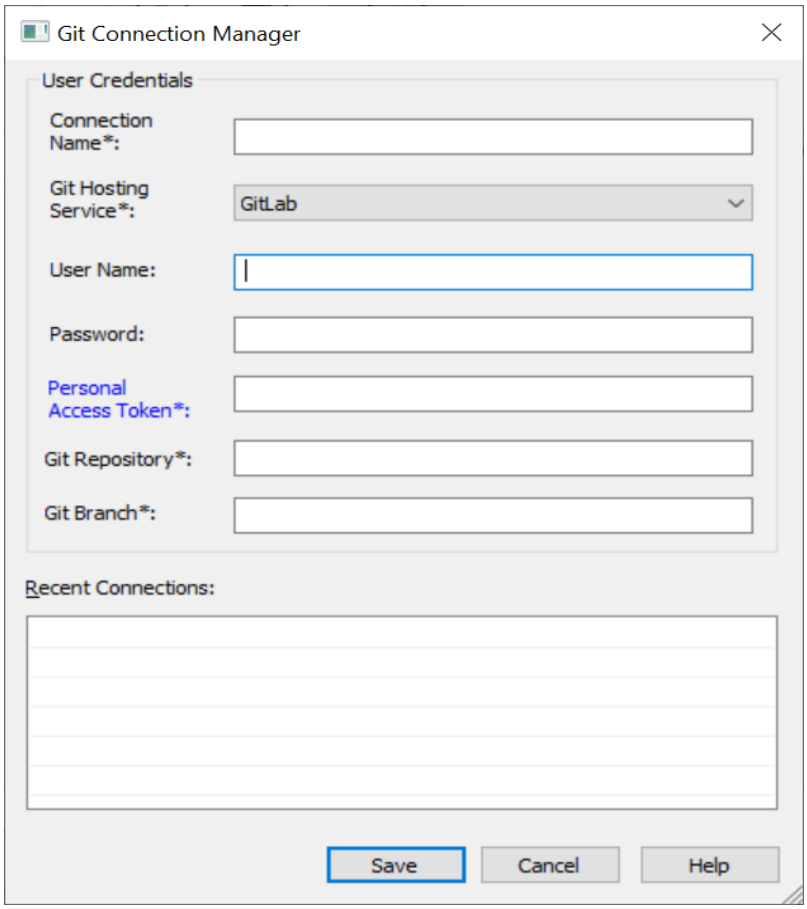
Once, these prerequisites are in place, to connect Git repositories to erwin DM, follow these steps:

1. On the ribbon, click **Mart**.



2. Click **Git Connections**.

The Git Connection Manager page appears.



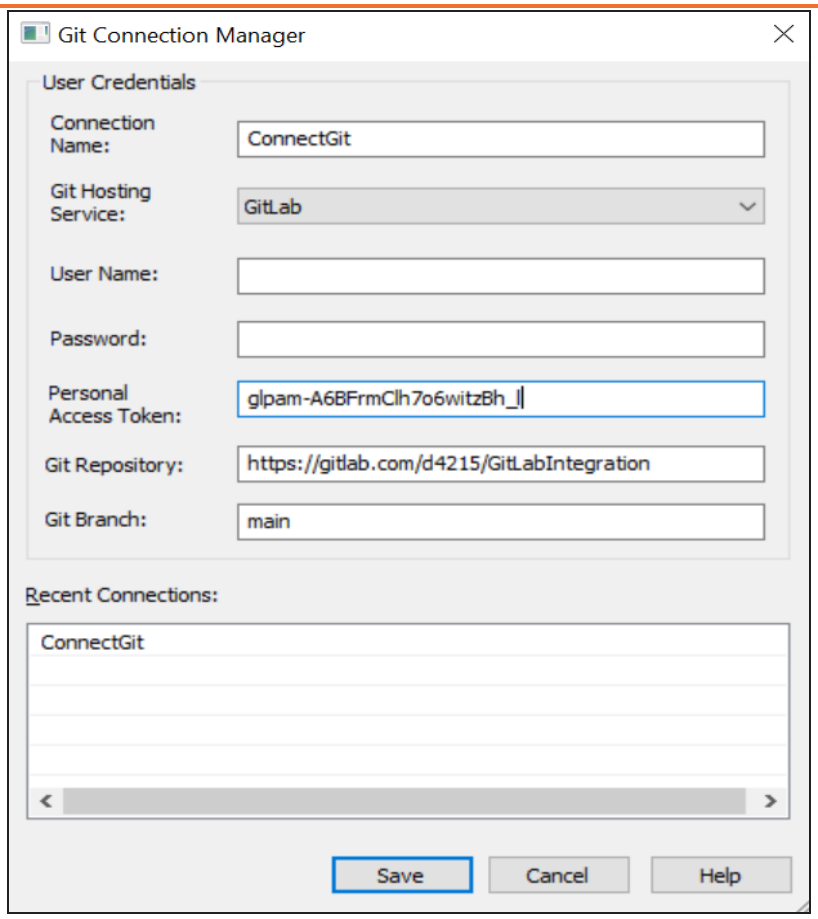
3. Enter appropriate values in the fields. Refer to the following table for field descriptions.

Field Name	Description	Additional Information
Connection Name	Specifies a user defined connection name	For example, ConnectGit. You can create multiple connections one for each Git repository.
Git Hosting Service	Specifies the Git hosting service to which erwin DM connects	**GitLab**: Indicates that erwin DM connects to GitLab **GitHub**: Indicates that erwin DM connects to GitHub

Field Name	Description	Additional Information
User Name	Specifies the username to log on to the Git hosting service	This field is not mandatory.
Password	Specifies the password to log on to the Git hosting service	This field is not mandatory.
Personal Access Token	Specifies the personal access token to connect to the Git hosting service	
Git Repository	Specifies the URL of a Git repository where you want to push the forward engineering script	For example, https://-gitlab.com/d4215/GitLabIntegration or https://github.com/poly-inc/poly-main-MCL
Git Branch	Specifies the branch that is used to push the forward engineering script	For example, main.

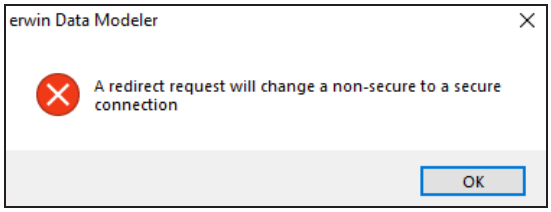
4. Click **Save**.

   On successful connection, the connection name appears under Recent Connections.



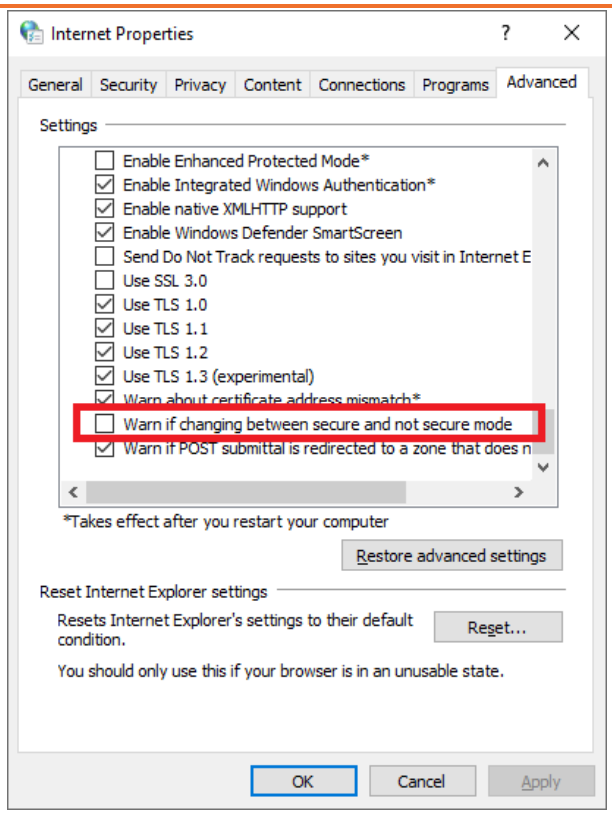
Troubleshooting

While setting up your connection, you may encounter the following error:



To resolve this error, follow these steps:

1. On your system, go to **Control Panel** > **Internet Options** > **Advance Tab**.
2. Clear the **Warn if changing between secure and not secure mode** check box.



3. Click **OK**.
4. Close and reopen erwin DM.
5. Connect erwin DM to Mart Server.
6. Launch the Git Connection Manager page and configure the Git connection.

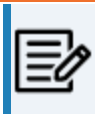
Once you are connected to a Git repository, you can commit FE scripts.

Committing Forward Engineering Scripts

There are two scenarios in which you commit Forward Engineering (FE) scripts to a Git repository:

- **Scenario 1: Committing new or full FE scripts**:

  Use the Forward Engineer Schema Generation Wizard to commit a physical database schema or FE script from a Mart Model.



To avoid script files from being overwritten, ensure that you use unique file names.

- **Scenario 2: Committing alter scripts**:

  Use the Forward Engineer Alter Script Schema Generation Wizard to commit an alter script after you make changes to a Mart Model. You can commit an alter script in two ways:

  - **Commit and append an alter script to an existing script file**
  - **Commit and create a new alter script file in the Git repository**

  For more information, refer to the Scenario 2: Committing Alter Scripts topic.

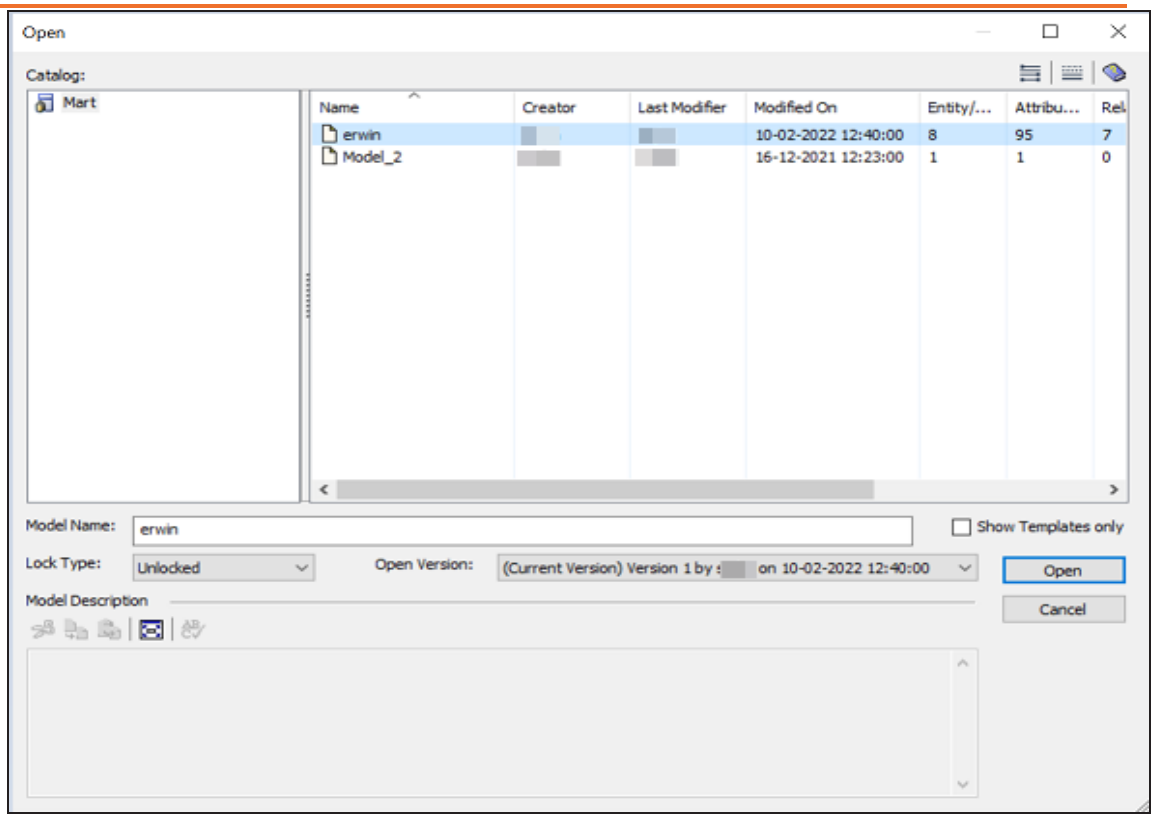
Scenario 1: Committing New or Full FE Scripts

The Forward Engineer Schema Generation Wizard generates a physical database schema or Forward Engineering (FE) script. For a Mart Model, you can push the FE script to a Git repository.

To commit new or full FE scripts to Git repositories, follow these steps:

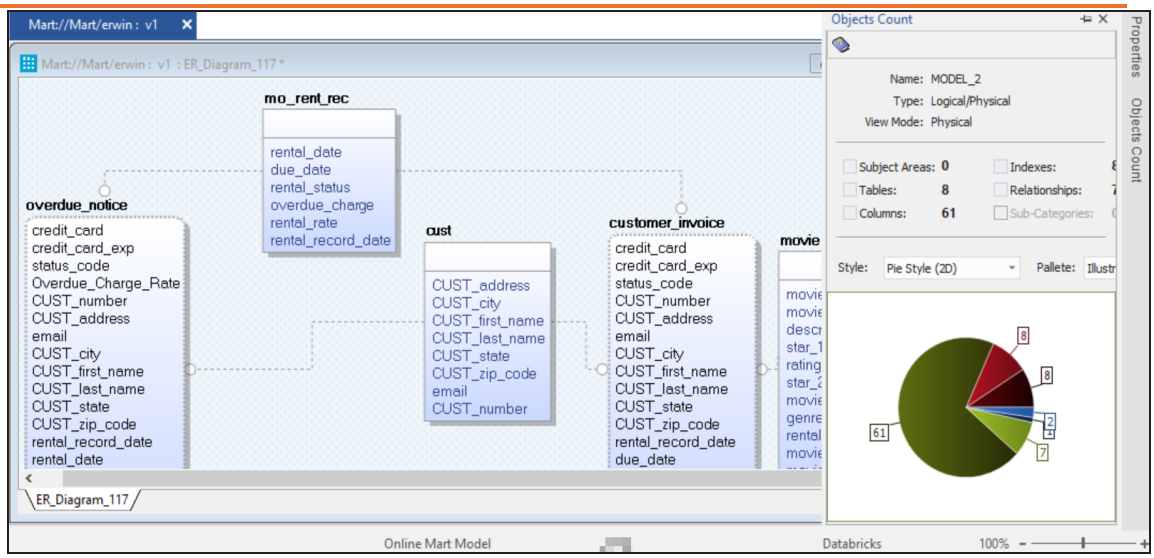
1. On the ribbon, go to **Mart** > **Open**.

   The Open page appears.



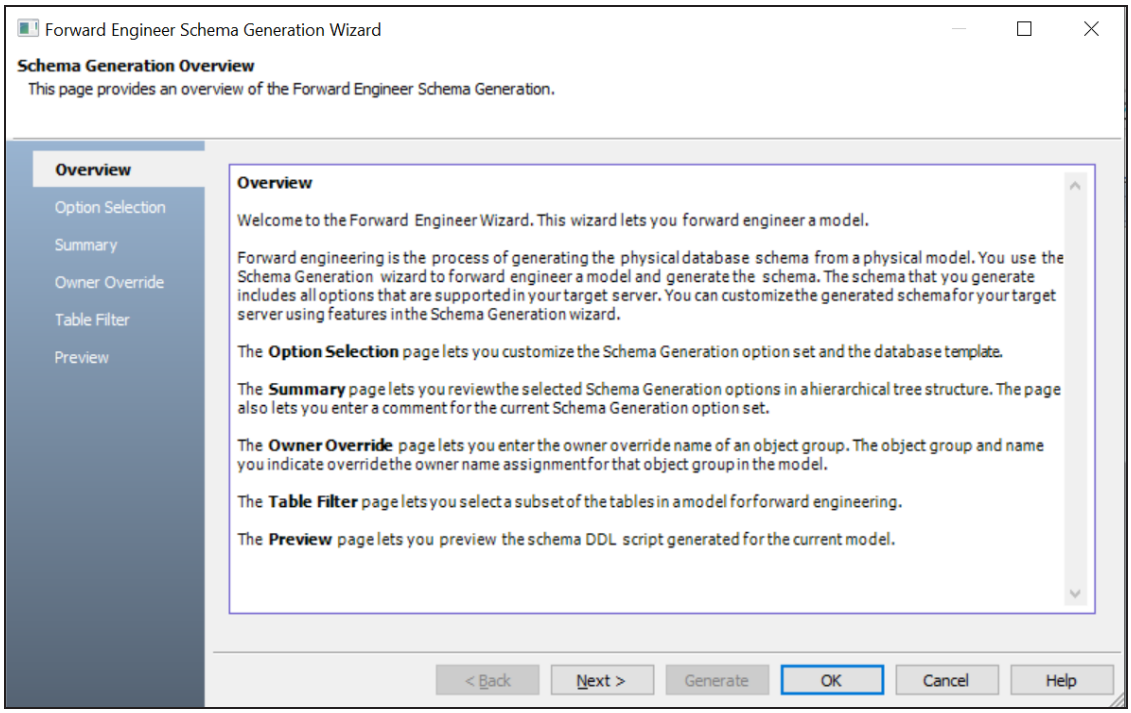
2. Select a model, and then click **Open**.

   The Mart Model opens.



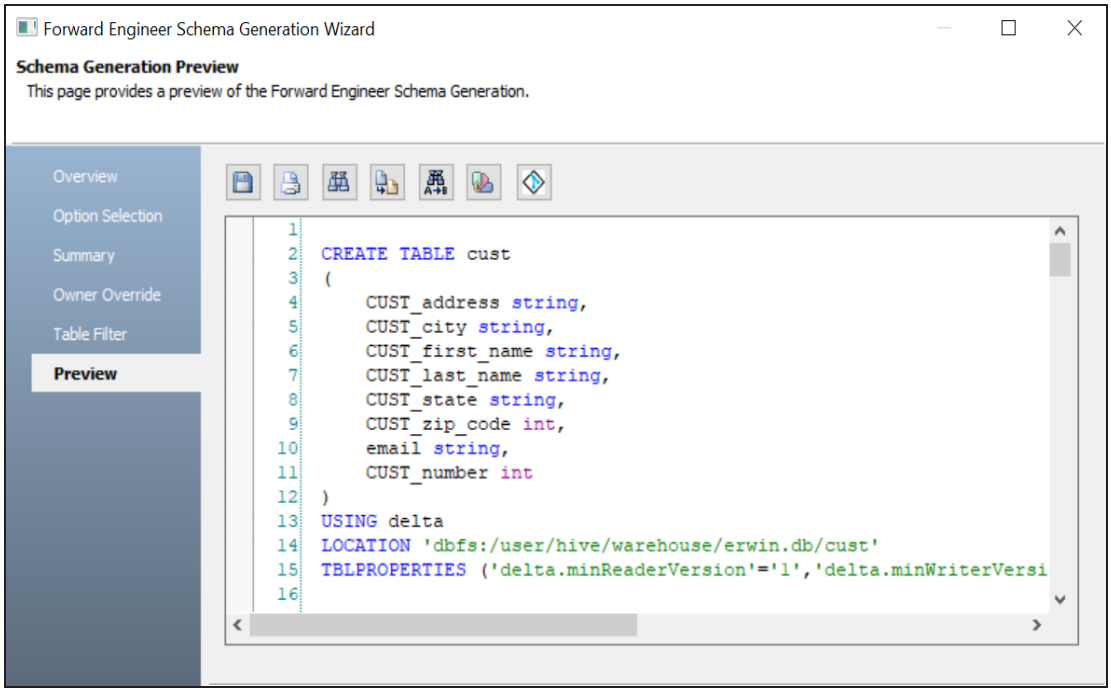
3. Go to **Actions** > **Schema**.

   The Forward Engineer Schema Generation Wizard appears.



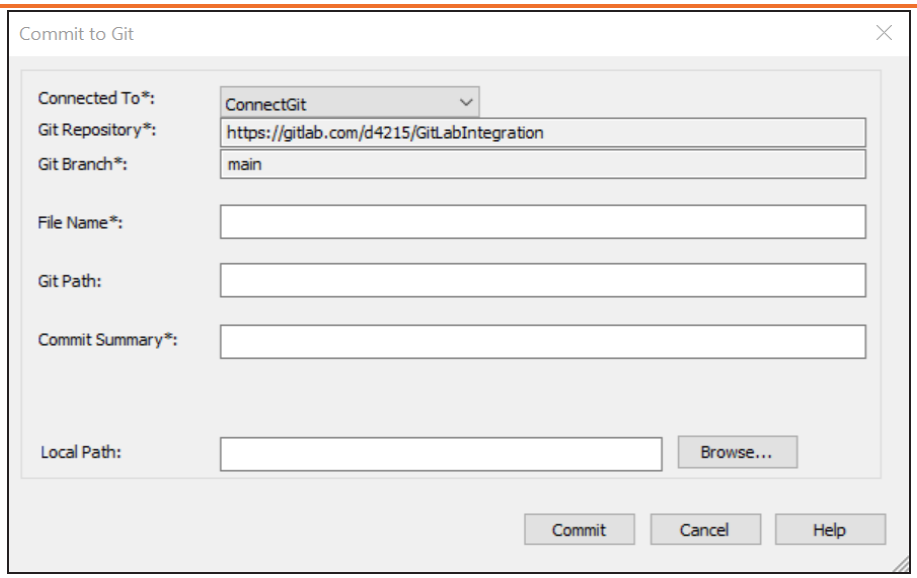
4. On the **Forward Engineer Schema Generation Wizard**, click the **Preview** section.

The FE script appears. For example, in the following image the Preview section displays FE script of a Databricks database. For more information on generating FE scripts, refer to the Forward Engineering/Schema Generation for Databases topic.



5. Click  .

   The Commit to Git screen appears.



6. Enter appropriate values in the fields. Fields marked with an asterisk (*) are mandatory. Refer to the following table for field descriptions.

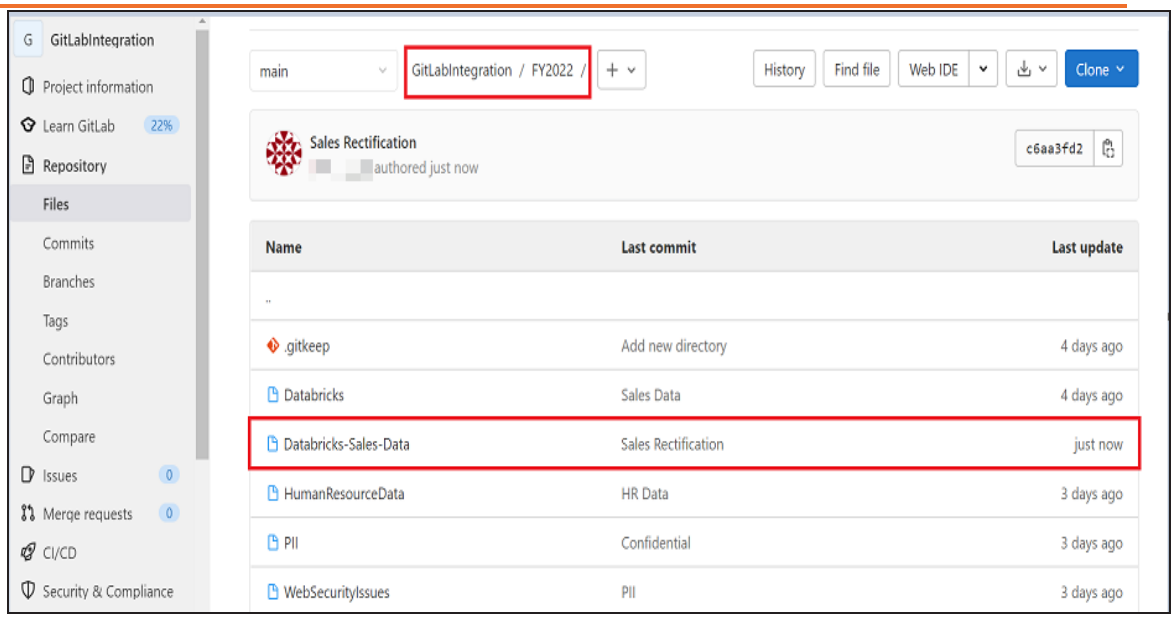
Field Name	Description	Additional Information
Connected To	Specifies the connection that connects erwin DM to a Git repository	For example, ConnectGit.
Git Repository	Specifies the Git repository configured for the connection	For example, https://-gitlab.com/d4215/GitLabIntegration is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.
Git Branch	Specifies the Git branch configured for the connection	For example, main is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.
File Name	Specifies the user-defined	For example, Databricks-Sales-Data.sql

Field Name	Description	Additional Information
	name of the FE script file being committed to a Git repository	To avoid script files from being overwritten, ensure that you use unique file names.
Git Path	Specifies the location in the Git repository where the FE script is committed	For example, FY2022/ The FE script is committed to the FY2022 folder inside the root folder of your Git repository.
Commit Summary	Specifies the summary of the push commit	For example, Sales Rectification.
Local Path	Specifies the location on your local machine where the FE script is saved	C:\Users\SO\Documents\Databricks

7. Click **Commit**.

   The FE script file is saved on the local path and committed to the Git repository.

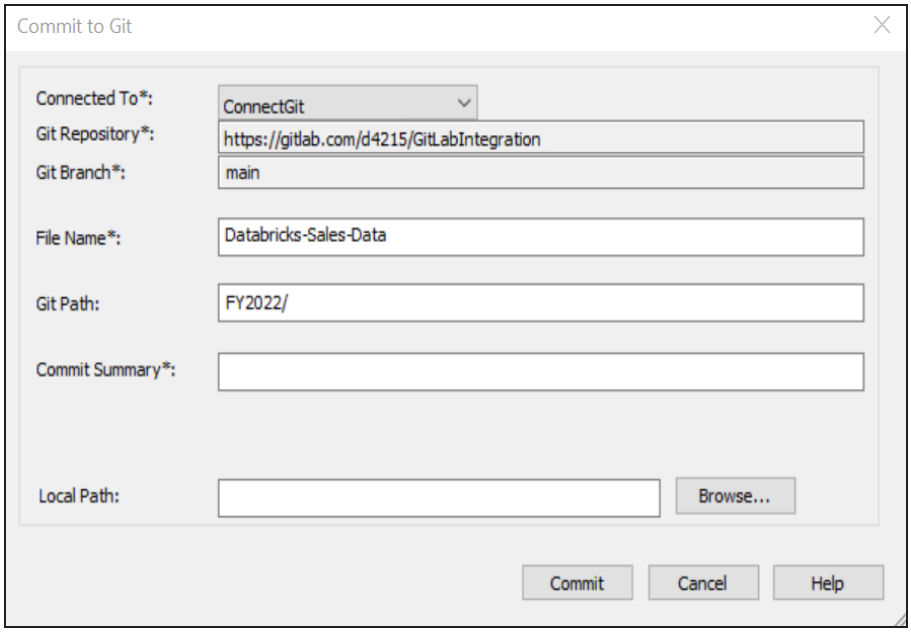
   For example, in the following image, FE script is committed to a GitLab repository in a file, Databricks-Sales-Data, with a commit summary, Sales Rectification using the main branch.



You can click the file to review its content. For example, in the following image, Databricks-Sales-Data's content is visible.

You can use FE Schema Generation Wizard to commit FE script using the same connection again. The Commit to Git screen autopopulates the previously set values in File Name and Git Path.

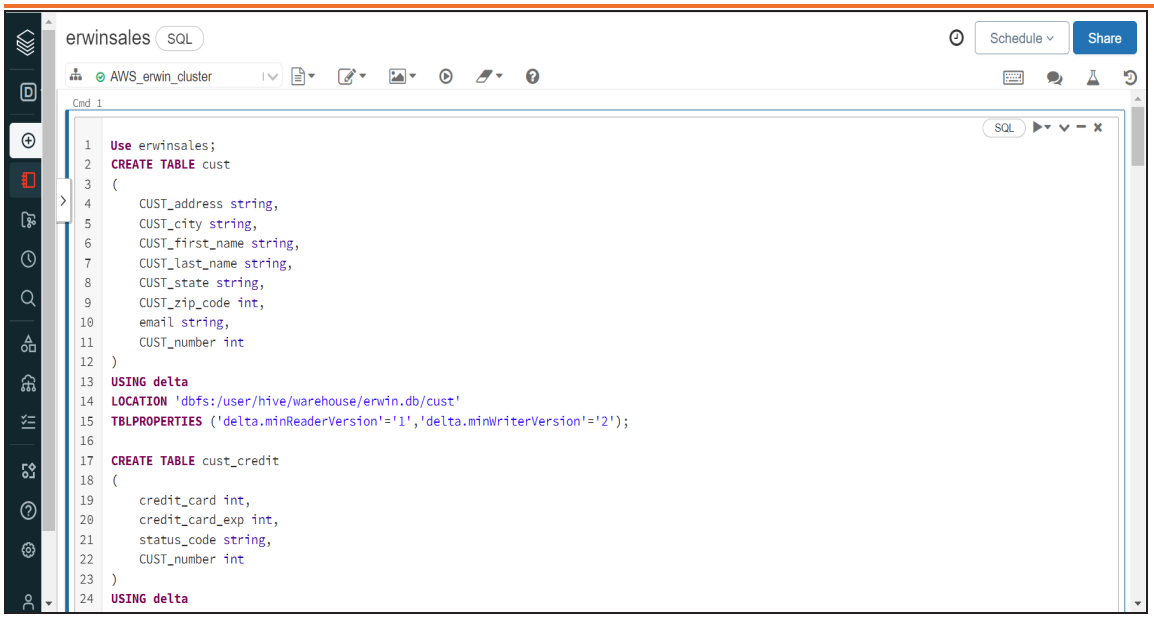
For example, in the following image File Name is set to Databricks-Sales-Data and Git Path is set to FY2022/.



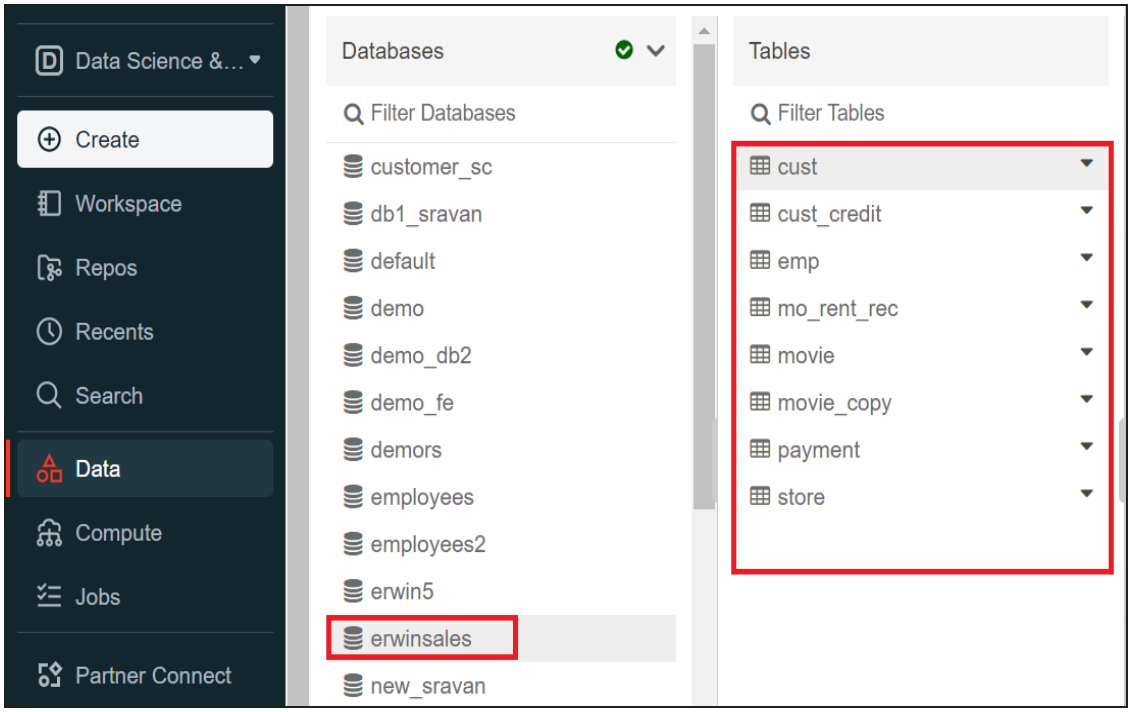
Committing the FE script again with the same File Name and Git Path overwrites the previous file in the Git repository.

Once the FE script is committed, you can run it on your database to generate and verify the physical schema.

For example, in the following Databricks database, the FE script copied from the Git repository is run.



After running the FE script, the required database objects are created. You can access these objects from the database. For example, the following tables can be accessed in a Databricks database.



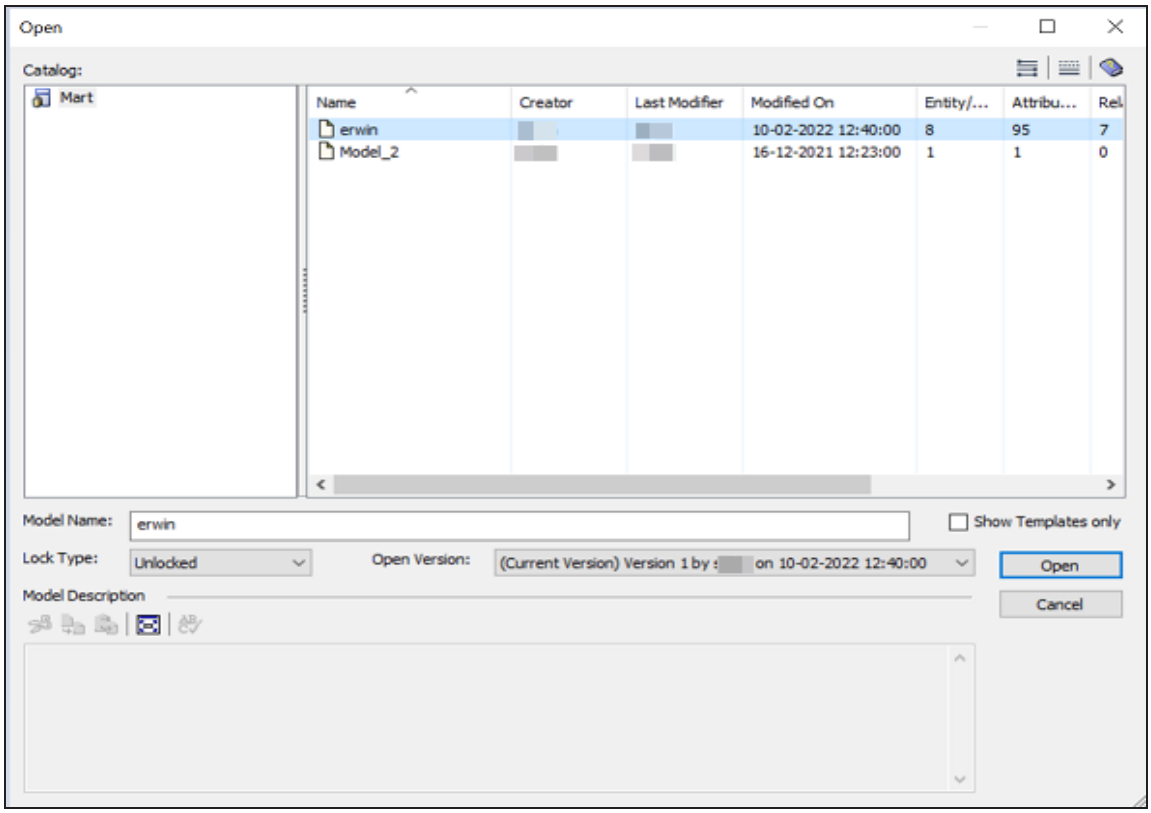
Scenario 2: Committing Alter Scripts

The Forward Engineer Alter Schema Generation Wizard generates an alter script for a database after you make changes to a model. For a Mart Model, you can push the alter script to a Git repository.

To commit alter scripts to Git repositories, follow these steps:

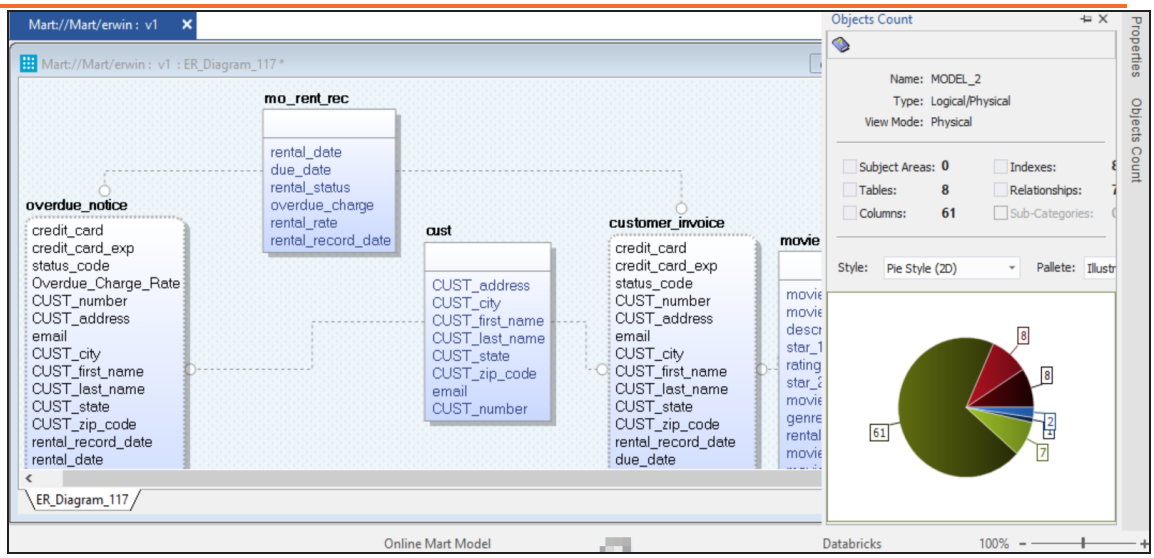
1. On the ribbon, go to **Mart** > **Open**

   The Open page appears.



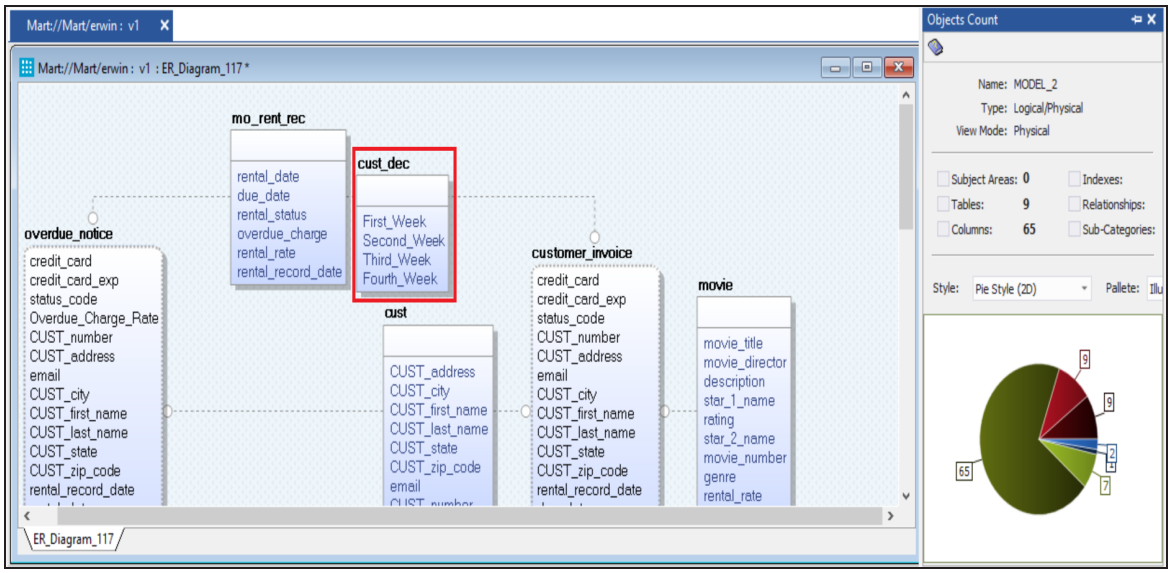
2. Select a model, and then click **Open**.

   The Mart Model opens.



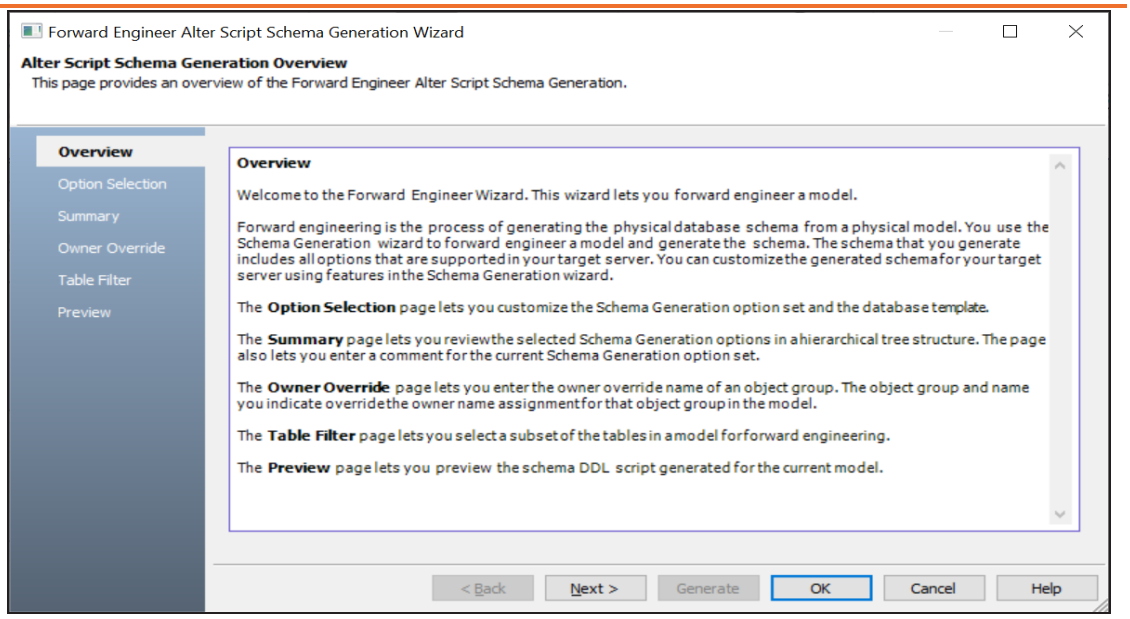
3. Make the required changes in the model.

   For example, in the following model, a new table, cust_dec with four columns is added.



4. Go to **Actions** > **Alter Script**.

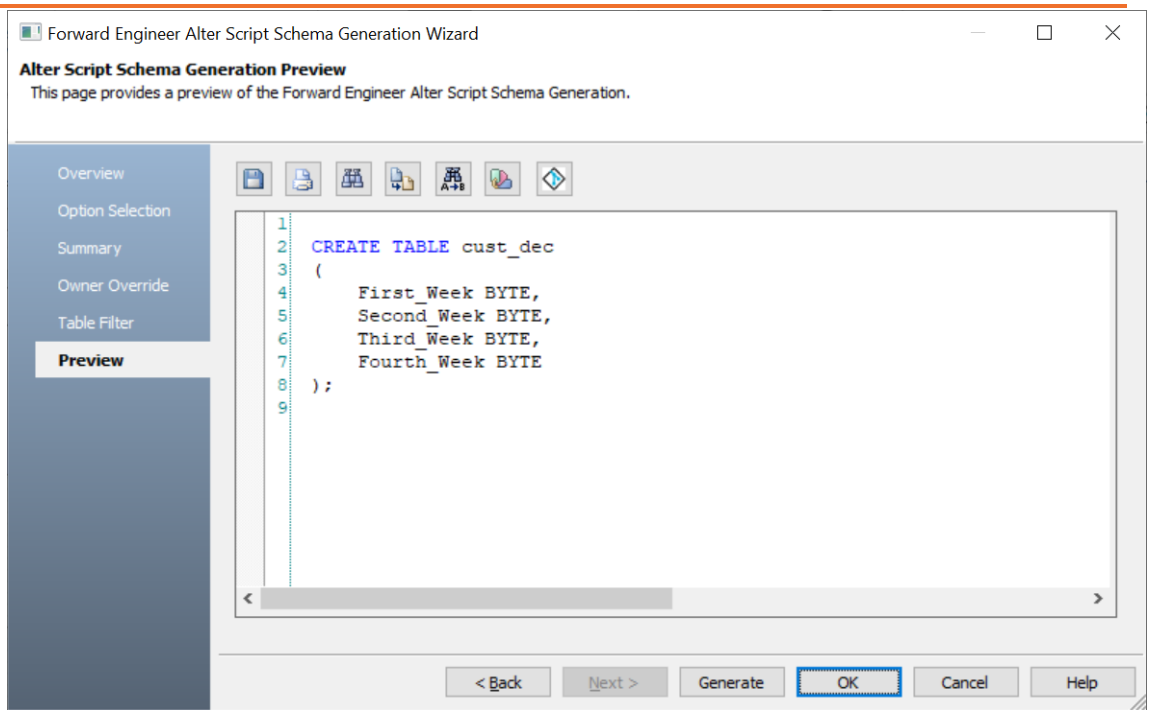
   The Forward Engineer Alter Script Schema Generation Wizard appears.



5. On the **Forward Engineer Alter Schema Generation Wizard**, click the **Preview** section.

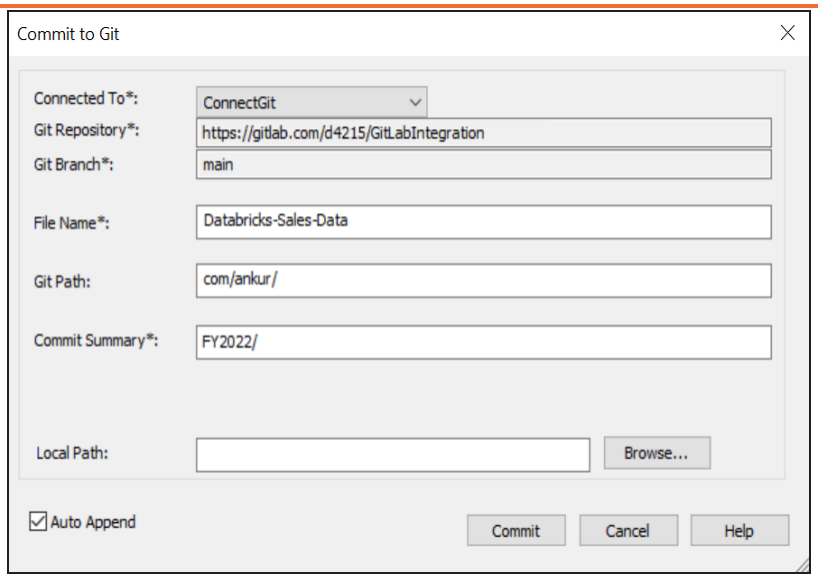
   The alter script appears. For more information on generating alter scripts, refer to the Generating Alter Script for Databases topic.

   For example, in the following image the Preview section displays an alter script of a Databricks database.



6. Click  .

   The Commit to Git screen appears. The File Name and Git Path values autopopulates with the values configured in the previous commit. You can update the File Name and Git Path as per the requirement.



7. Enter appropriate values in the fields. Fields marked with an asterisk (*) are mandatory. Refer to the following table for field descriptions.

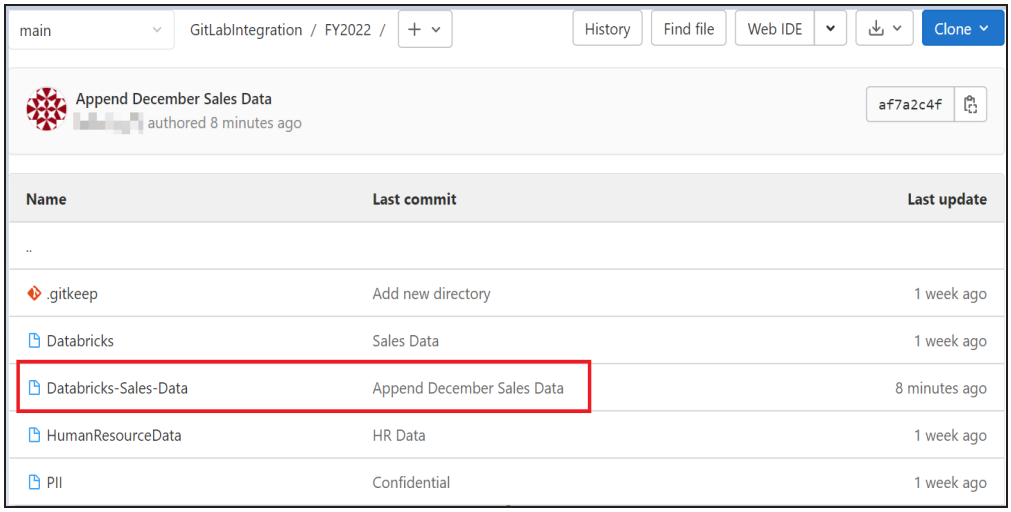
Field Name	Description	Additional Information
Connected To	Specifies the connection that connects erwin DM to a Git repository	For example, ConnectGit.
Git Repository	Specifies the Git repository configured for Connection	For example, https://-gitlab.com/d4215/GitLabIntegration is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.
Git Branch	Specifies the Git branch that was set for connection in the Git Connection Manager	For example, main is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.

Field Name	Description	Additional Information
File Name	Specifies the user-defined name of the FE script file being committed to a Git repository	For example, Databricks-Sales-Data.sql
Git Path	Specifies the location in the Git repository where the FE script is committed	For example, FY2022/ The FE script is committed to the FY2022 folder inside the root folder of your Git repository.
Commit Summary	Specifies the summary of the push commit	For example, Append December Sales.
Local Path	Specifies the location on your local machine where the Alter script is saved	C:\Users\SO\Documents\Databricks
Auto Append	Specifies whether the alter script is appended to the file set in File Name and Git Path	By default, the Auto Append check box is selected. To create a new script file, clear the Auto Append check box and set the File Name and File Path belonging to an existing file. A new file with the following naming convention: <File Name>_YYYY-MM-DD_HH-MM-SS is created. Ensure that you use this check box consistently every time you commit an alter script.

8. Click **Commit**.

   The alter script file is saved on the local path and committed to the Git repository.

For example, in the following image, an alter script file is committed to a GitLab repository and appended to an existing file, Databricks-Sales-Data, with a commit summary, Append December Sales using the main branch.



You can click the file to review its content. For example, in the following image, Databricks-Sales-Data contains the alter script.

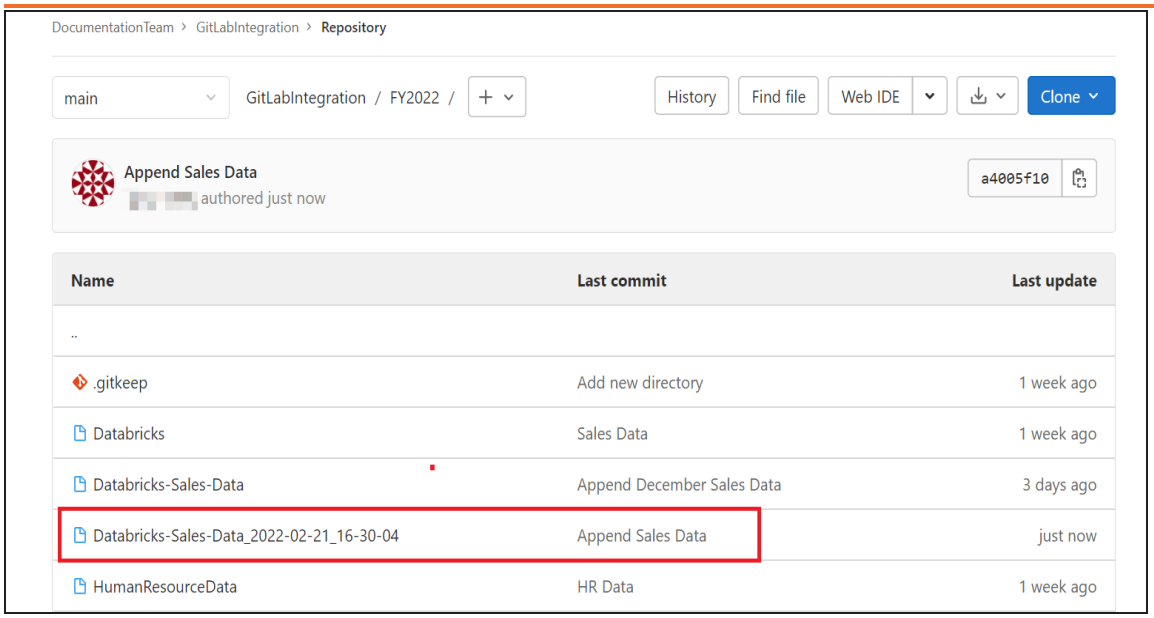
```
        status_code string,
        CUST_number int
)
USING delta
LOCATION 'dbfs:/user/hive/warehouse/erwin.db/cust_credit'
TBLPROPERTIES ('delta.minReaderVersion'='1','delta.minWriterVersion'='2');

CREATE TABLE cust_dec
(
        First_Week BYTE,
        Second_Week BYTE,
        Third_Week BYTE,
        Fourth_Week BYTE
);

CREATE TABLE emp
(
        EMP_first_name string,
        EMP_address string,
        EMP_phone int,
        EMP_address_2 string,
        email string,
        salary int,
        hire_date timestamp,
        soc_sec_number int,
        EMP_number string
```

Clearing the Auto Append check box and setting the File Name and File Path belonging to an existing file creates a new file with the following naming convention: <File Name>_YYYY-MM-DD_HH-MM-SS.

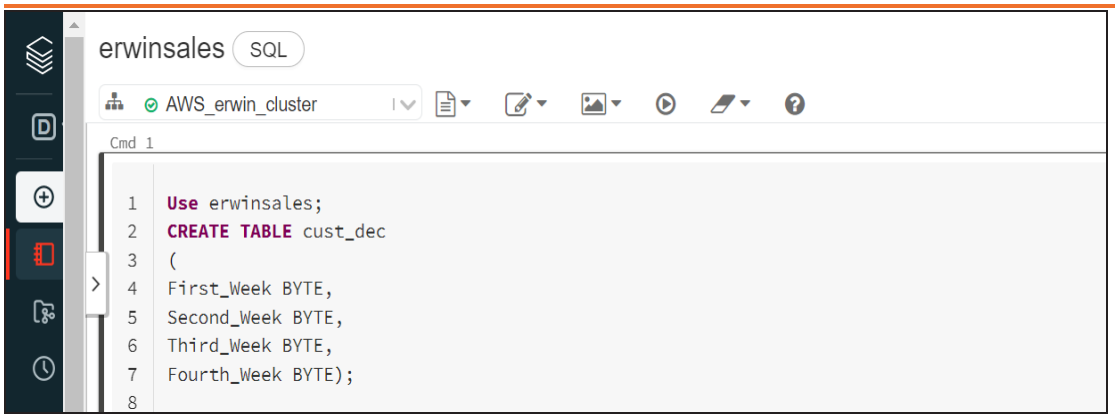
For example, in the following image, a file is created with a time stamp in a Git repository.



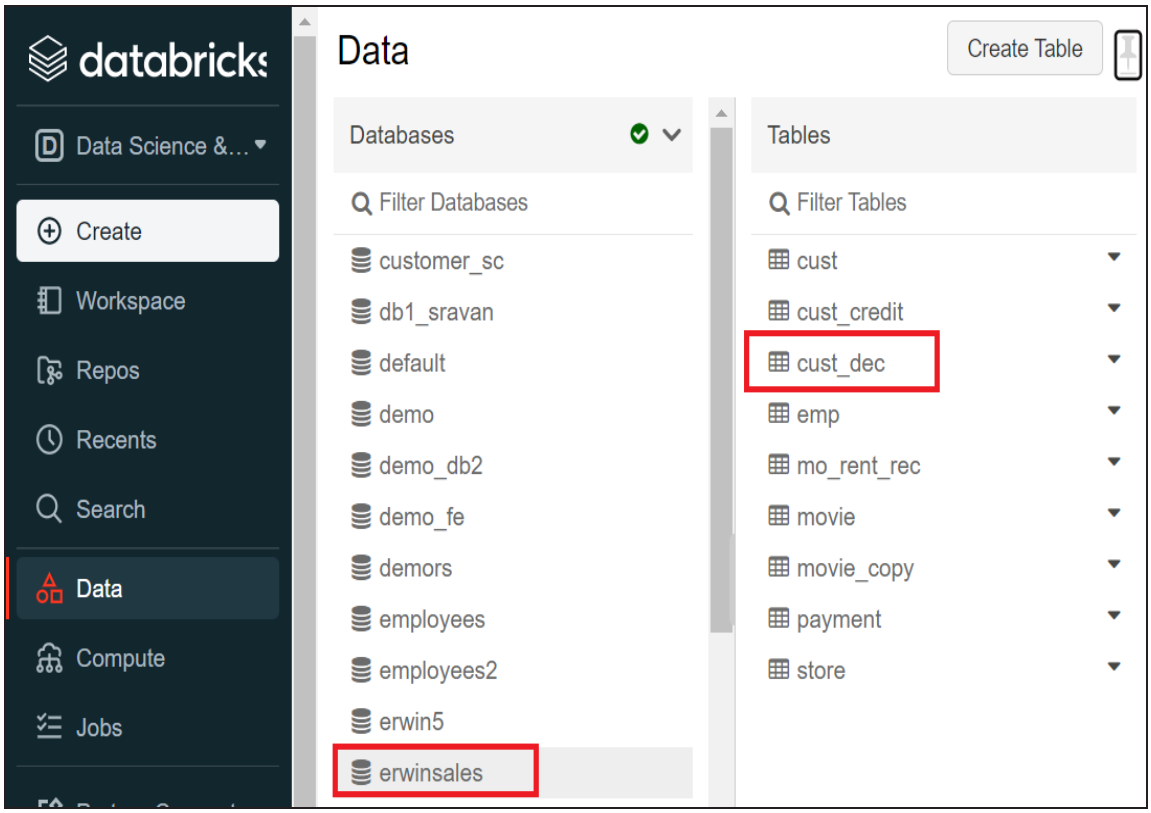
This file contains only the alter script.

Use the committed FE script to generate a physical schema in your database. To generate schema, copy the FE script from your Git repository and run the script in the database.

For example, in the following Databricks database, the FE script copied from the Git repository is run.



The cust_dec table is created in a Databricks database.



MongoDB: Schema Validation

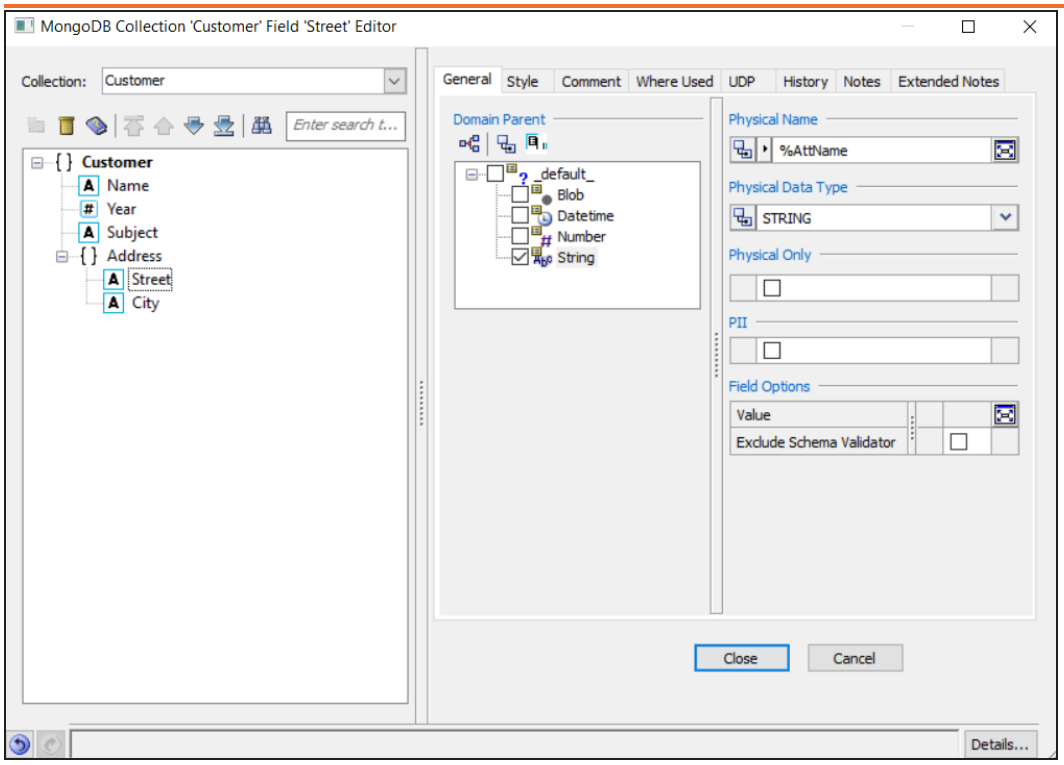
erwin Data Modeler (DM) 12.0 now supports schema validation for MongoDB forward engineering scripts. The schema validator verifies fields in collections based on their data type. If enabled, this feature restricts forward engineering script generation in case the data in the fields is not in line with the assigned data type. Thus, enabling you to avoid any errors in the forward engineering script. This feature is enabled by default for fields in collections.

In case of any exceptions, you can select the Exclude Schema Validator check box to exclude validations for one or more fields. Doing so overrides the assigned data type of a field and generates the script successfully.

This topic walks you through the steps to generate schema validation for fields in a simple table (collection) and generate schema using an example. The following table lists the fields and assigned data types in a MongoDB collection, Customer.

Fields	Data Type
Name	STRING
Year	INTEGER
Subjects	STRING
Address	OBJECT
Street	STRING
City	STRING

The following image displays the fields in a collection and the assigned data types.



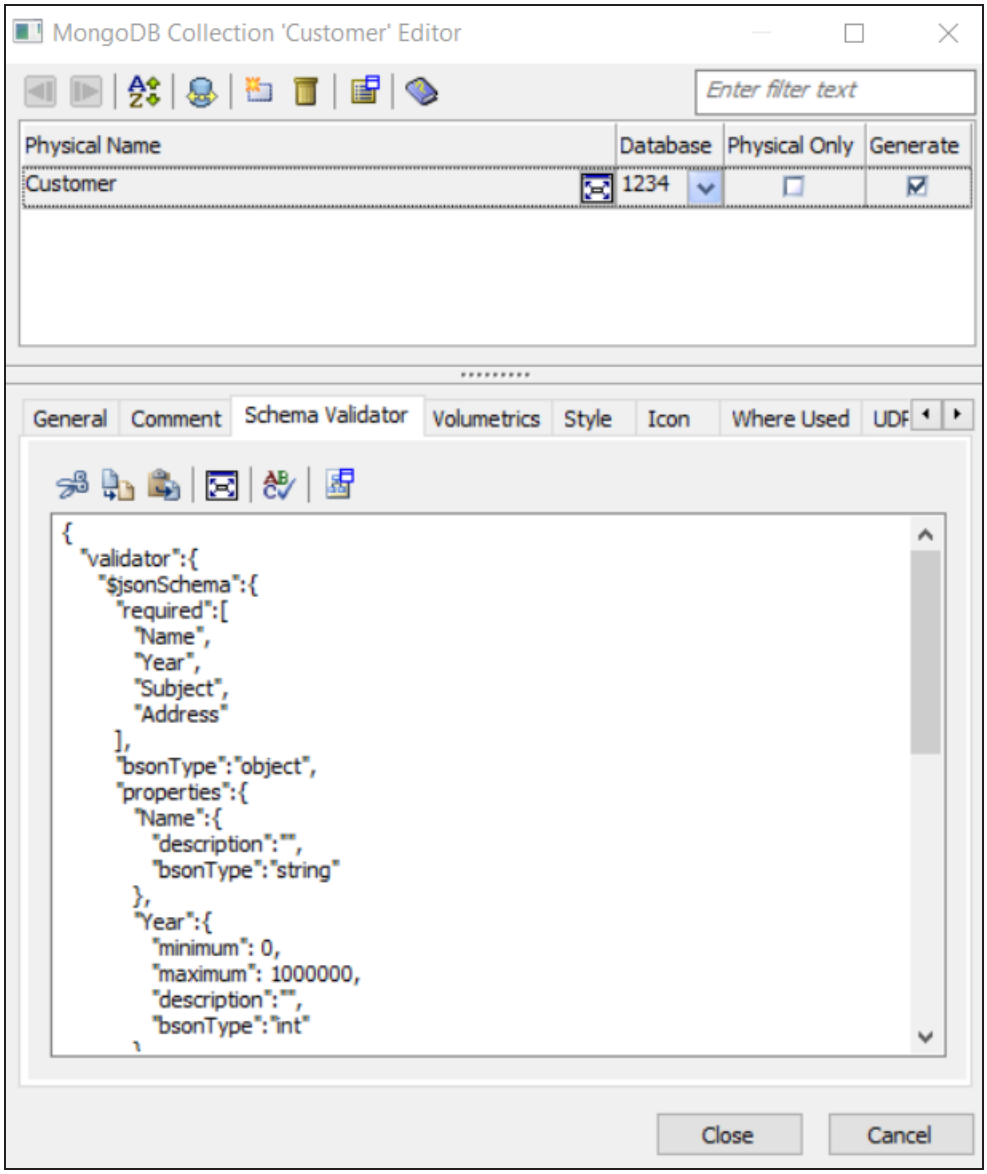
To generate schema validation for a MongoDB collection, follow these steps:

1. In erwin DM, create a collection with fields, and assign relevant datatypes.

   Ensure that a database in assigned to the collection.

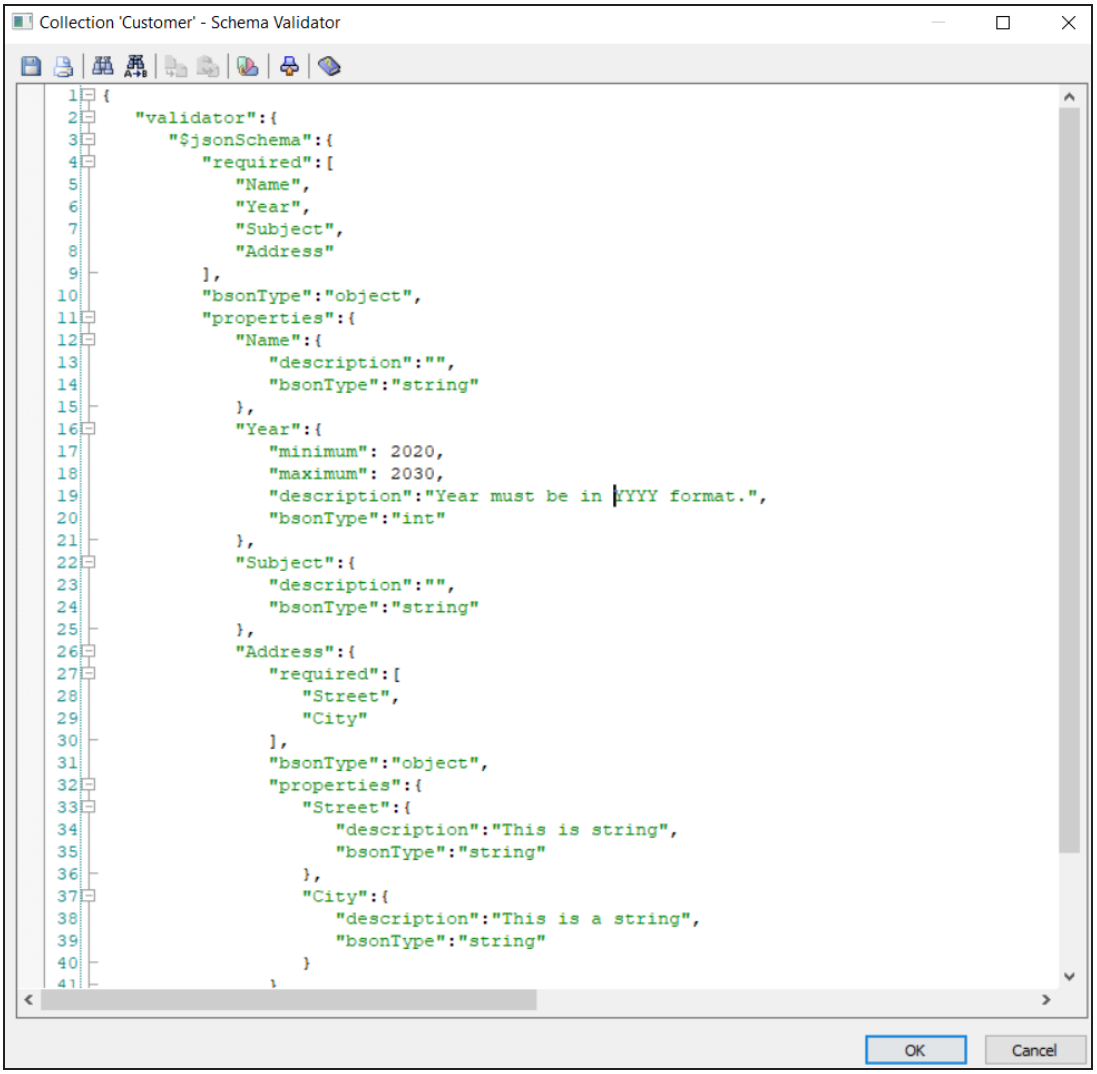
2. Right-click the collection and click **Collection Properties**.
   The MongoDB Collection Editor appears.
3. Go to the **Schema Validator** tab.
4. Click  to generate schema validation template.
   This displays the generated schema validation template based on the field properties

and assigned datatypes.



5. Click to edit the schema.
   The Collection Schema Validator editor appears. This displays the generated schema validation template with validations in edit mode.
6. Edit the schema template for validations and add a description for the fields based on your requirement.

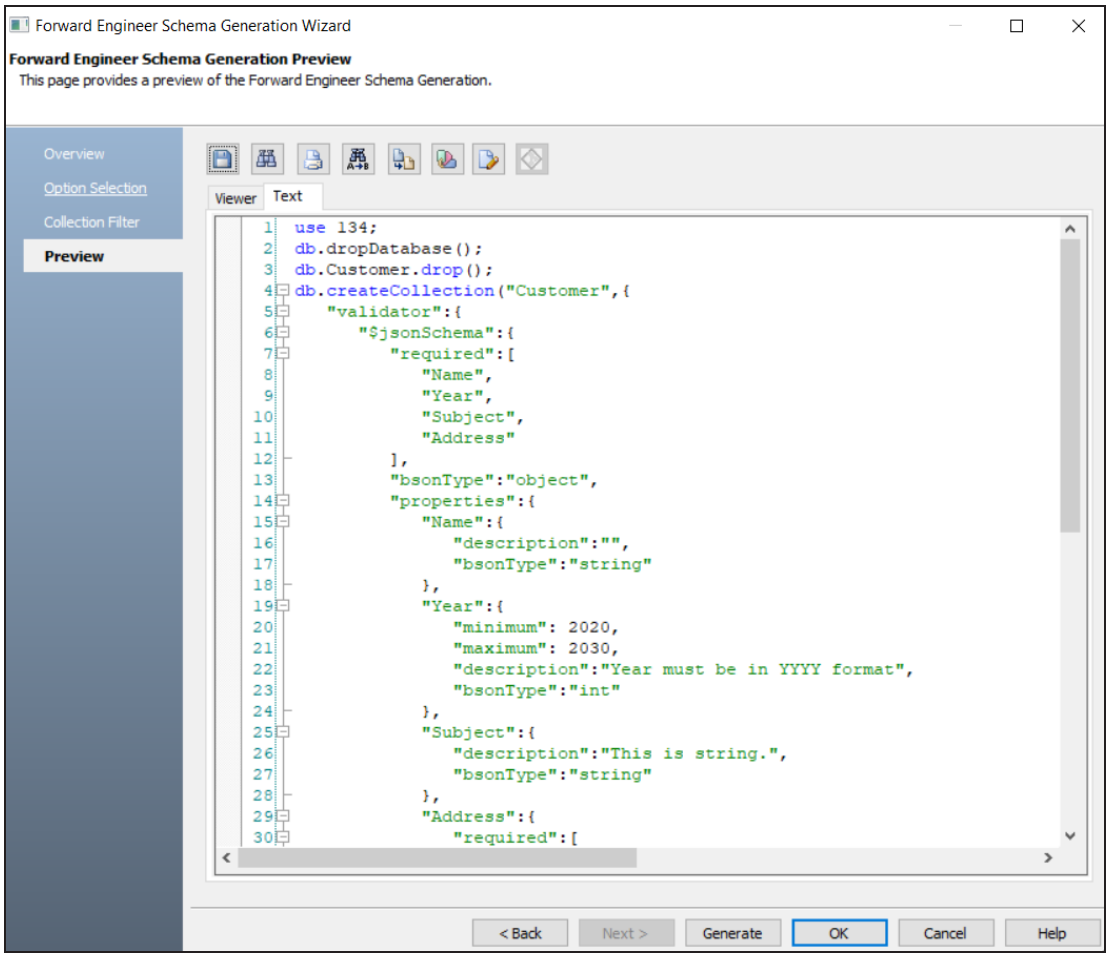
In this example, the validation properties of the fields and descriptions are updated as shown in the following image.



7. Once you have made changes to the schema template, click **OK**.
   The schema template validation changes are saved.

8. On the application ribbon, click **Action** > **Schema**.
   The Forward Engineer Schema Generation Wizard appears.

9. Go to **Preview** tab.
   The updated schema with the validations appear.

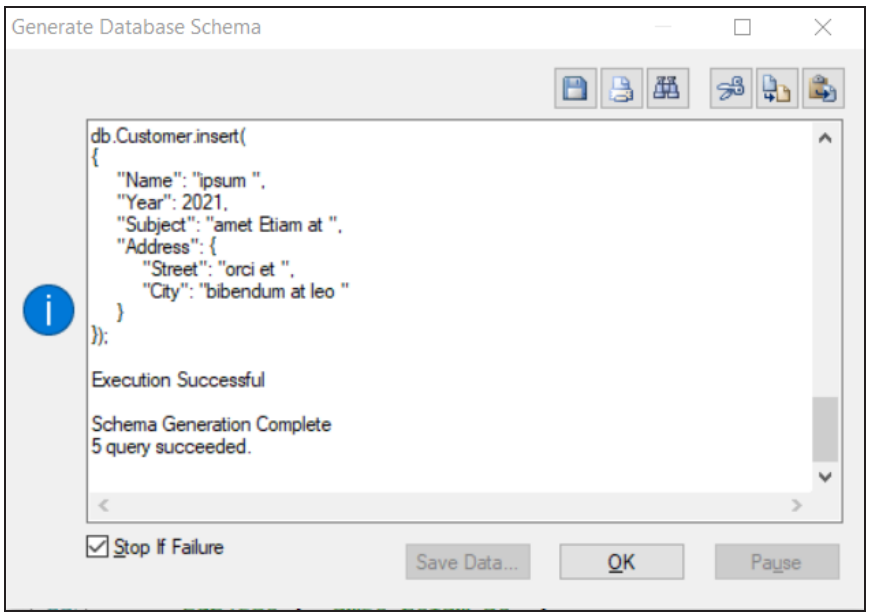


10. In the Preview section, edit the schema to correct the random integer values.

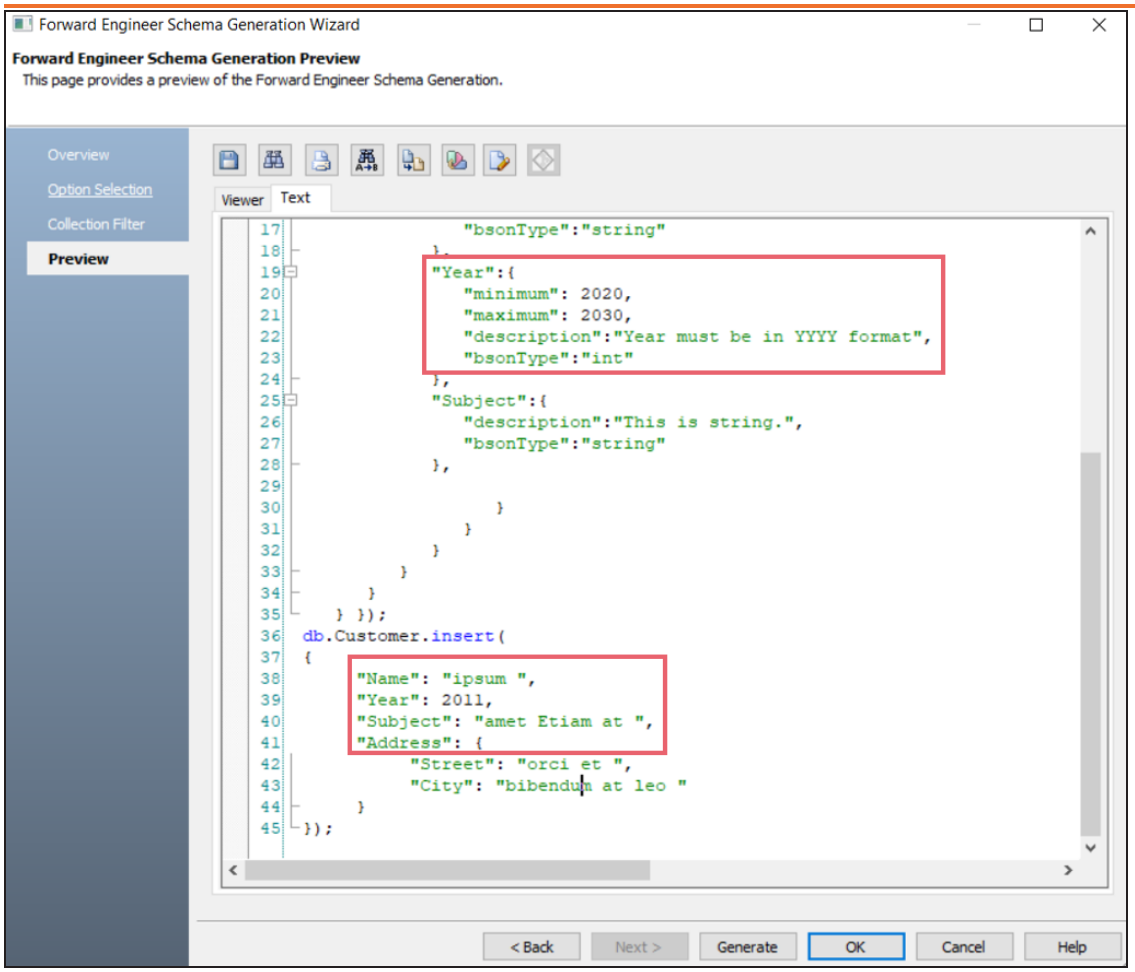
> Generating schema validation template adds random integer values to the fields with INTEGER datatype. Ensure that you replace this with valid value to generate the schema successfully.

11. Click **Generate**.
    The MongoDB Connection screen appears. Connect to the database. For more information, refer to the Database Connection Parameters topic.

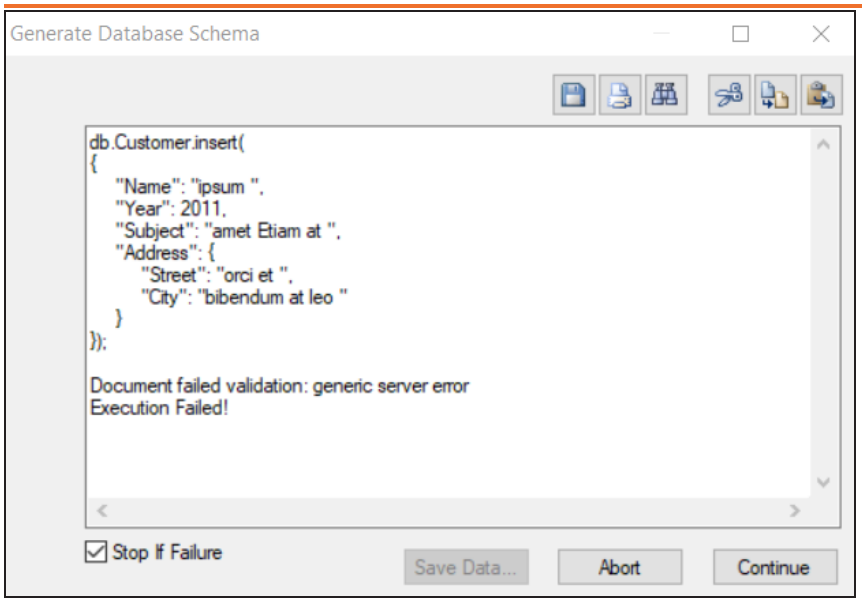
The schema is generated successfully along with the validations.



The schema generation wizard displays errors if the field-level validations are not met. For example, the schema in the following image accepts values between 2020 and 2030 in the Year field. The, Year field is updated with the value 2011.

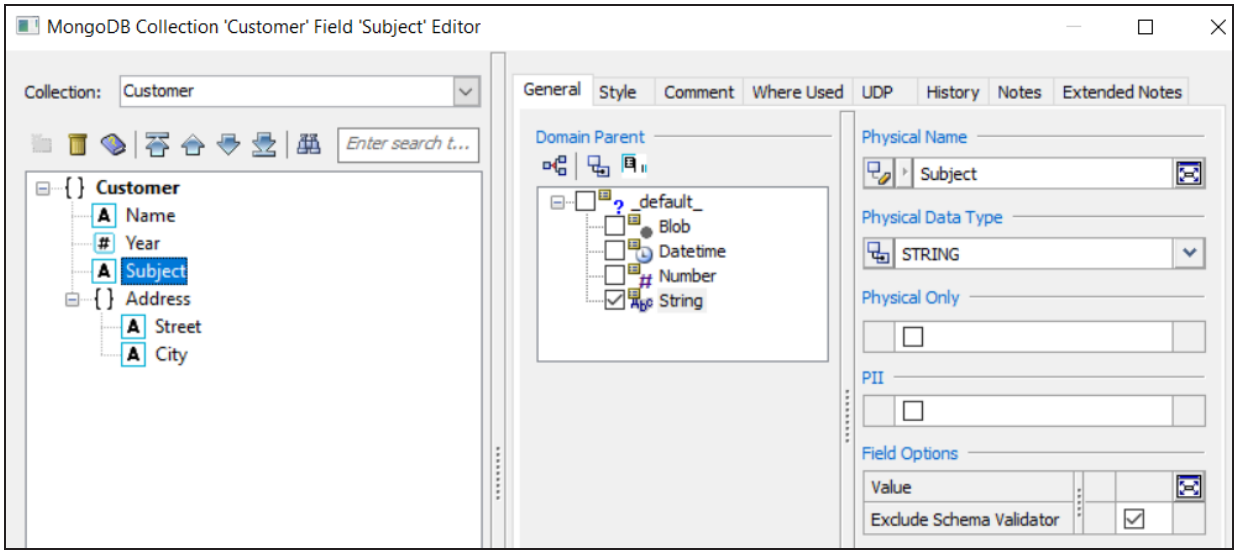


When you click **Generate**, the schema generation fails as the validation rules are not met and displays the following error.

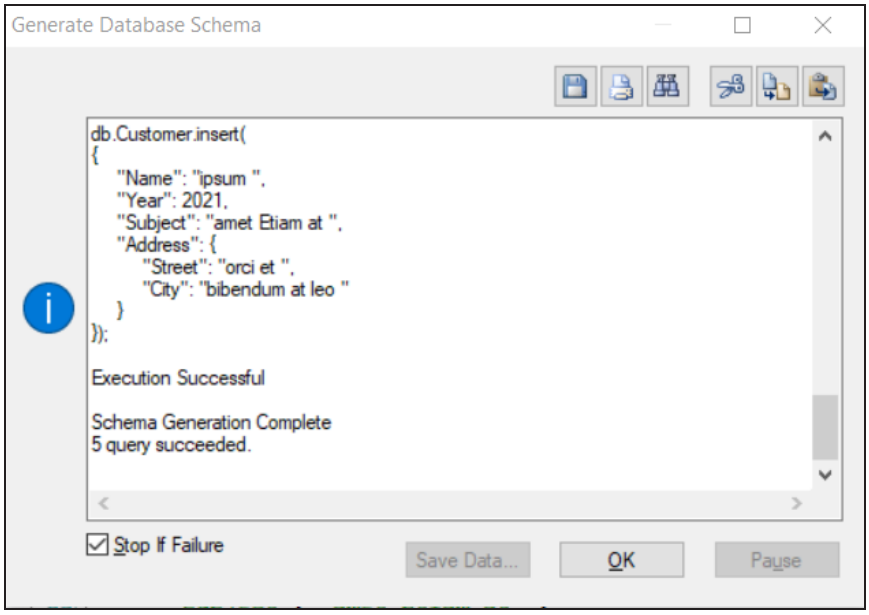


Alternatively, you can exclude validations for one or more fields to generate schema. For example, you can exclude schema validation for the field, Subject (STRING), in this collection. Then, replace a string with a number (INTEGER). This overrides the assigned data type of a field and generates the script successfully.

To exclude schema validation for a field, open Collection Field Editor, select a field, and then, select the **Exclude Schema Validator** check box.



In the Forward Engineering Schema Wizard, when you click **Generate**, the schema is generated successfully.



Snowflake Enhancements

Snowflake support in erwin Data Modeler (DM) 12.0 has been enhanced with the following features:

- Object Tagging
- Table, View, and Materialized View Filters
- Key-Pair Authentication

Object Tagging

Snowflake implementation in erwin DM now supports object tagging via the Tags object for the following objects:

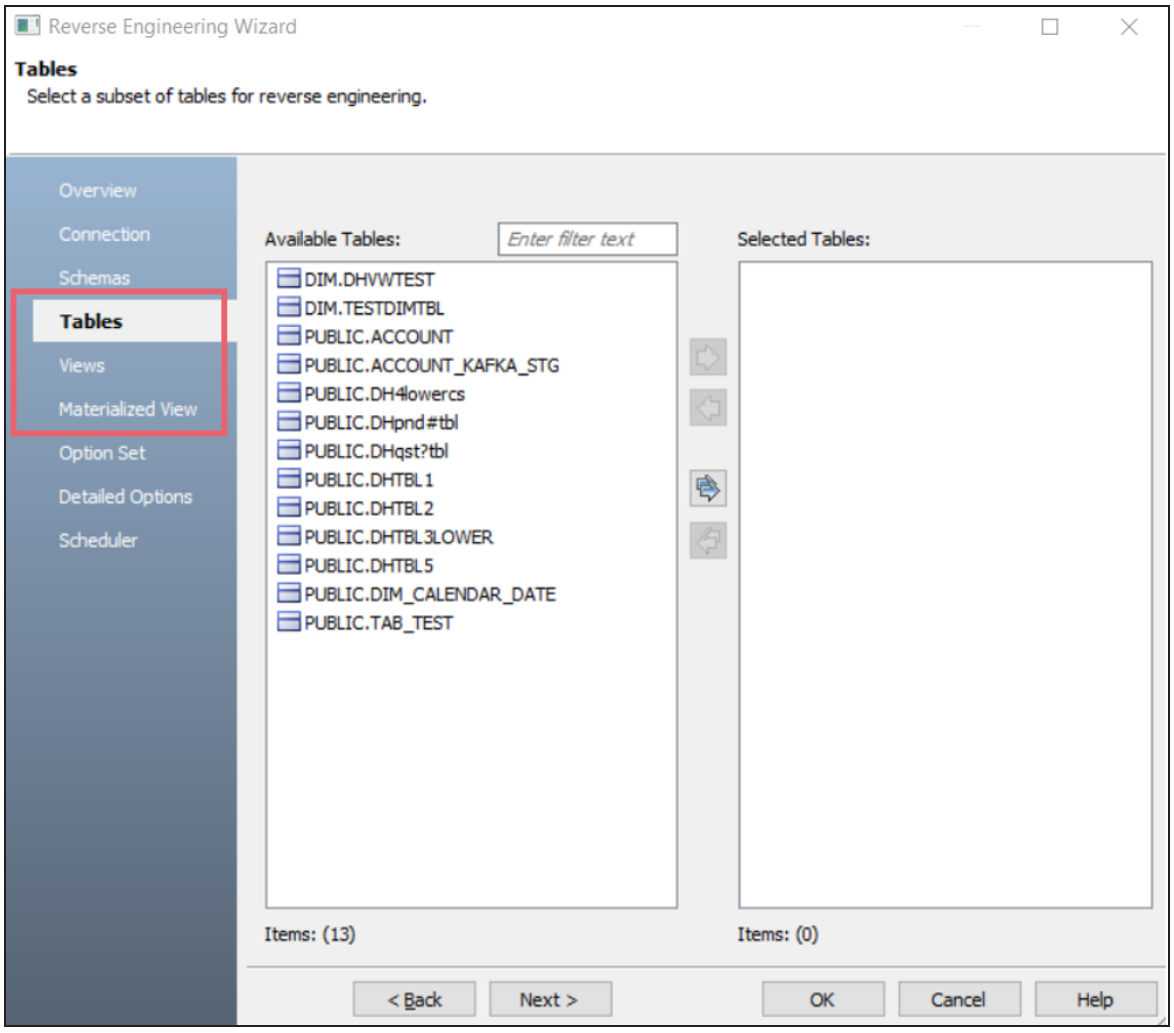
- Table
- Database
- Column
- Materialized View
- Stage
- View
- Role
- User
- Schema
- Warehouse

For more information on object tagging, refer to the Snowflake Support Summary topic.

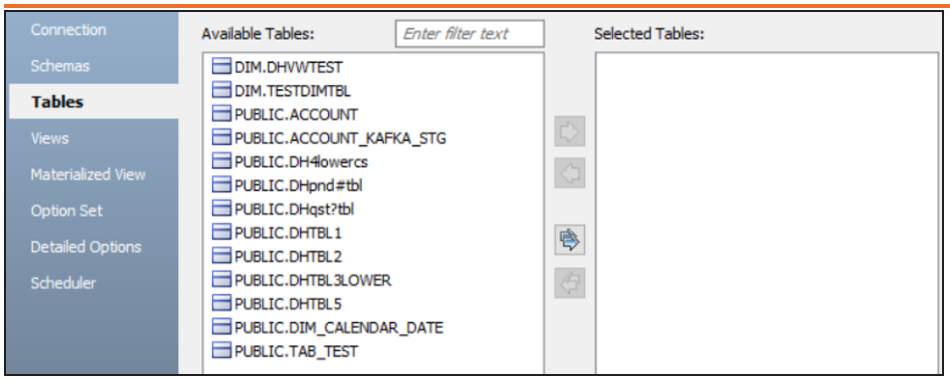
Table, View, and Materialized View Filters

The Reverse Engineering Wizard now includes filters for Tables, Views, and Materialized Views. These filter sections display all tables, views, and materialized views available in your database without having to select a schema during JDBC connection.

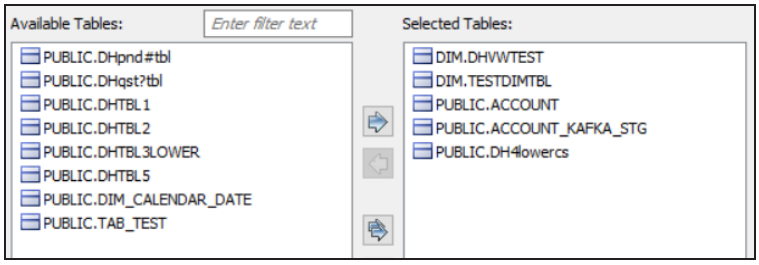
The below image displays the tables available in the database after establishing a database connection. For more information, refer to the Database Connection Parameter topic.



You can select tables, under **Available Tables**, that you want to reverse engineer. Then, click  .



This moves the selected tables under Selected Tables.



Similarly, you can select view and materialized view objects in a database for reverse engineering. For more information on reverse engineering options, refer to the Reverse Engineering Options for Snowflake topic.

Key-Pair Authentication

Snowflake database supports user-based key-pair authentication for reverse engineering and forward engineering models. This method requires a private key, public key, and password to connect to a database.

The following methods are available for authenticating a database connection:

- Authenticate using unencrypted key
- Authenticate using an encrypted key

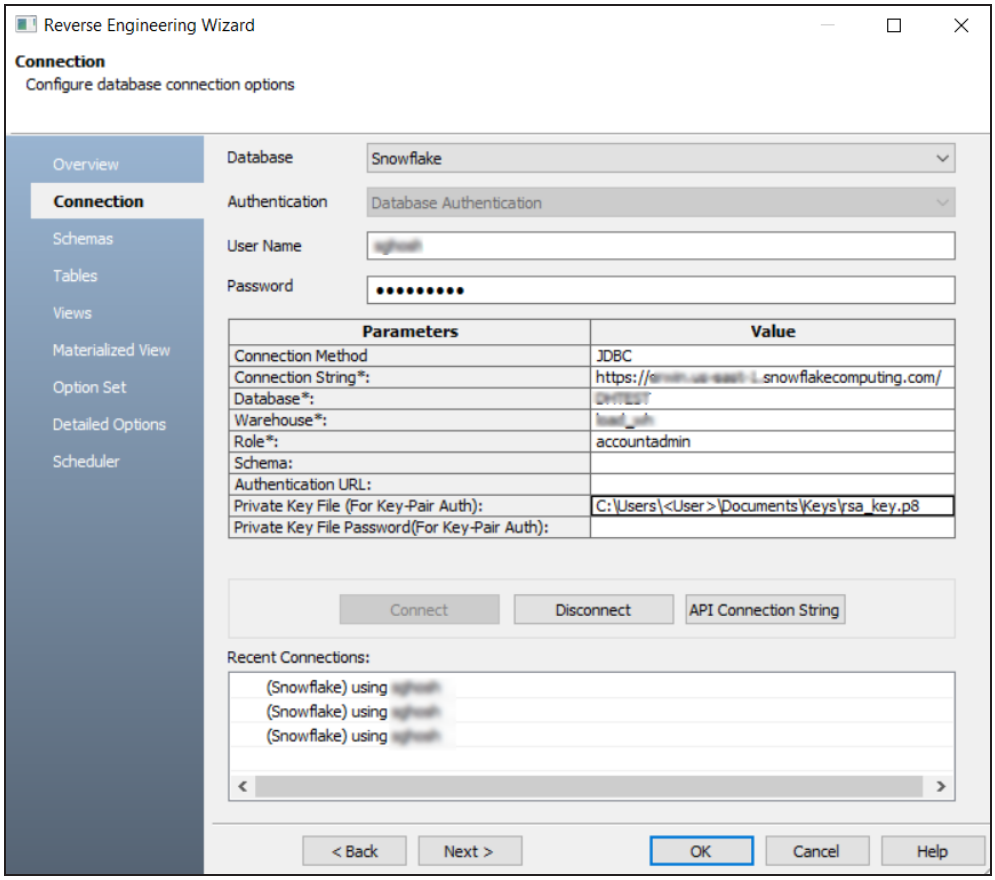
Authentication using Unencrypted Key

To set up a database connection using an unencrypted key, you first need to generate a private key and a public key. For more information on generating keys, refer to the Key Pair

Authentication & Key Pair Rotation topic in the Snowflake Documentation. Then, run the script on the Snowflake console to establish a successful connection.

To authenticate database connection using an unencrypted key, follow these steps:

1. Once the key is generated, save the key in a folder.
2. In erwin DM, specify the file path in the **Private Key File (For Key-Pair)** field while you establish connection during reverse engineering or forward engineering. For more information about database connection parameters, refer to the Database Connection Parameters topic.
   Specify the file path and the file name as shown in the example below.
   C:\Users\<User>\Documents\Keys\rsa_key.p8



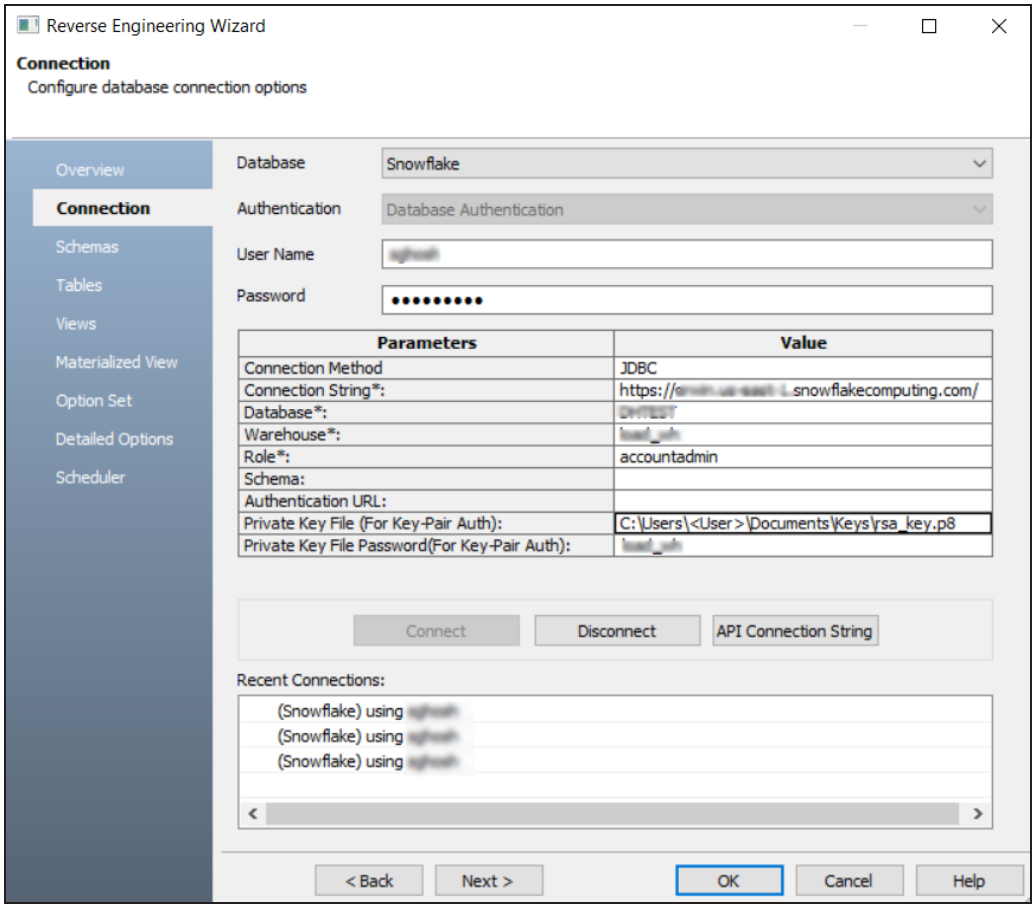
Authentication using an Encrypted Key

To set up a database connection using an encrypted key, you need to generate private key, public key and, set an authentication password. This process also involves creating two private keys that are used to setup authentication using an encrypted key. For more information on generating keys, refer to the Key Pair Authentication & Key Pair Rotation topic in the Snowflake Documentation. Then, run the script in the Snowflake console to establish a successful connection.

To authenticate database connection using an encrypted key, follow these steps:

1. Once the key is generated, save the key in a folder.
2. In erwin DM, specify the file path in the **Private Key File (For Key-Pair Auth)** field while you establish connection during reverse engineering or forward engineering. For more information about database connection parameters, refer to the Database Connection Parameters topic.
   Specify the file path and the file name as shown in the example below.
   C:\Users\<User>\Documents\Keys\rsa_key.p8

3. Enter the password for the Private Key in the **Private Key File Password (For Key-Pair Auth)**.



PostgreSQL Certification

erwin Data Modeler (DM) and erwin Mart Server 12.0 are now certified to work with PostgreSQL versions as follows:

- **erwin DM**: Versions 9.6.24, 10.20, and 11.14
- **erwin Mart Server**: Versions 9.6.24, 10.20, 11.14, 12.9, 13.5, and 14.1

Cassandra: Deriving Models and Advanced Denormalization

You can now perform advanced denormalization when you derive a Cassandra model. This feature provides you with options to do a manual or automatic denormalization.

- Use the auto-denormalization option to merge tables with the target table automatically. This embeds the tables with one-one relationships as User Defined Type and one-to-many relationships as normal columns.
- Use the manual denormalization option to selectively merge columns from source tables to target tables. Further, manual denormalization provides you options to merge multiple columns into single combined column, decide the column embedding type and the embedding process, retain relationships and much more.

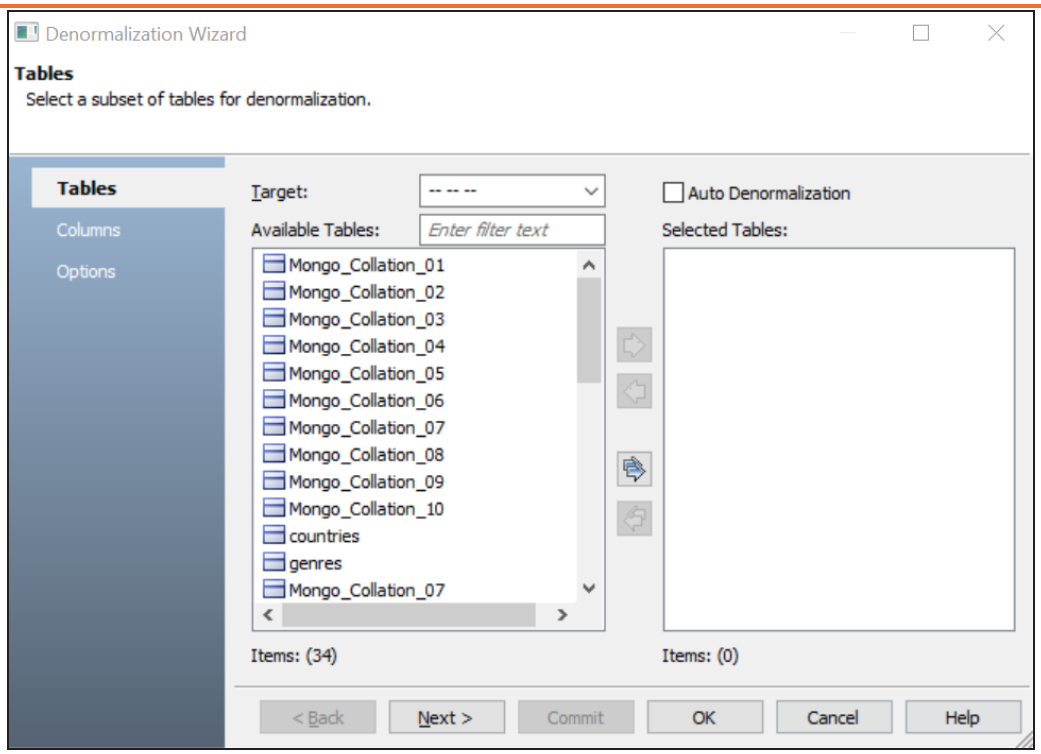
Advanced Denormalization

You can select table and column subsets for denormalizing Cassandra database after deriving a model. Using the Advanced denormalization option, you can merge the source tables and columns with the target based on the requirement.

The denormalization options for Cassandra appear only when the Advanced Denormalization option is selected while deriving a model. Once the model is derived, the Denormalization Wizard for Cassandra model appears and has different sections. By default, Table section is displayed.

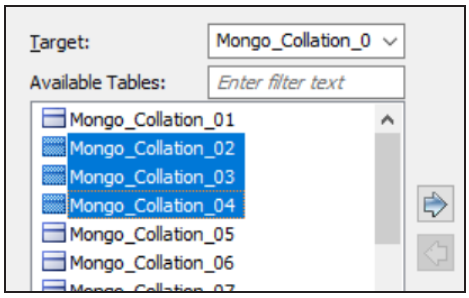
To denormalize the model further, follow these steps:

1. On the Tables section, click the **Target** drop-down to select a target table. All the tables will be merged into the target table.

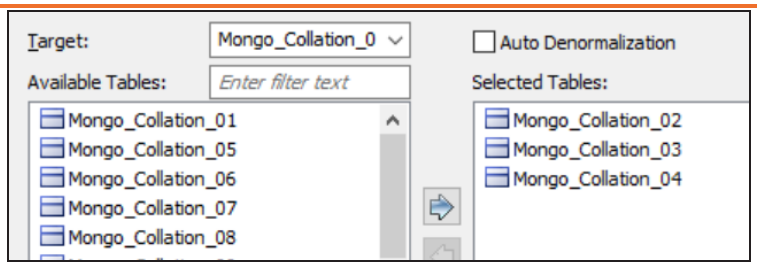


Select **Auto Denormalization** option to merge tables with the target automatically. This embeds the tables in the model with one-one relationships as User Defined Type and one-to-many relationships as normal columns.

2. Under **Available Tables**, select the that you want to merge. Then, click .

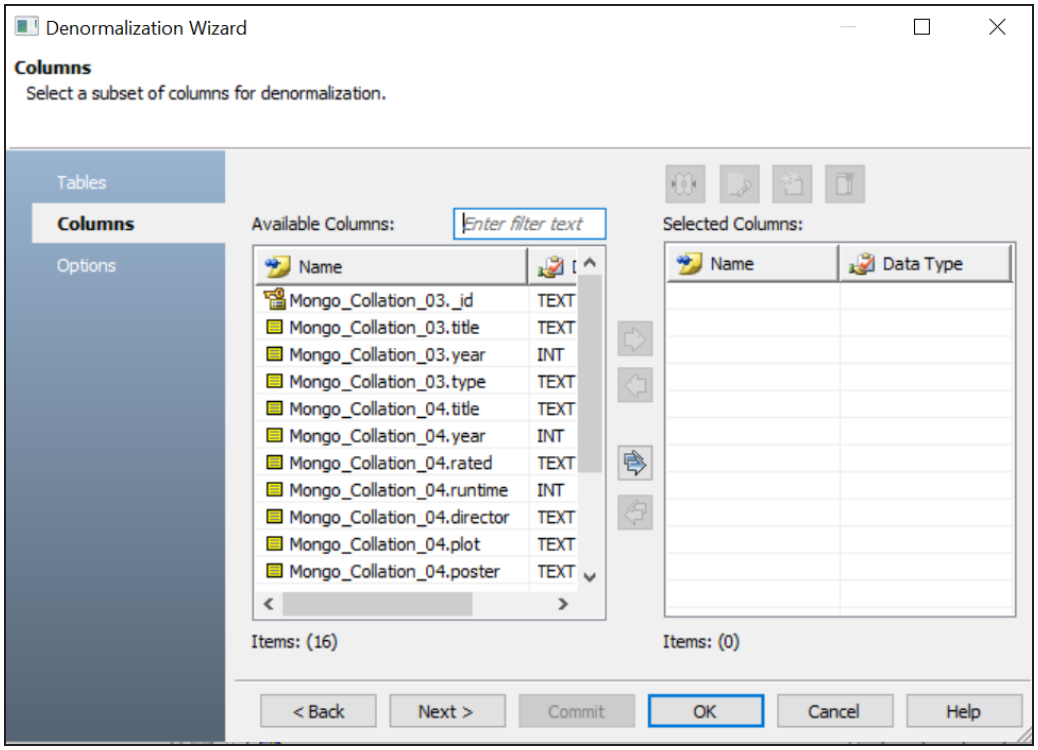


This moves the selected tables under Selected Tables.

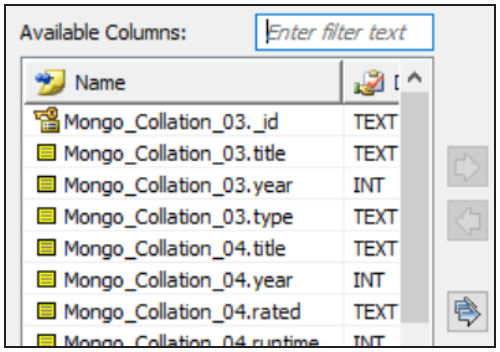


3. Click **Next**.

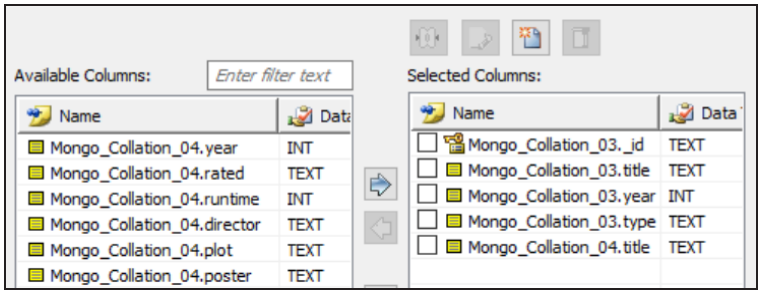
   The Column section appears. It displays a list of available columns.



4. Under **Available Columns**, select the that you want to merge. Then, click .



This moves the selected databases under Selected Columns.



Once you have added the selected columns, you can use any of the following:

New ( )

Use this option to add a new column under Selected Columns.

Update ( )

Use this option to edit column details such as column name, domain parent, and data type for a selected column.

Merge ( )

Use this option to merge the selected columns and create a new column under Selected Columns.

Delete ( )

Use this option to delete to the selected columns.

5. Click **Next**.
   The Options section appears.
6. Select an **Embedding Type**.

   You can select the following embedding options:

   - **Embed as Auto**: Use this option to embed tables through an auto-mechanism based on one-to-many and one-to-one relationships.
   - **Embed as Normal**: Use this option to embed collections using normal column styles.
   - **Embed as UDT**: Use the option to embed collections using a User Defined Type (UDT) styles.
7. Select **Relationships** option to include table relationships to the model.
8. Select **Cascading** options to determine how multiple collections are merged into a single collection.
   Use the following cascading options:

   - **All**: Use this option to denormalize all relationship levels in a collection into a single collection.
   - **Levels**: Use the option to specifies the number of levels up to which collections are denormalized into one collection. For example, if you set Level to 1, all the collections up to level 1 in the relationship hierarchy will be denormalized into a single collection.
   - **Auto Cleanup**: Use the option to delete the source collection after denormalization.

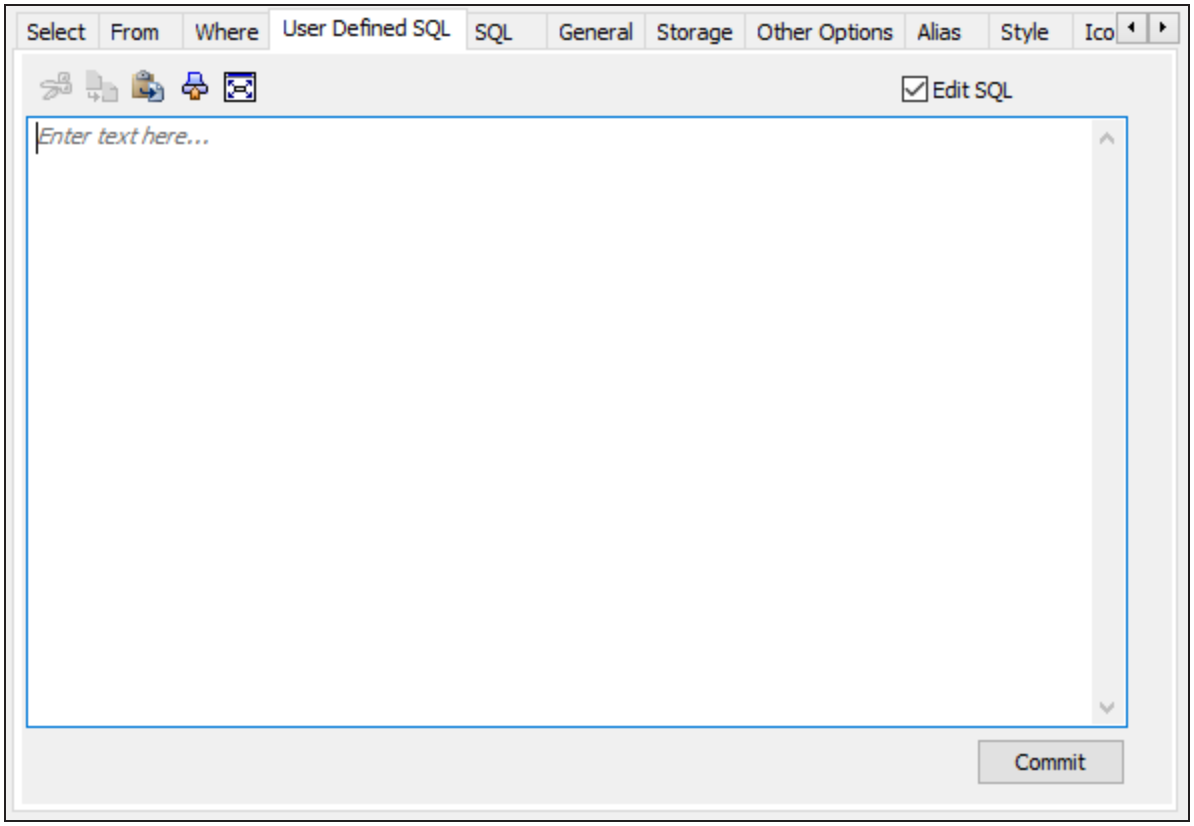
   Alternatively, click **Commit** to apply changes to the model without exiting the Denormalization Wizard.
9. Click **OK**.

The denormalization process starts and displays collections based on the denormalization option.

Oracle: View and Materialized View Enhancement

For Oracle models with views and materialized views that have JOINS, GROUP BY and CTE clauses and/or wildcards, you can now run Reverse Engineering from Script (RES) without hampering the resulting model. Such views and materialized views now result into objects with appropriate columns and relationships with tables.

For complex views and materialized views, use the User Defined SQL tab to view and change a user-provided DDL statement.



Edit SQL

Select the check box to change the SQL code in the SQL Statement box. Select this check box only if you want the object to contain syntax that erwin Data Modeler cannot represent, for example, a UNION statement. Or Views and Materialized Views

with JOINS, GROUP BY and CTE clauses and/or Wildcards.

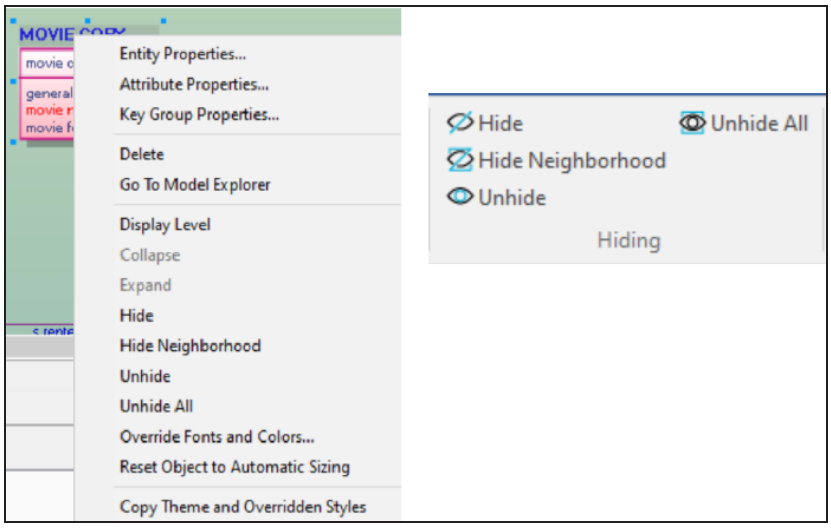
When you select this check box, you no longer maintain references to the base tables and columns to which the object refers. After updating the SQL code, click **Commit**.

Azure Synapse: Table Constraint Enhancement

For Azure Synapse models, you can now process Table Constraints via Reverse Engineering from Script (RES).

Diagramming: Hide and Unhide Diagram Nodes

For complex models with many nodes in the diagram, the diagram menu provides hide and unhide options. These options enable you to selectively view or hide Entity, View, and Materialized View nodes from the complex and large diagram and focus only on necessary nodes. These options are also accessible via right-clicking the node.



For Neo4j database, you can hide or unhide nodes only in the orthogonal lay-out. Also, you cannot hide the Supertype or Subtype Entity and isolated nodes.

To hide or unhide nodes

1. Open the diagram in which you want to hide or unhide nodes.
2. Select one or multiple nodes.
3. On the diagram menu, work with the following available options:

   Hide

   Hides a single or multiple selected nodes in the diagram.

   Hide Neighborhood

Hides a single selected node and all its neighboring nodes in the diagram.

Unhide

Unhides the neighboring hidden nodes of a single selected node with the visual hiding cue () in the diagram. This option resets the selected objects to their default sizes.

Unhide All

Unhides all the hidden nodes that the single selected node can reach in the diagram or unhide all the hidden nodes when no node is selected in the diagram.

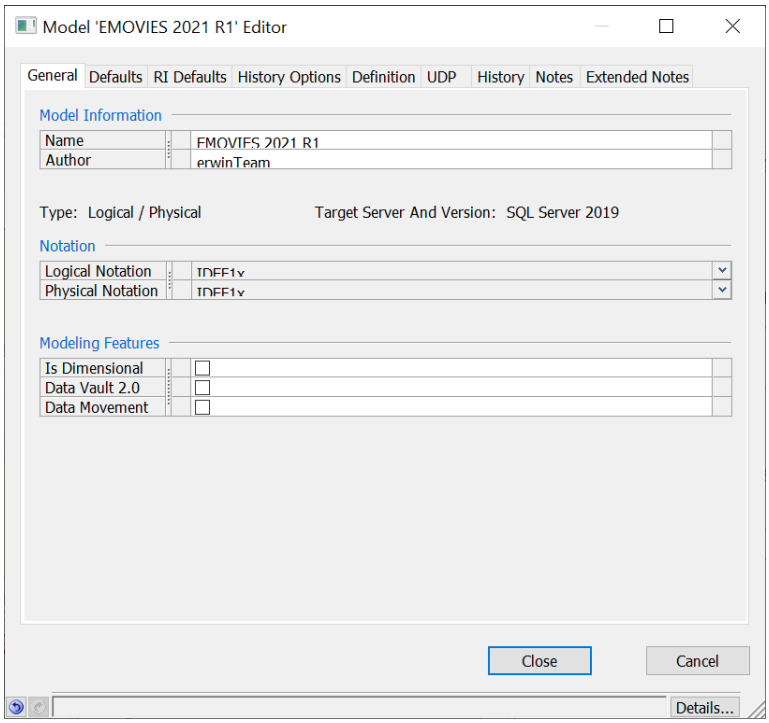

You can also access these options by right-clicking the nodes.

4. Click Save on the File menu.

Your diagram is saved and can be retained whenever you open a model.

Data Vault Enhancements

erwin Data Modeler (DM) now supports model-level and table-level rollback function for Data Vault models. To rollback your model or table to its earlier state, open **Model Editor** > **General** tab and clear the **Data Vault 2.0** check box. This restores your model or table to its earlier state.



Additionally, the Data Vault Component Type selector is now available only when the model is configured to be a Data Vault model.

For more information, refer to the Data Vault topic.

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:

- Copy Neighborhood
- Object Browser
- Column Editor Shortcut
- Graph Display Level
- Denormalization and deriving models now creates new models instead of overwriting source models.
- Generate diagram picture in multiple formats
- Enhanced HTML report

Copy Neighborhood

You can copy neighboring objects of an object in a diagram and paste it to a different model.

To copy neighboring objects to a different model

1. Open a diagram and select an object of which you want to copy neighboring objects.
2. In Home menu, click Copy Neighborhood.

   The neighboring objects are copied.
3. Open the diagram to which you want to copy the objects.
4. Click Paste.

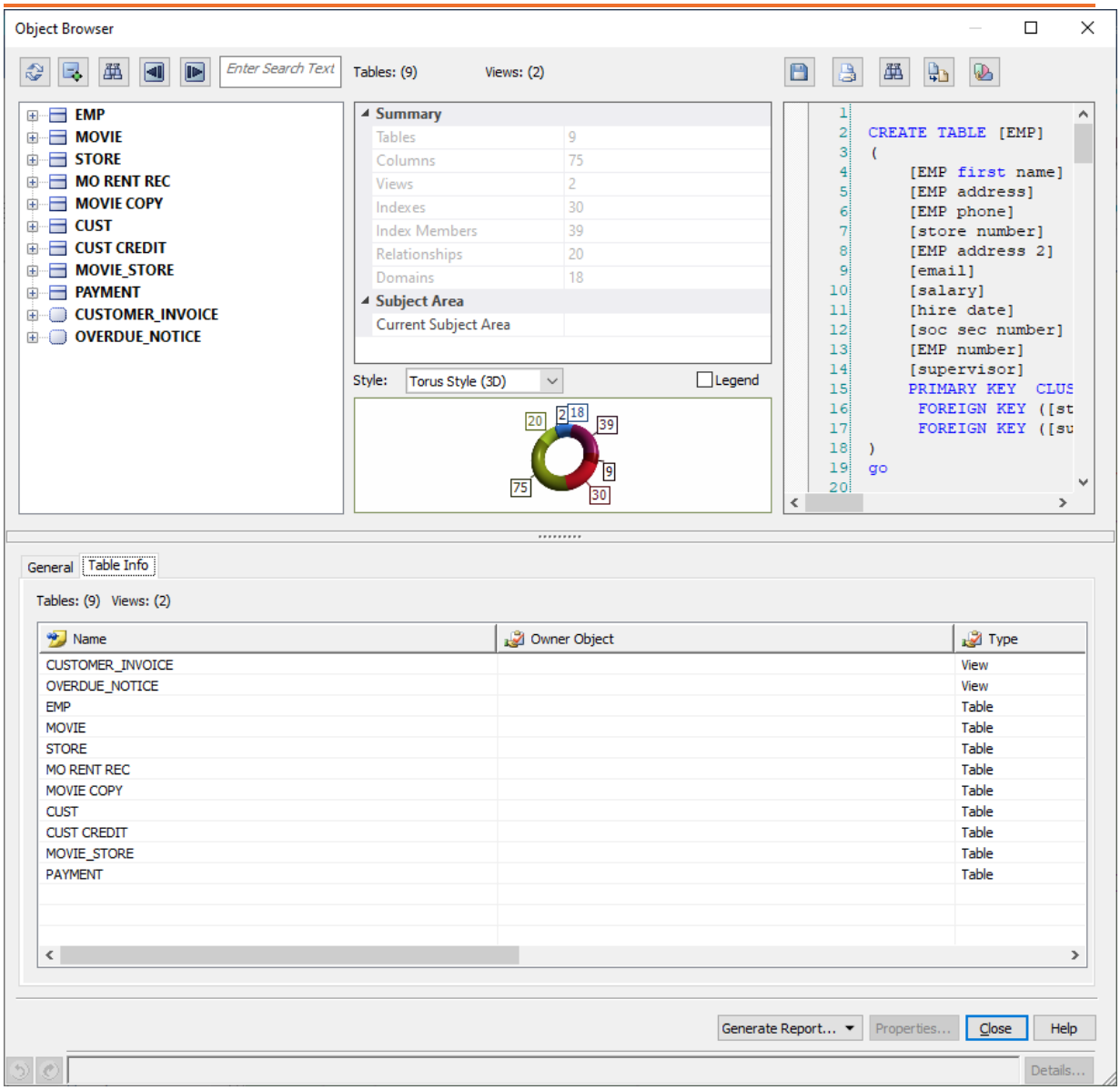
   If none of the neighboring objects exist in the source diagram, the selected object is copied and can be added to new model.
   If one or more selected objects exist in the target diagram, a message appears informing you that the objects are pasted as new objects in the model. The new objects follow the naming standards of the model, and are displayed in the same location as in the source diagram. In addition, the new objects appear in the Model Explorer.

If none of the selected objects exists in the target diagram, the selected objects are only displayed in the diagram. They are not added to the model as new objects.

Cut or copied objects remain on the clipboard even after you paste them into another location. This is convenient if you want to paste more than one copy. But, if you have a large copy selection on the clipboard, it can take up too much memory. To free up memory, after you finish copying and pasting, select an entity and copy it to the clipboard to replace the large copy set.

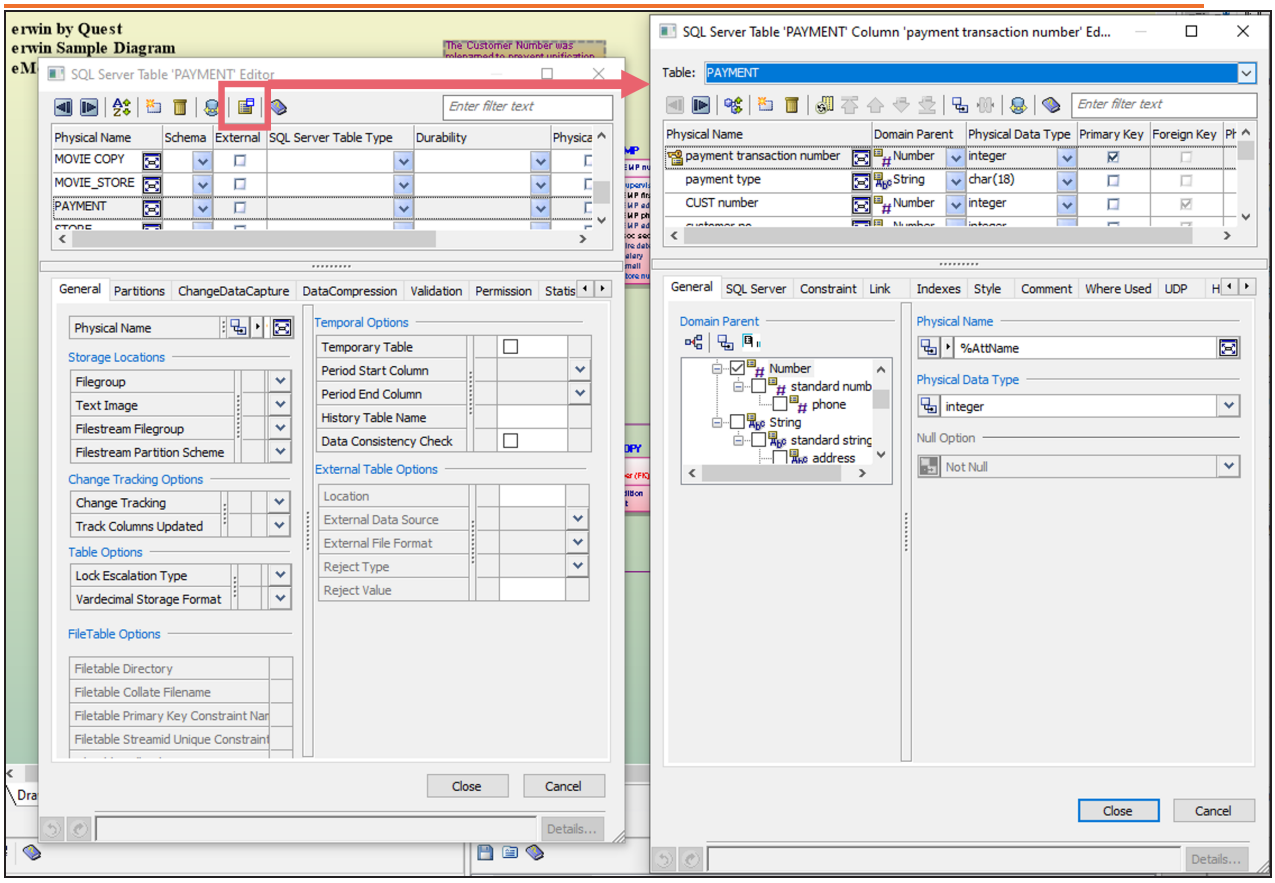
Object Browser

- A new tab, <Object> Info has been added for all databases. It displays tables, records, collections, documents, JSON Objects, and nodes, in your model or in the selected object along with the counts based on database. Apart from this, for Couchbase and Neo4j models, database-specific tabs have been added:
  - The Couchbase tab displays global indexes and full text indexes along with their counts.
  - The Neo4j tab displays global indexes and global constraints along with their counts.



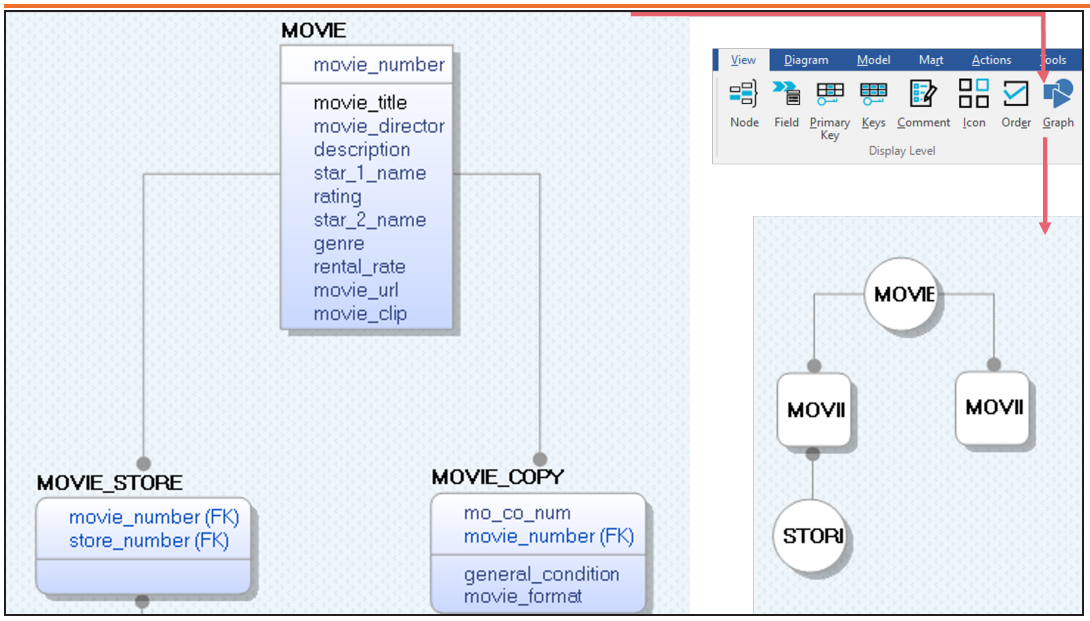
Column Editor Shortcut

You can now access the columns of a table via the table editor instead of having to open the column editor explicitly. Use the icon on the table editor.



Graph Display Level

A new display level, Graph, has been added to facilitate easier switch for graph databases. Derived NoSQL graph models have table-like representation by default. To convert such models to graph-like representation, on the ribbon, go to **View** > **Display Level** group. Then, click  . This converts the model diagram as follows:



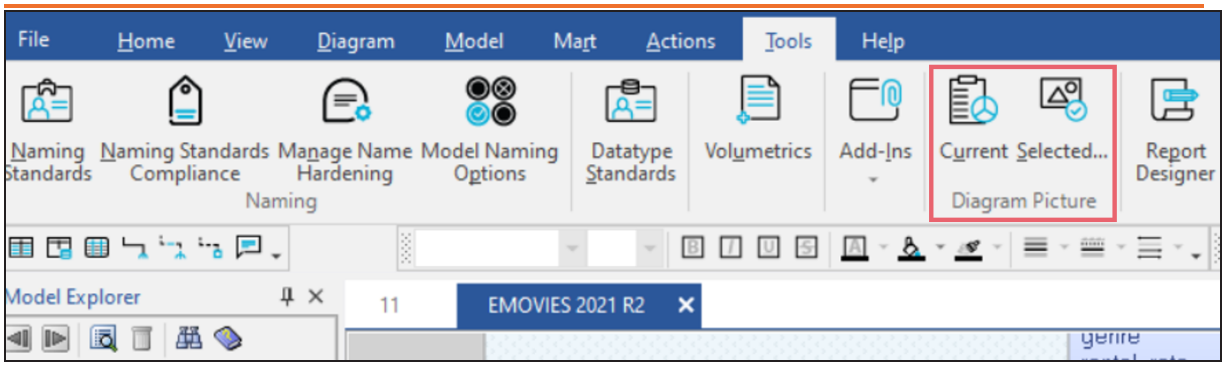
Generate Diagram Picture in Multiple Formats

You can now generate picture reports of a single or multiple diagrams in one submission in the following formats:

- Enhanced Metafiles (.emf)
- PNG (.png)
- JPG (.jpg)
- SVG (.svg)
- PDF (.pdf)

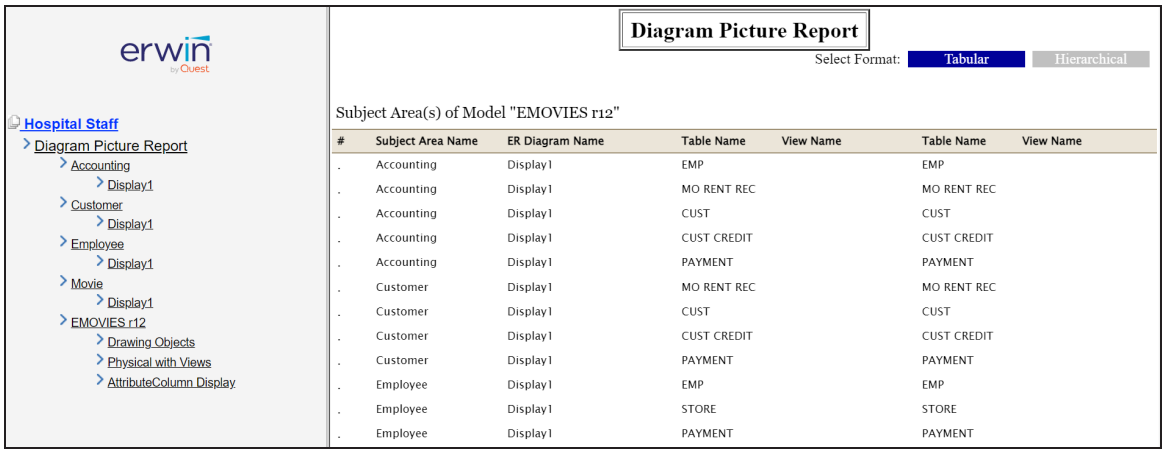
To generate diagram picture, open a model, go to **Tools** > **Diagram Picture**. Then, select either of the following option to generate picture:

- Click **Current** to generate a single picture diagram of you current model. For more information, refer to the Generate Current Diagram Picture topic.
- Click **Selected** to generate multiple diagrams based on your selection. For more information, refer to the Generate Multiple Diagram Pictures topic.



HTML Report

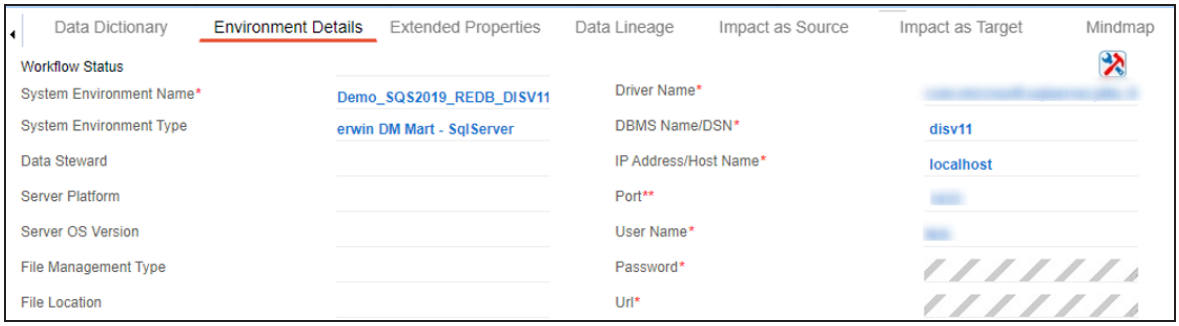
You can now generate an enhanced report in the HTML format. To generate reports, open a model, go to **Tools** > **Report Designer**. The **erwin Report Designer** page appears. Use erwin Report Designer to generate reports in the HTML format. For more information, refer to the Create Reports Using Report Designer topic.



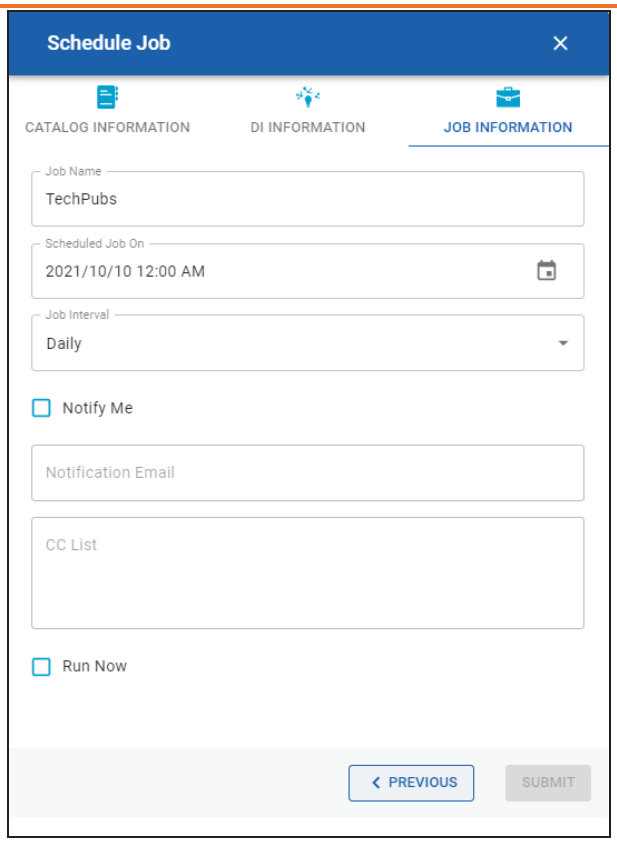
DM Connect for DI

DM Connect for DI has undergone several enhancements as follows:

- The REDB process now stores the database connection parameters, such as DBMS Name/DSN, IP Address/Host Name, and Port under Environment Details in erwin Data Intelligence Suite.



- You can now run jobs immediately using the Run Now feature.



- DM Connect for DI has been upgraded to support:
  - all new databases
  - erwin Data Intelligence Suite (DI Suite) v10.2, v11.0, and v11.1

erwin Mart Server Enahncements

erwin Mart Server has undergone several enhancements as follows:

- You can now test LDAP connections using the erwin Mart Configuration screen.
- Session timeout has been updated to 30 minutes.
- Special characters support has been updated.
- Configuration to use IIS and SSL has been updated.