erwin Data Modeler

Feature Tour

Release 12.0

Legal Notices

Contact erwin

Understanding your Support

Review support maintenance programs and offerings.

Registering for Support

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

Accessing Technical Support

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

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

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

Provide Feedback

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

erwin Data Modeler News and Events

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

Contents

Introduction

The Feature Tour guide walks Data Architects, Data Administrators, Application Administrators, Database Administrators, and Partners through the features introduced in erwin Data Modeler (DM) 12.0 release.

The features and enhancements introduced in this release are:

- ArangoDB
- Amazon Keyspaces
- Google BigQuery
- DynamoDB
- Neo4j
- Parquet
- Databricks
- PostgreSQL
- Couchbase 7.0
- Central Scheduler
- Google BigQuery
- MongoDB: Schema Validation
- Snowflake Enhancements
- Git Integration
- 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:

ArangoDB Support

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

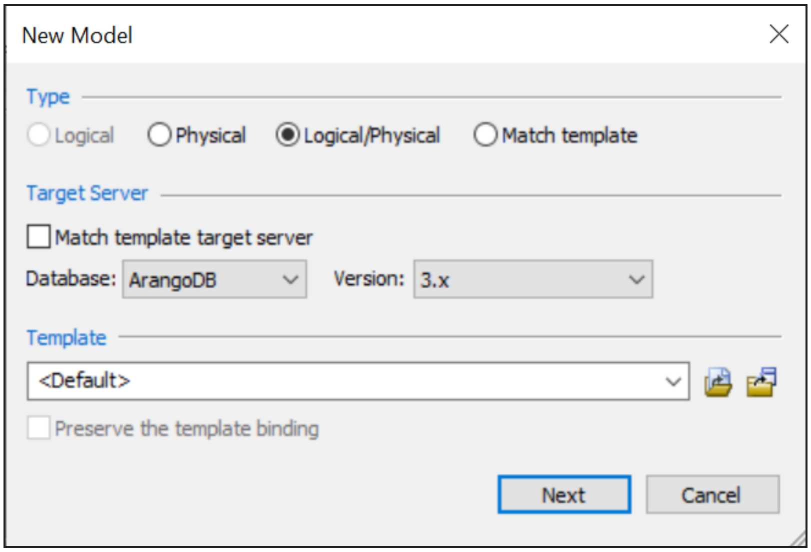
Reverse Engineering Models

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 each reverse engineering option, 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.



New Model

Type

○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

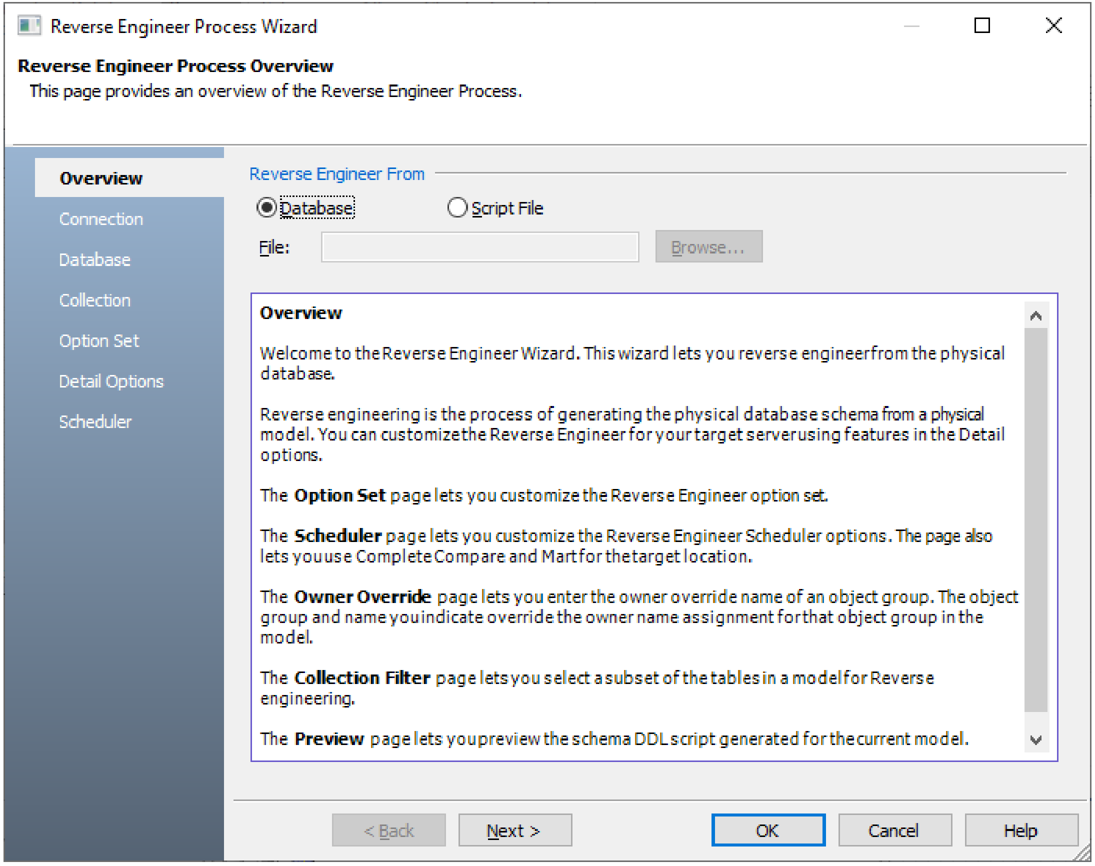
Database: ArangoDB Version: 3.x

Template

<Default>

□ Preserve the template binding

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



Reverse Engineer Process Overview

This page provides an overview of the Reverse Engineer Process.

Overview

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

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

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

The **Scheduler** page lets you customize the Reverse Engineer Scheduler options. The page also lets youuse Complete Compare and Martfor thetarget location.

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

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

The **Preview** page lets youpreview the schema DDL script generated for thecurrent model.

4. Click one of the following options:

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

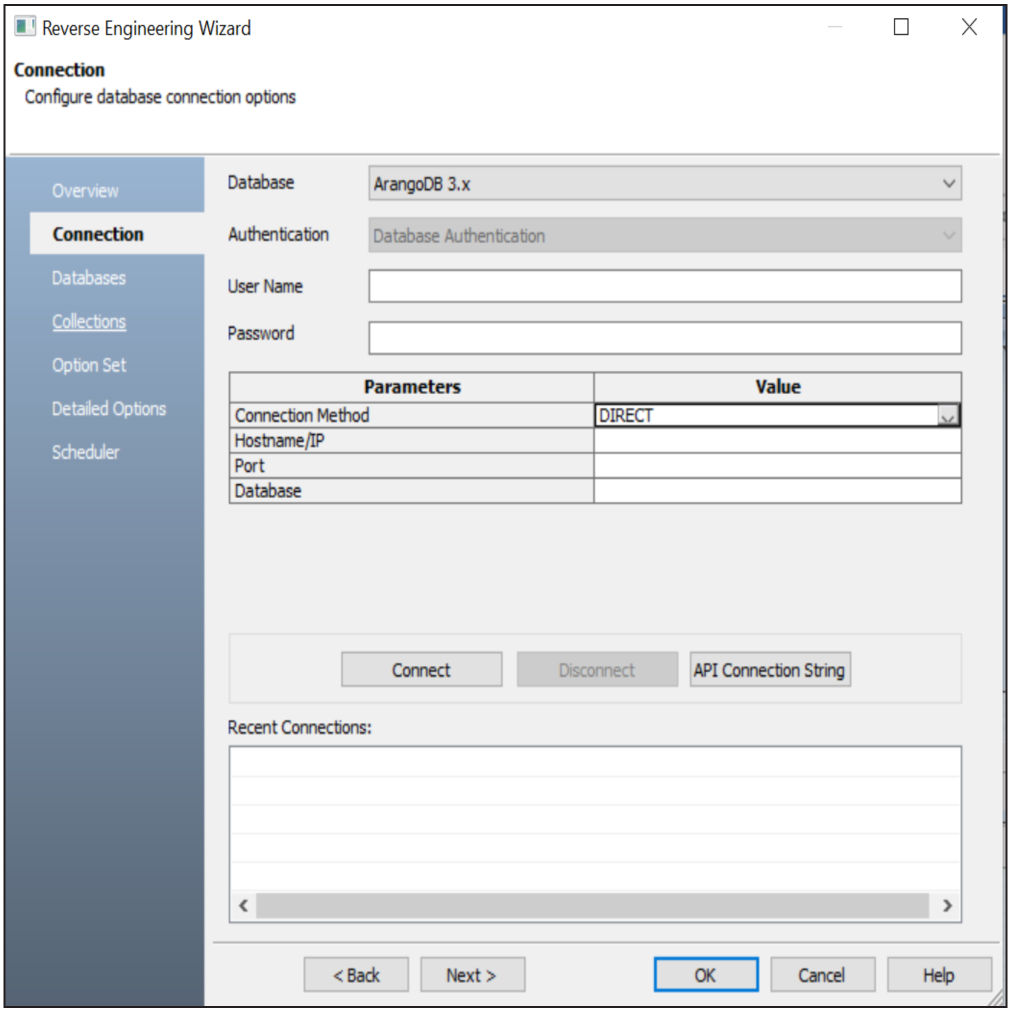
     If you click **Database**, continue to step 5.

   - **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 section appears.



Reverse Engineering Wizard

Connection

Configure database connection options

Database: ArangoDB 3.x

Authentication: Database Authentication

User Name:

Password:

Parameters	Value
Connection Method	DIRECT
Hostname/IP	
Port	
Database	

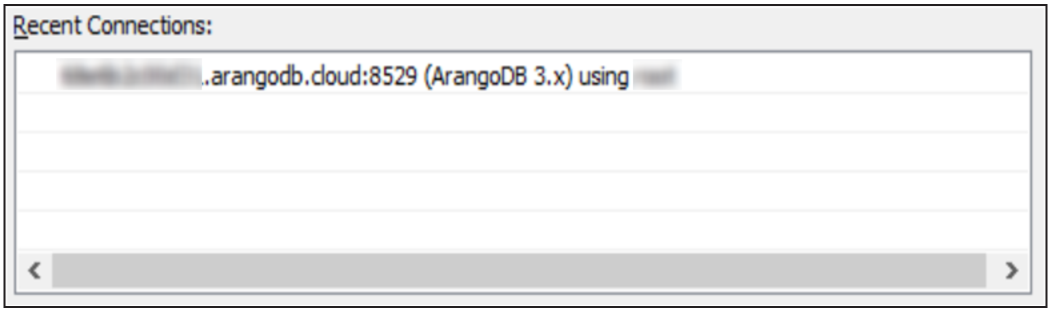
Recent Connections:

Use this section 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	

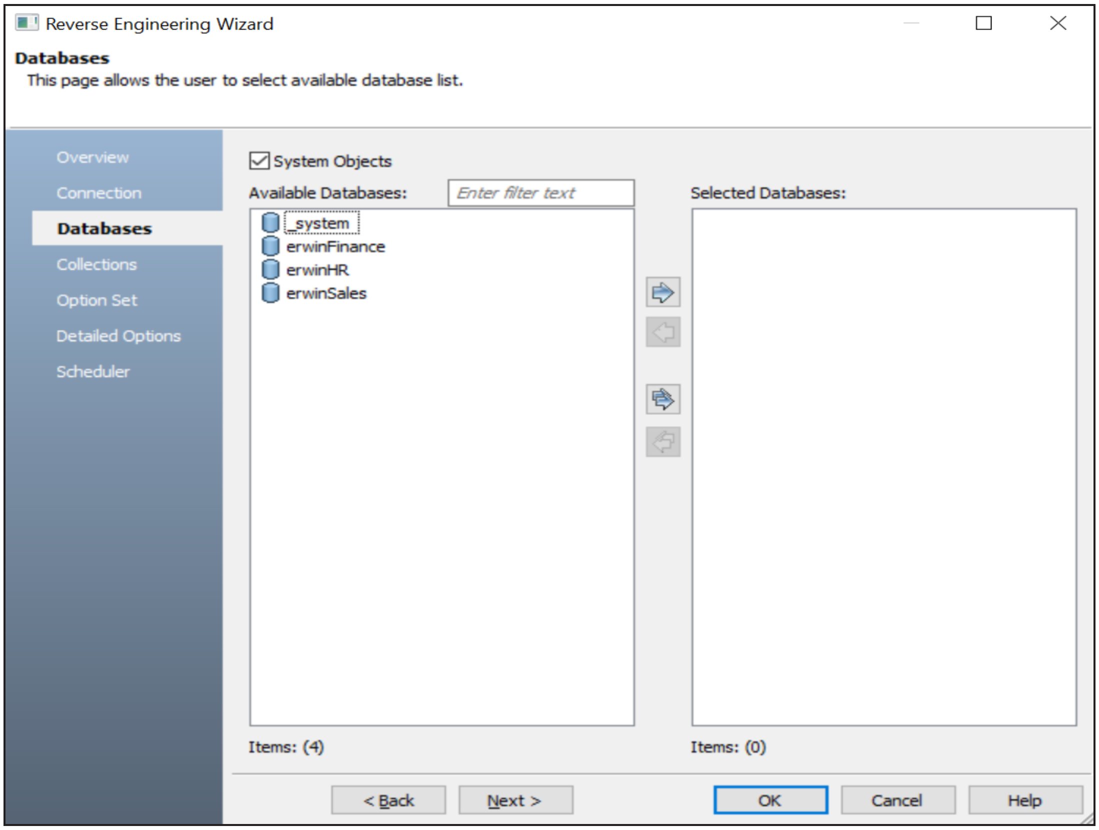
6. Click **Connect**.
   On successful connection, your connection information is displayed under Recent Connections.



[redacted].arangodb.cloud:8529 (ArangoDB 3.x) using [redacted]

7. Click **Next**.

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



Reverse Engineering Wizard

Databases

This page allows the user to select available database list.

☑ System Objects

Available Databases: Enter filter text

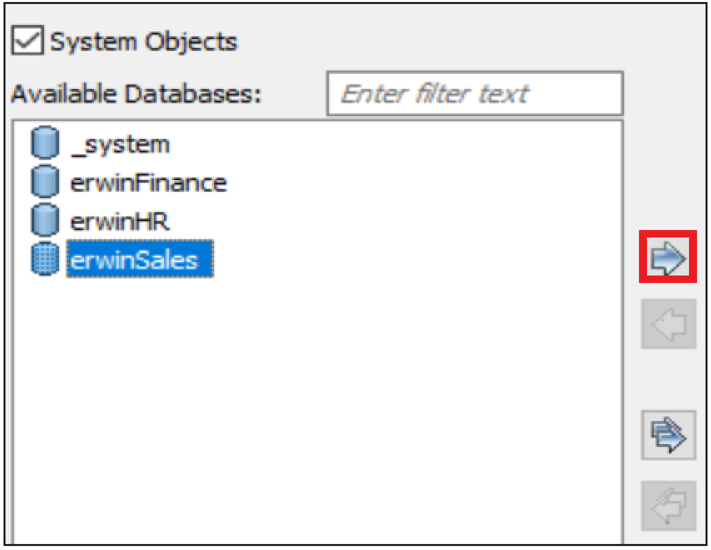
- _system
- erwinFinance
- erwinHR
- erwinSales

Items: (4)

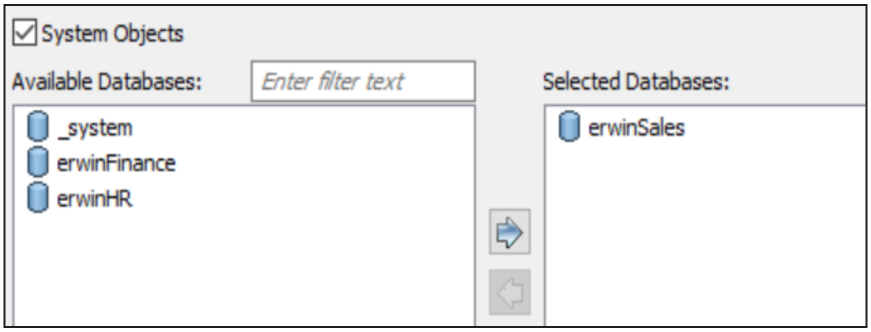
Selected Databases:

Items: (0)

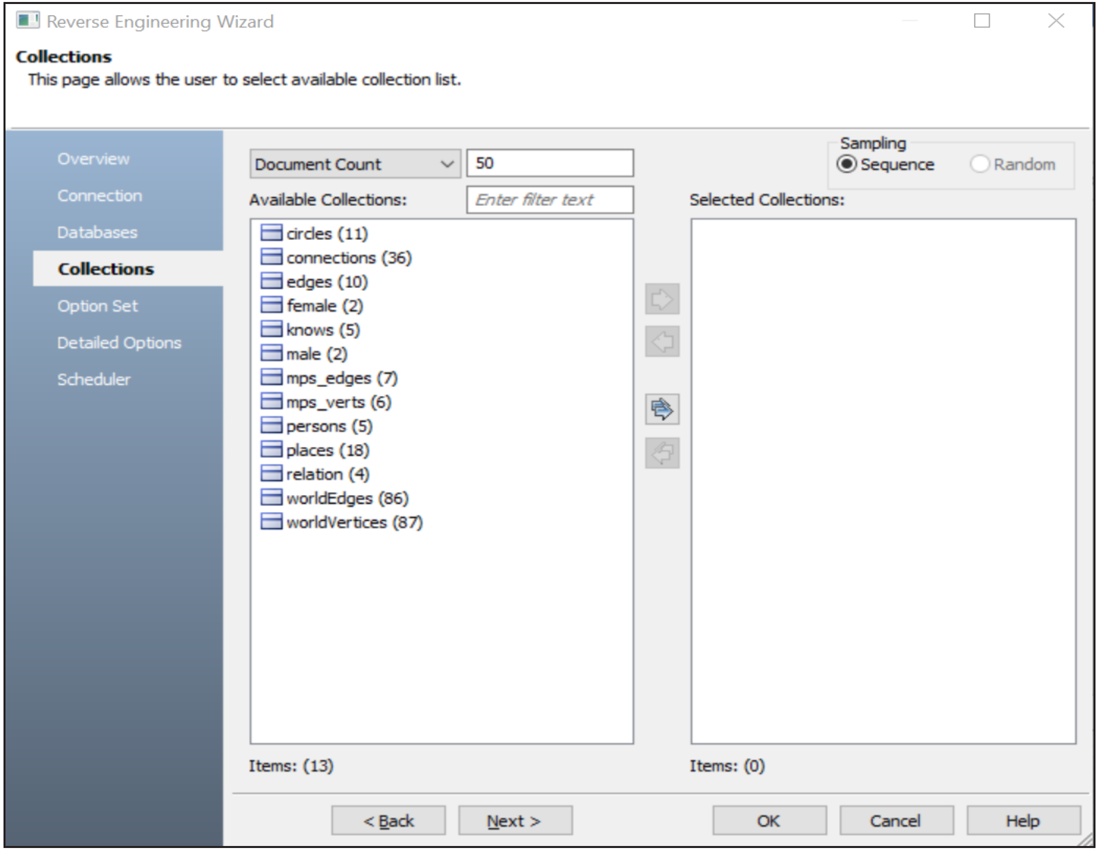
8. Under **Available Databases**, select the databases that you want to reverse engineer. Then, click ⇨.



This moves the selected databases under Selected Databases.



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



Reverse Engineering Wizard

Collections

This page allows the user to select available collection list.

Document Count: 50

Sampling: ● Sequence ○ Random

Available Collections: Enter filter text

- circles (11)
- connections (36)
- edges (10)
- female (2)
- knows (5)
- male (2)
- mps_edges (7)
- mps_verts (6)
- persons (5)
- places (18)
- relation (4)
- worldEdges (86)
- worldVertices (87)

Items: (13)

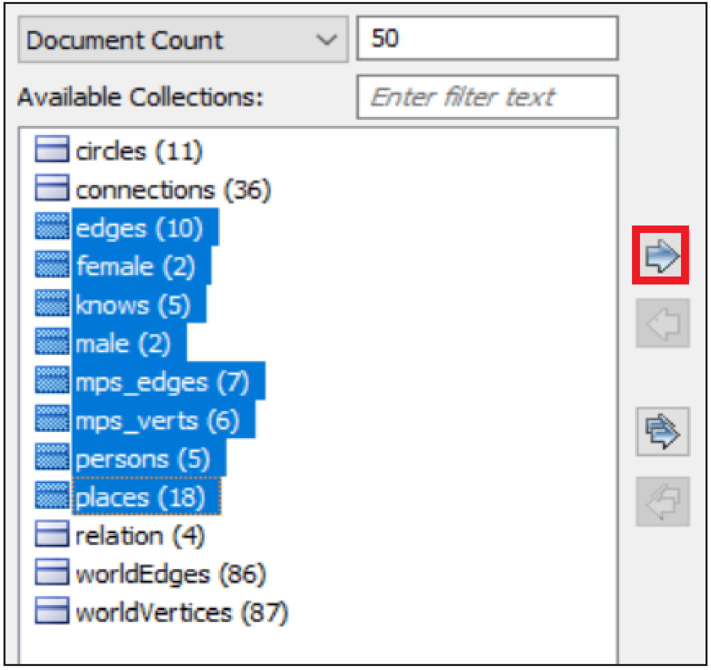
Selected Collections:

Items: (0)

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 ⇨.

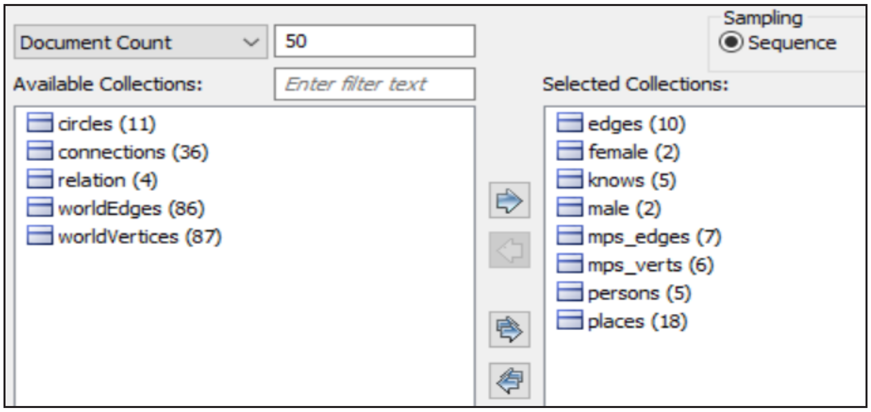


Document Count 50

Available Collections: Enter filter text

circles (11)
connections (36)
edges (10)
female (2)
knows (5)
male (2)
mps_edges (7)
mps_verts (6)
persons (5)
places (18)
relation (4)
worldEdges (86)
worldVertices (87)

This moves the selected collections under Selected Collections.



Document Count 50

Sampling
● Sequence

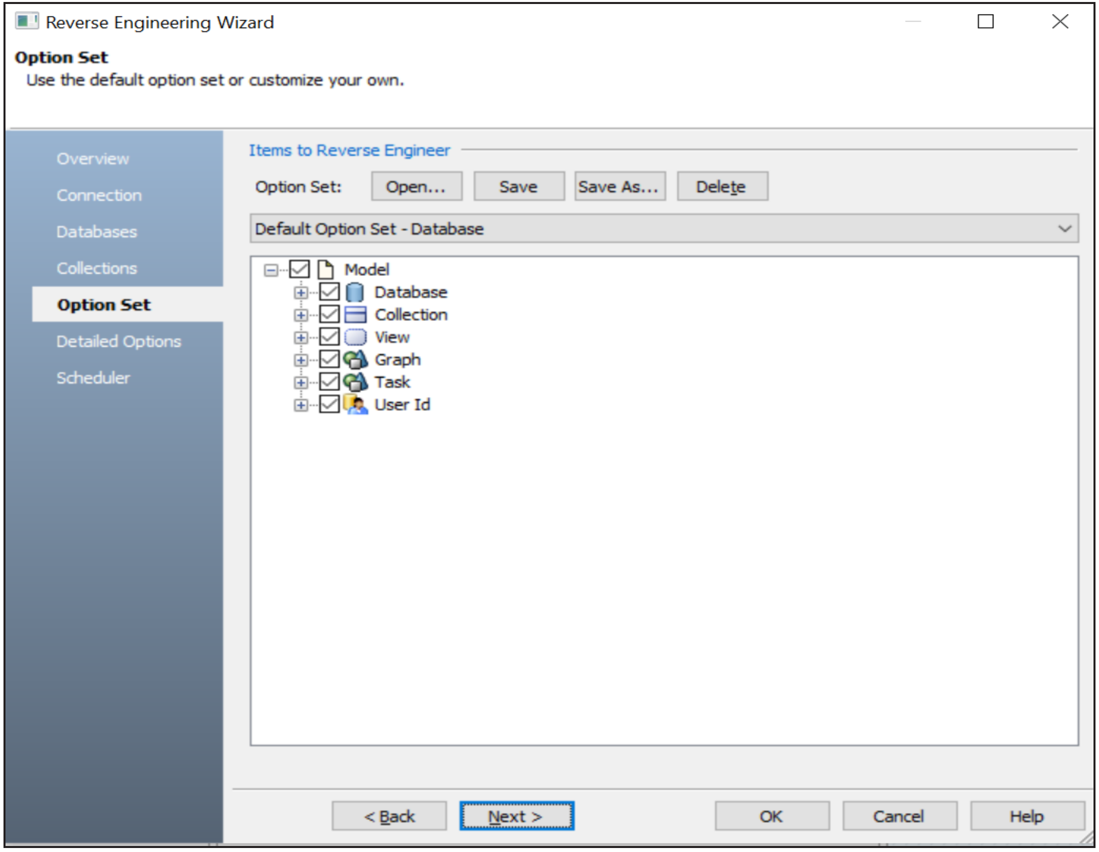
Available Collections: Enter filter text

circles (11)
connections (36)
relation (4)
worldEdges (86)
worldVertices (87)

Selected Collections:

edges (10)
female (2)
knows (5)
male (2)
mps_edges (7)
mps_verts (6)
persons (5)
places (18)

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



Reverse Engineering Wizard

Option Set

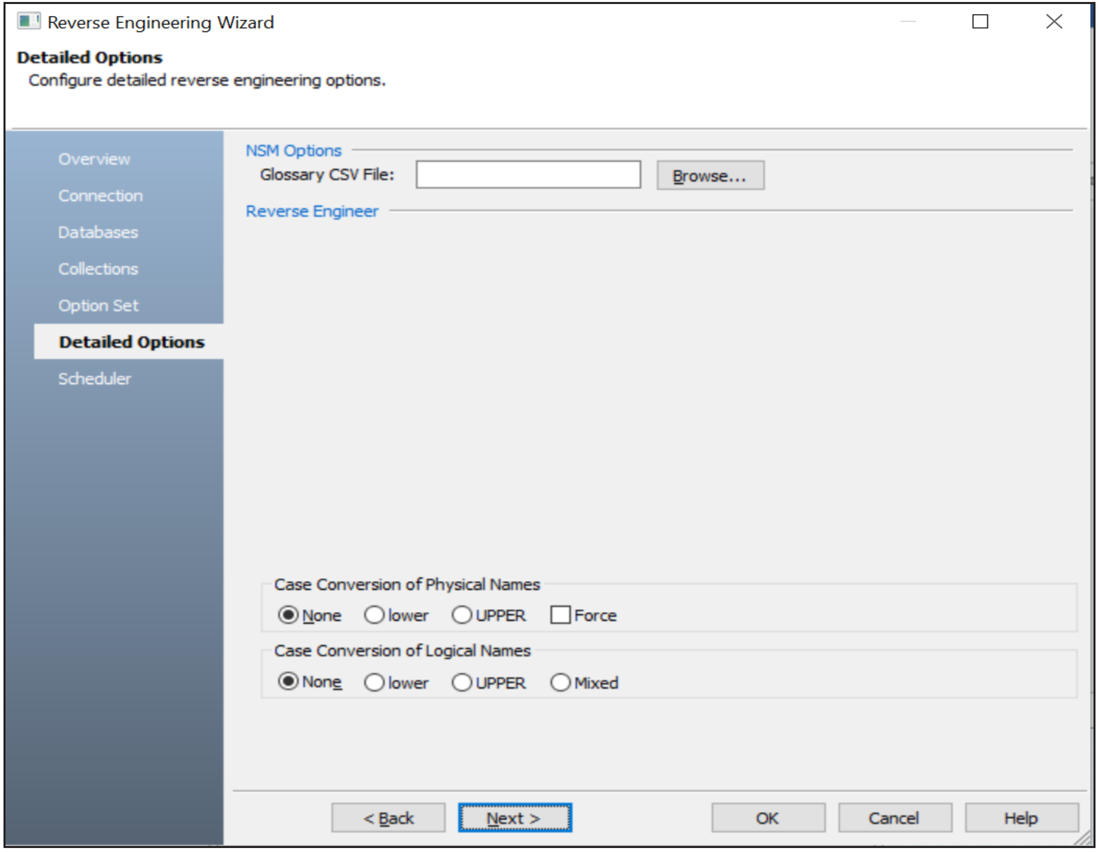
Use the default option set or customize your own.

Default Option Set - Database

- ☑ Model
  - ☑ Database
  - ☑ Collection
  - ☑ View
  - ☑ Graph
  - ☑ Task
  - ☑ User Id

13. Click **Next**.

    The Detail Options section appears. Set up appropriate options based on your requirement.



Detailed Options

Configure detailed reverse engineering options.

NSM Options

Glossary CSV File:

Reverse Engineer

Case Conversion of Physical Names

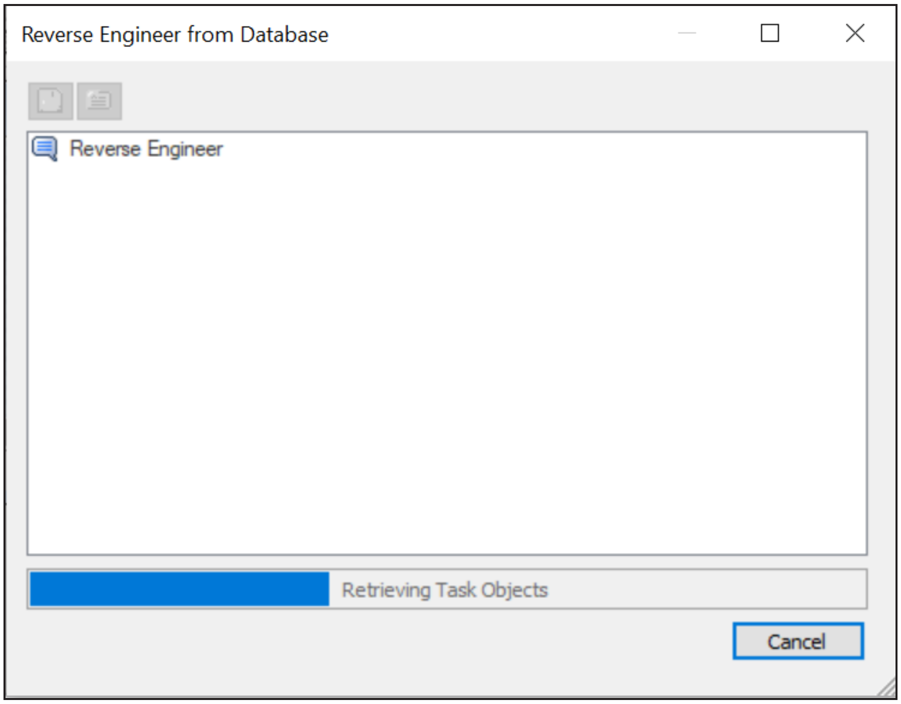
● None ○ lower ○ UPPER □ Force

Case Conversion of Logical Names

● None ○ lower ○ UPPER ○ Mixed

14. Click **OK**.

The reverse engineering process starts.

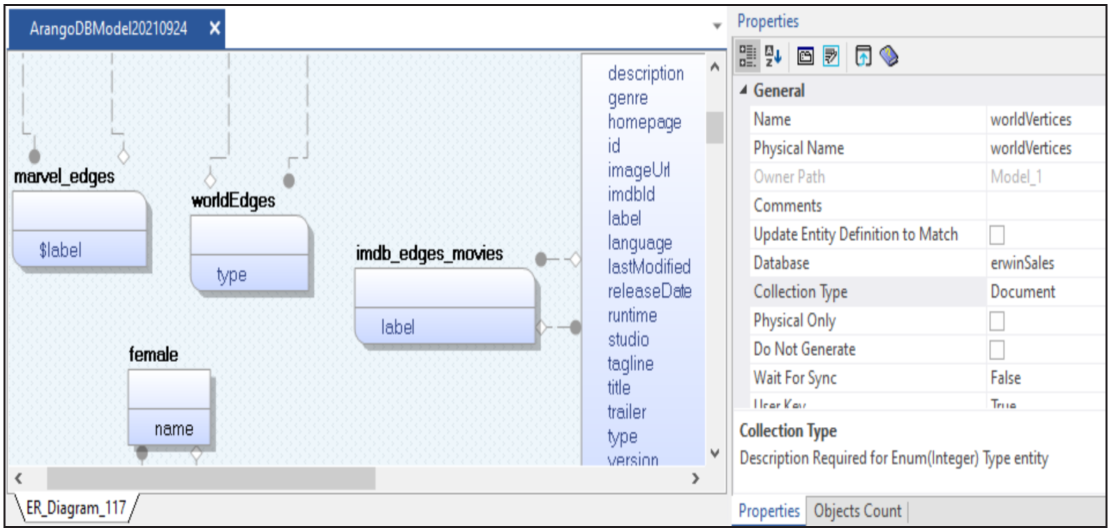


Reverse Engineer from Database

Reverse Engineer

Retrieving Task Objects

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



Properties

General

Name	worldVertices
Physical Name	worldVertices
Owner Path	Model_1
Comments	
Update Entity Definition to Match	□
Database	erwinSales
Collection Type	Document
Physical Only	□
Do Not Generate	□
Wait For Sync	False
User Key	True

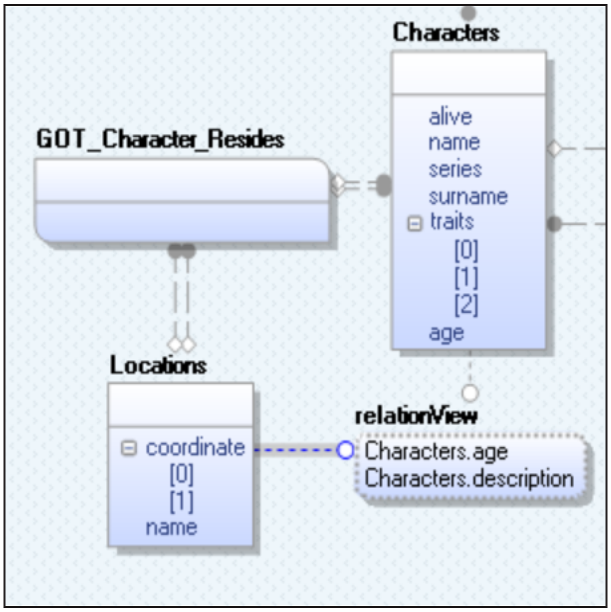
Collection Type

Description Required for Enum(Integer) Type entity

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

The following are the reverse engineering options for ArangoDB 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 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 where the reverse engineered model	When you schedule a job on a remote server, ensure the model path is same for remote and

	should be saved and its name	local server. For example: C:\Scheduler\<Model Name>.erwin
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.

Forward Engineering Models

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

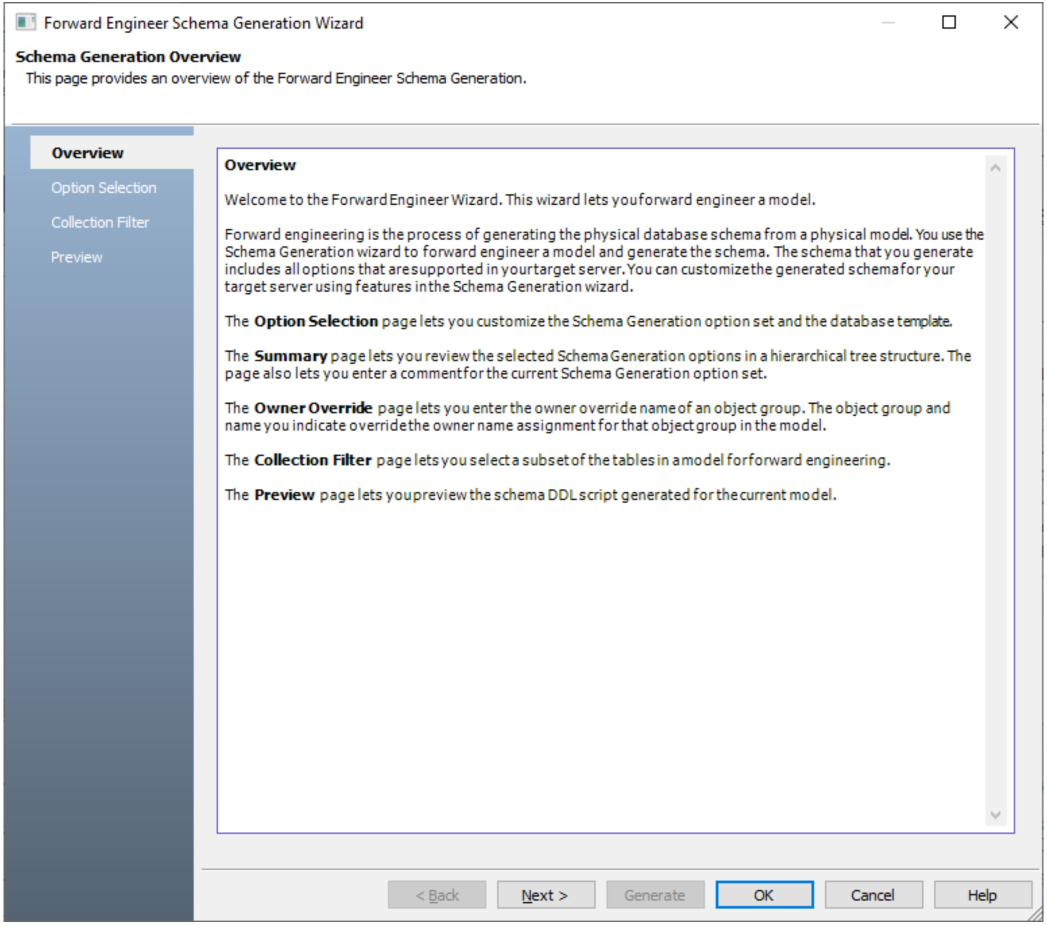
This topic walks you through the steps to forward engineer an ArangoDB model. For detailed description of each forward engineering option refer to the Forward Engineering Options topic.

To forward engineer a model:

1. Open your ArangoDB 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.



Schema Generation Overview

This page provides an overview of the Forward Engineer Schema Generation.

Overview

Welcome to the Forward Engineer Wizard. This wizard lets you forward engineer a model.

Forward engineering is the process of generating the physical database schema from a physical model. You use the Schema Generation wizard to forward engineer a model and generate the schema. The schema that you generate includes all options that are supported in your target server. You can customize the generated schema for your target server using features in the Schema Generation wizard.

The **Option Selection** page lets you customize the Schema Generation option set and the database template.

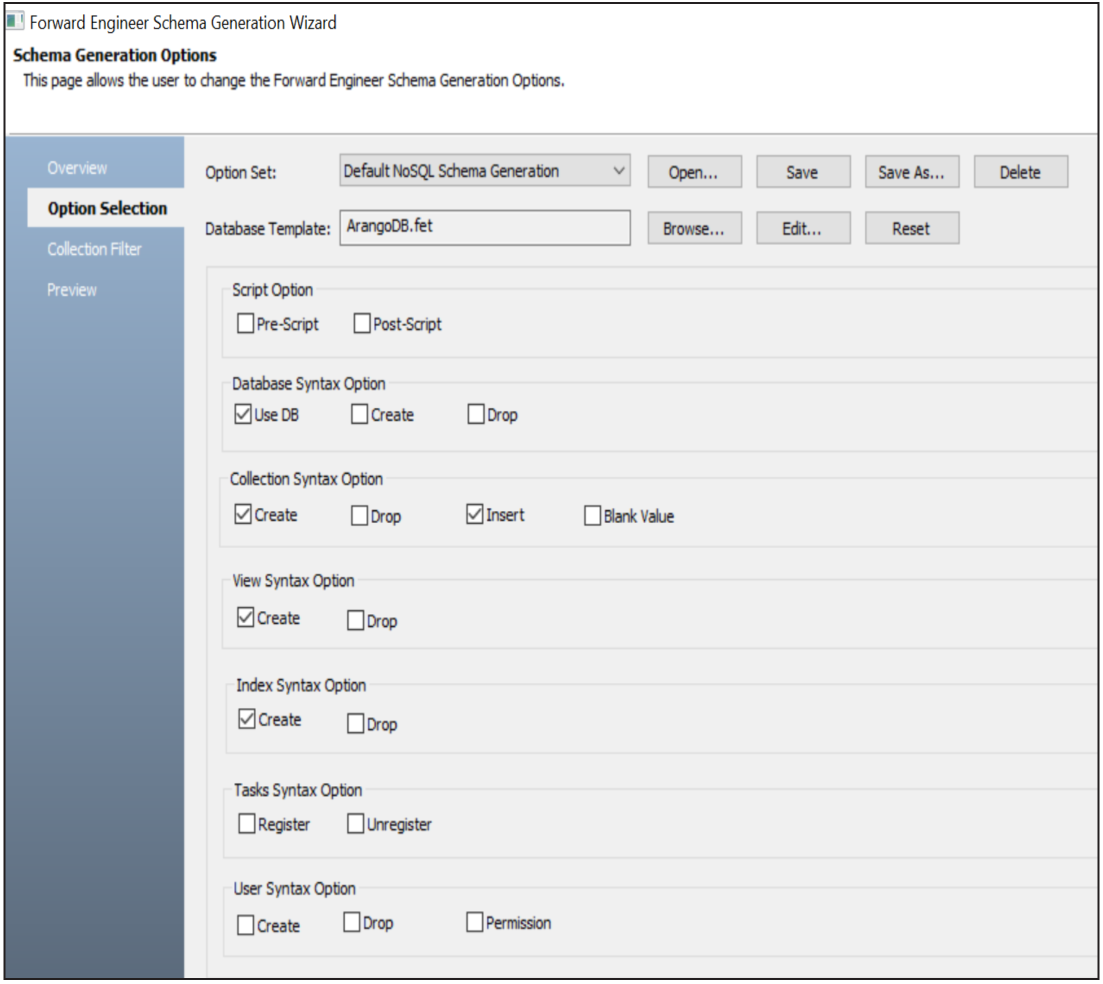
The **Summary** page lets you review the selected Schema Generation options in a hierarchical tree structure. The page also lets you enter a comment for the current Schema Generation option set.

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

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

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

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



Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default NoSQL Schema Generation

Database Template: ArangoDB.fet

Script Option
□ Pre-Script □ Post-Script

Database Syntax Option
☑ Use DB □ Create □ Drop

Collection Syntax Option
☑ Create □ Drop ☑ Insert □ Blank Value

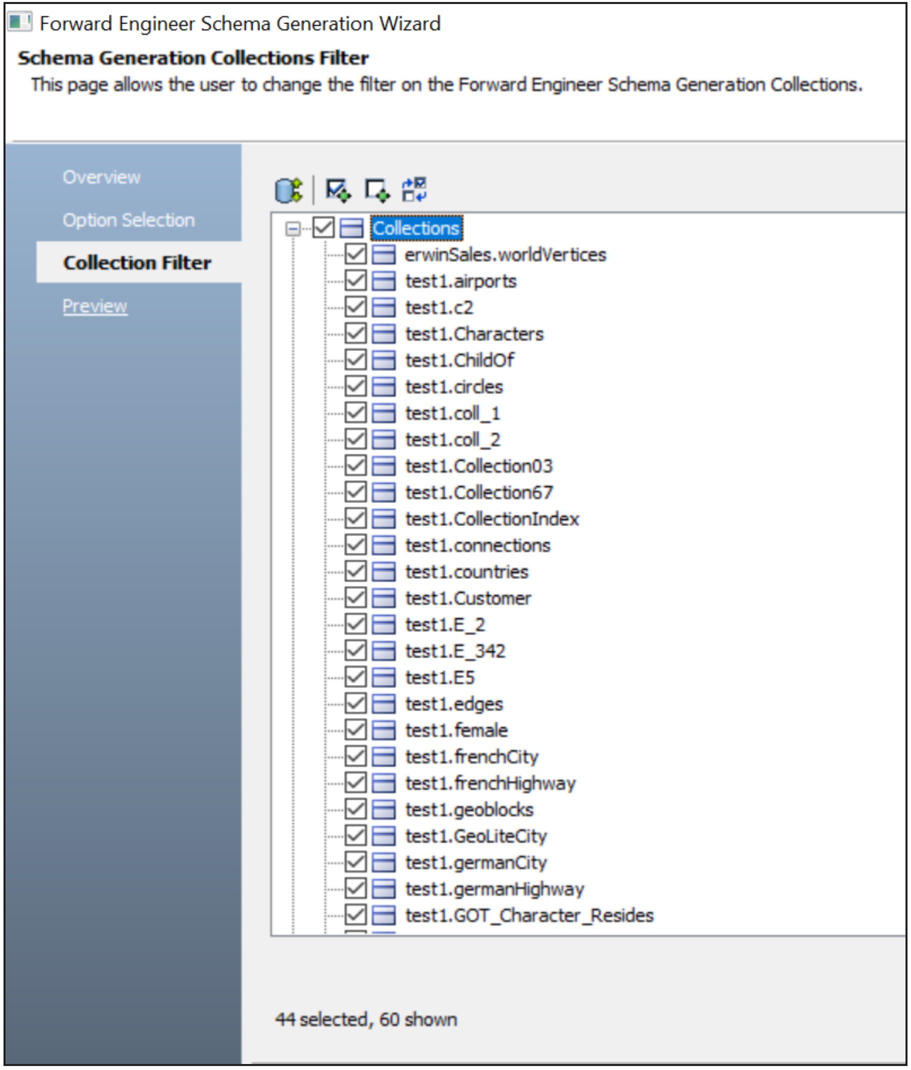
View Syntax Option
☑ Create □ Drop

Index Syntax Option
☑ Create □ Drop

Tasks Syntax Option
□ Register □ Unregister

User Syntax Option
□ Create □ Drop □ Permission

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



Forward Engineer Schema Generation Wizard

Schema Generation Collections Filter

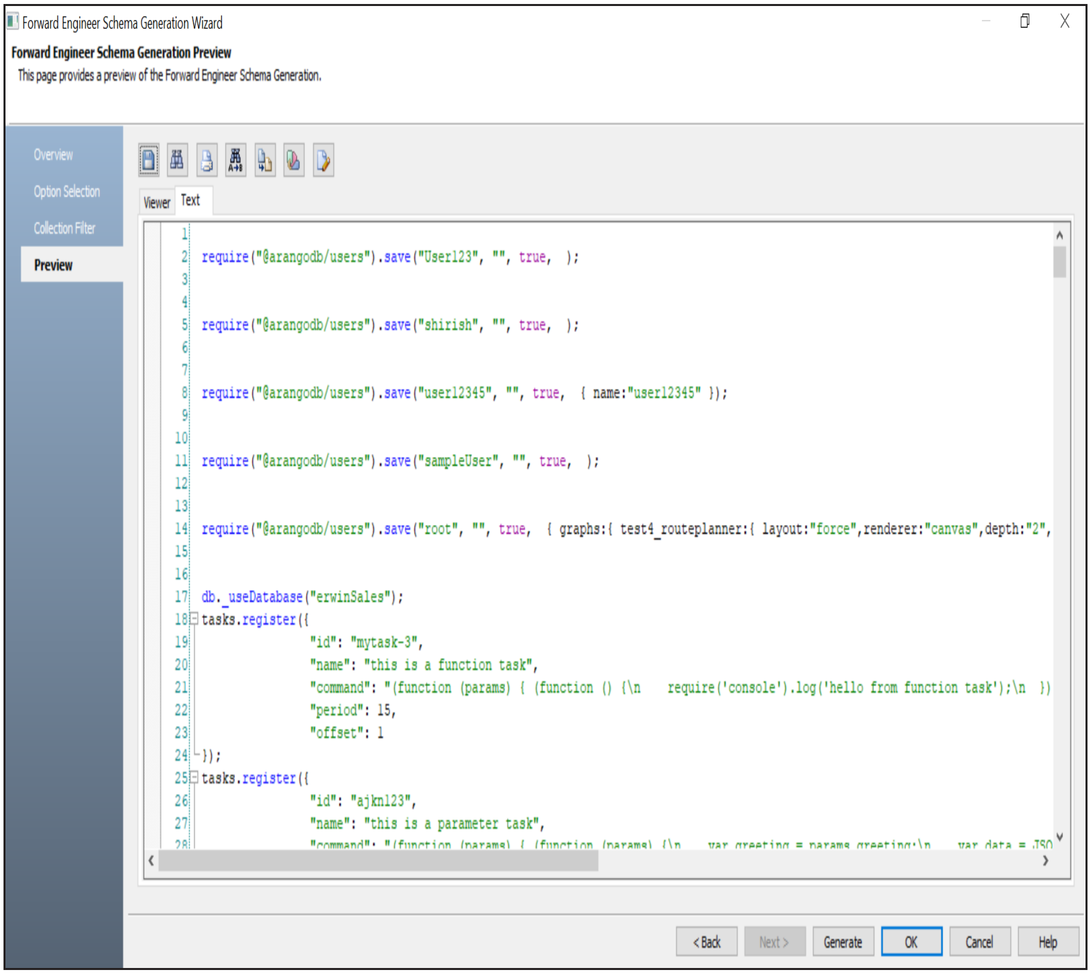
This page allows the user to change the filter on the Forward Engineer Schema Generation Collections.

- ☑ Collections
  - ☑ erwinSales.worldVertices
  - ☑ test1.airports
  - ☑ test1.c2
  - ☑ test1.Characters
  - ☑ test1.ChildOf
  - ☑ test1.circles
  - ☑ test1.coll_1
  - ☑ test1.coll_2
  - ☑ test1.Collection03
  - ☑ test1.Collection67
  - ☑ test1.CollectionIndex
  - ☑ test1.connections
  - ☑ test1.countries
  - ☑ test1.Customer
  - ☑ test1.E_2
  - ☑ test1.E_342
  - ☑ test1.E5
  - ☑ test1.edges
  - ☑ test1.female
  - ☑ test1.frenchCity
  - ☑ test1.frenchHighway
  - ☑ test1.geoblocks
  - ☑ test1.GeoLiteCity
  - ☑ test1.germanCity
  - ☑ test1.germanHighway
  - ☑ test1.GOT_Character_Resides

44 selected, 60 shown

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

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



Forward Engineer Schema Generation Wizard

Forward Engineer Schema Generation Preview

This page provides a preview of the Forward Engineer Schema Generation.

```

require("@arangodb/users").save("User123", "", true,  );


require("@arangodb/users").save("shirish", "", true,  );


require("@arangodb/users").save("user12345", "", true,  { name:"user12345" });


require("@arangodb/users").save("sampleUser", "", true,  );


require("@arangodb/users").save("root", "", true,  { graphs:{ test4_routeplanner:{ layout:"force",renderer:"canvas",depth:"2",


db._useDatabase("erwinSales");
tasks.register({
                "id": "mytask-3",
                "name": "this is a function task",
                "command": "(function (params) { (function () {\n    require('console').log('hello from function task');\n  })
                "period": 15,
                "offset": 1
});
tasks.register({
                "id": "ajkn123",
                "name": "this is a parameter task",
                "command": "(function (params) { (function (params) {\n    var greeting = params.greeting;\n    var data = JSO
```

Use the following options:

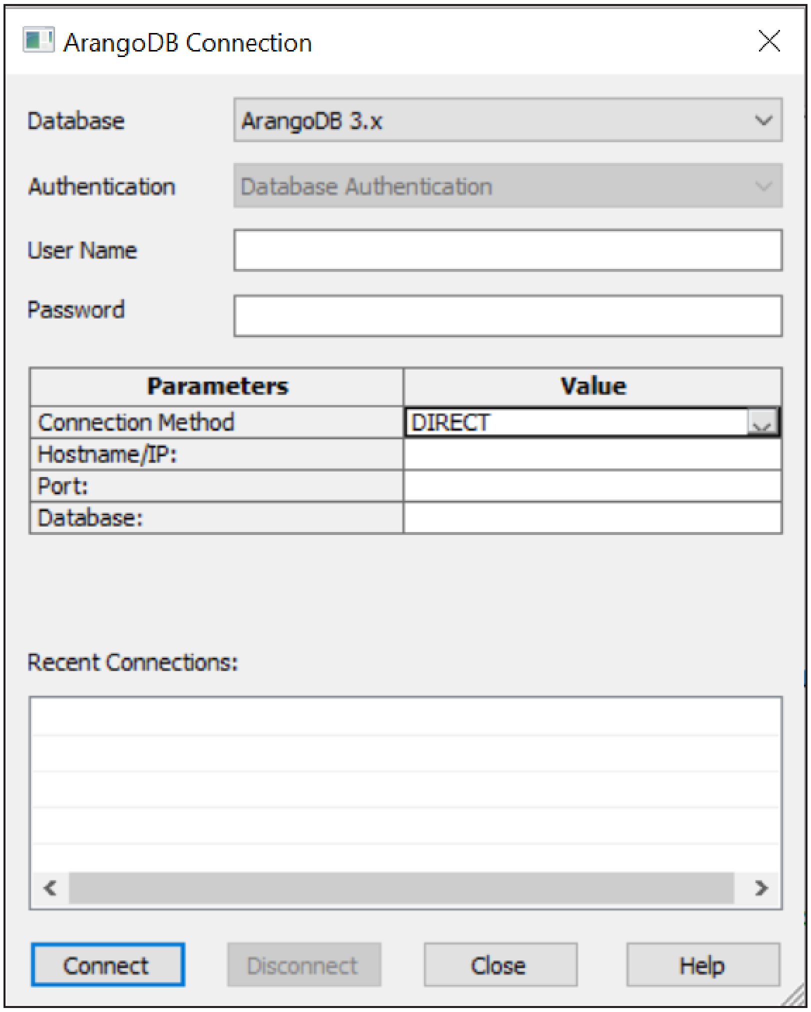
- **Auto Error Check**: Select this option to enable auto error check by the forward engineering wizard.
- **Error Check** (📝): Use this option to run an error check. Based on the results, you can correct the generated script.
- **Text Options** (🎨): Use this option to configure the preview text editor's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



ArangoDB Connection

Database: ArangoDB 3.x

Authentication: Database Authentication

User Name:

Password:

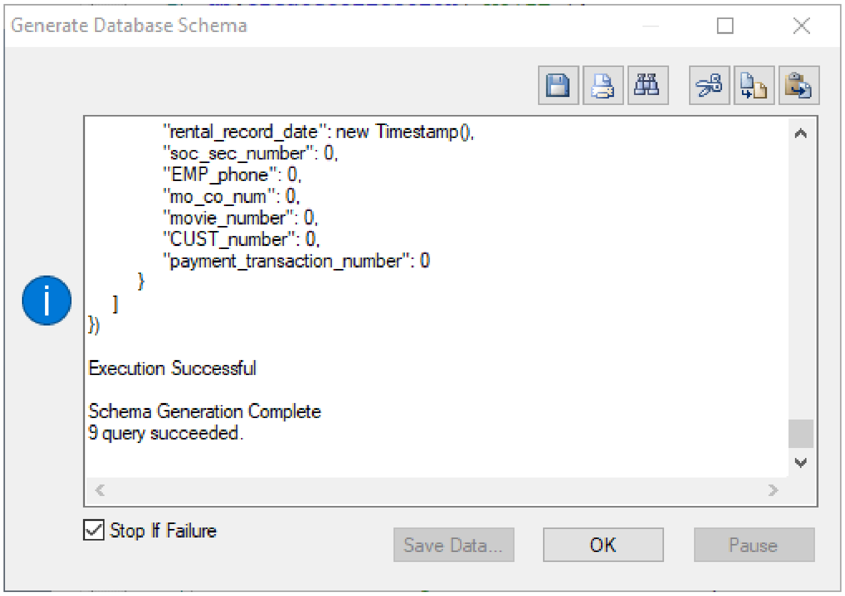
Parameters	Value
Connection Method	DIRECT
Hostname/IP:	
Port:	
Database:	

Recent Connections:

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

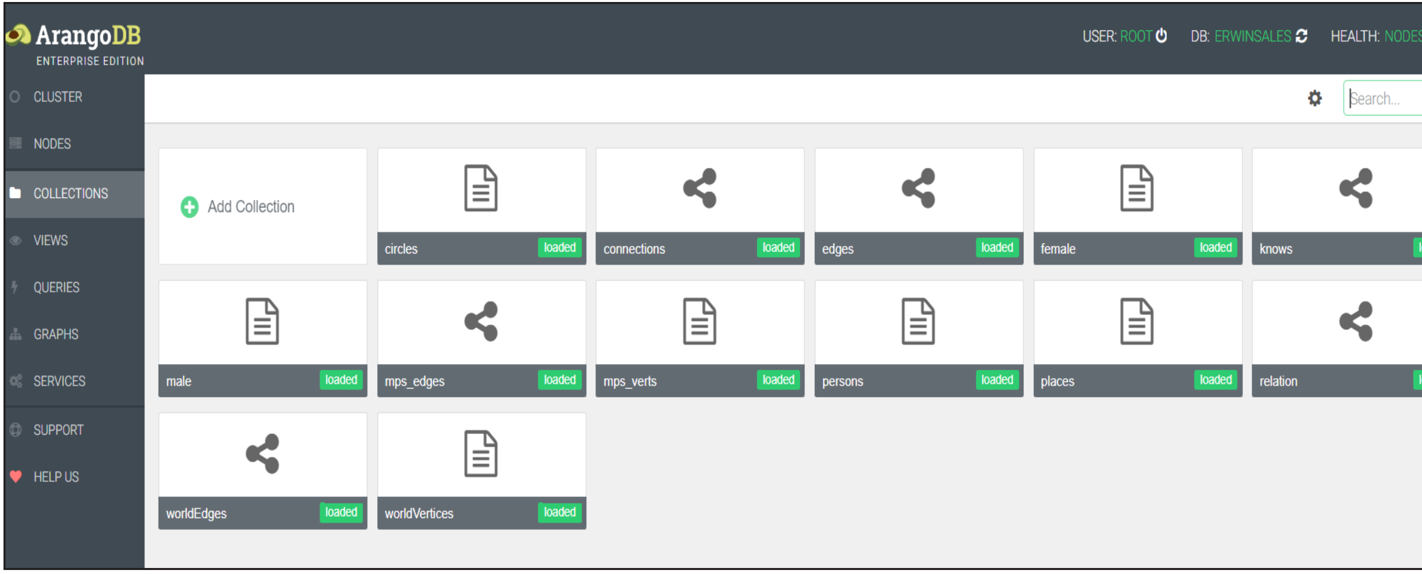
The objects move to a database entered on the ArangoDB Connection page irrespective of the databases entered on the object editor pages. If you want to move objects to databases as entered on object editors page then do not enter any database on the ArangoDB Connection page.

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

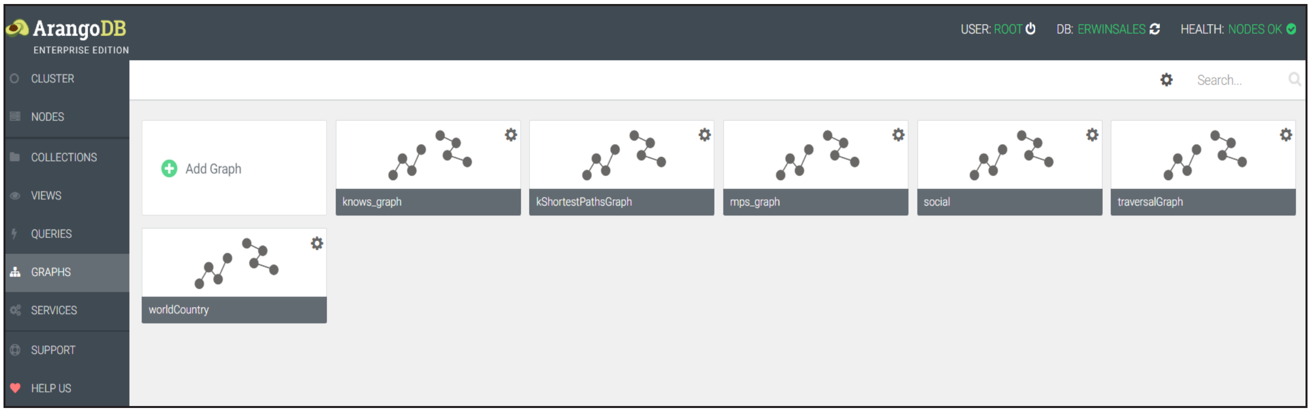


```
            "rental_record_date": new Timestamp(),
            "soc_sec_number": 0,
            "EMP_phone": 0,
            "mo_co_num": 0,
            "movie_number": 0,
            "CUST_number": 0,
            "payment_transaction_number": 0
        }
    ]
})

Execution Successful

Schema Generation Complete
9 query succeeded.
```

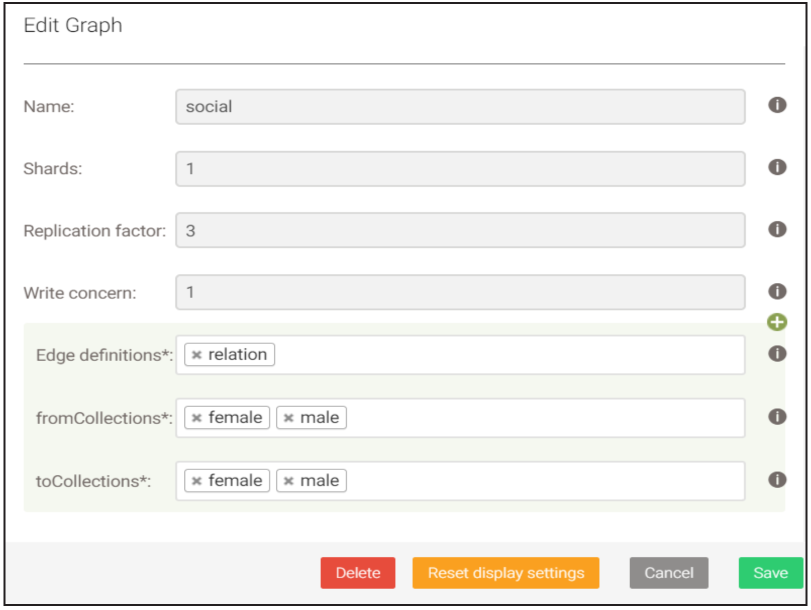
The forward engineering process creates database objects in the database entered on the ArangoDB Connection page. For example, in the following image, the forward engineering process creates 13 collections in the ERWINSALES database. ArangoDB represents edge collections using ⋘ whereas document collections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

Forward Engineering Options for ArangoDB

The following are the forward engineering options for ArangoDB in erwin DM.

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.

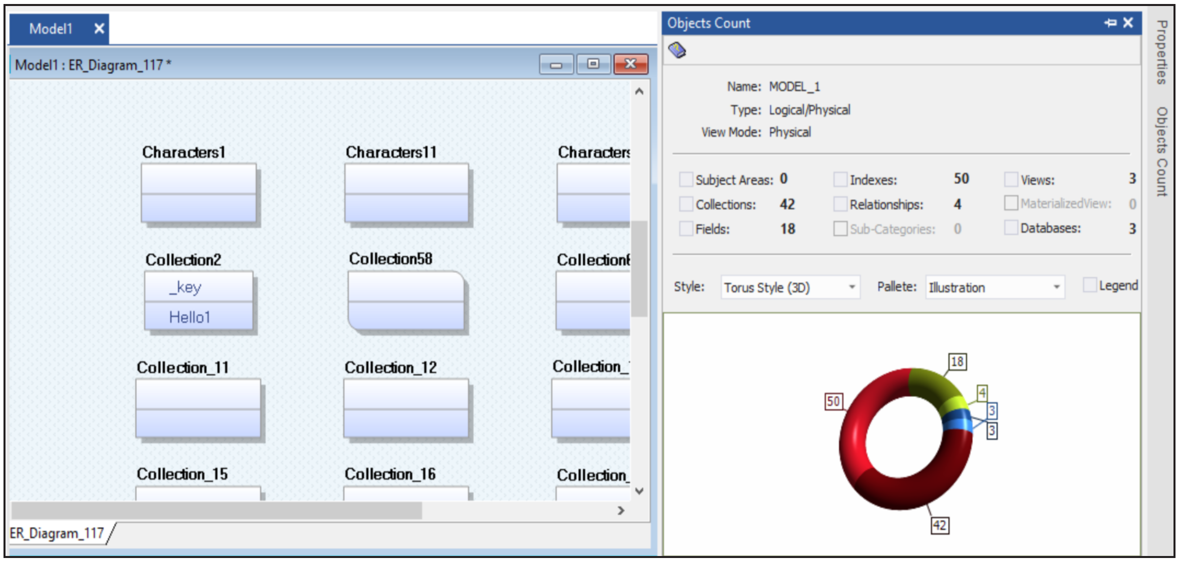
This topic walks you through the steps to compare an ArangoDB model with database.

To compare models with database:

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

   > Ensure that you are in the Physical mode.

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

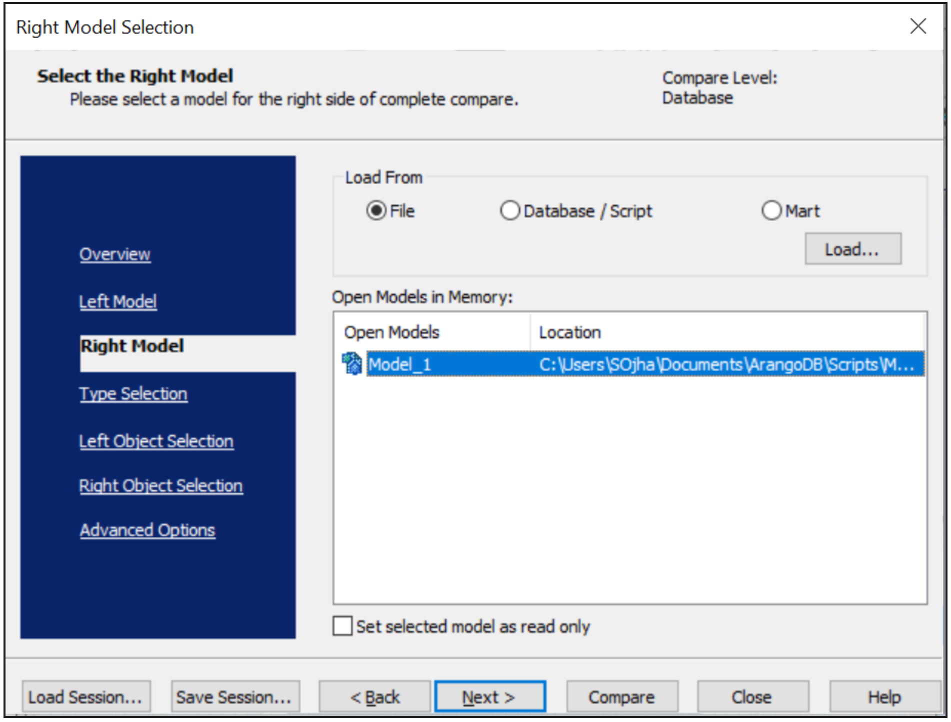


Name: MODEL_1
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	0	□ Indexes:	50	□ Views:	3
□ Collections:	42	□ Relationships:	4	□ MaterializedView:	0
□ Fields:	18	□ Sub-Categories:	0	□ Databases:	3

Style: Torus Style (3D) Pallete: Illustration □ Legend

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.



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

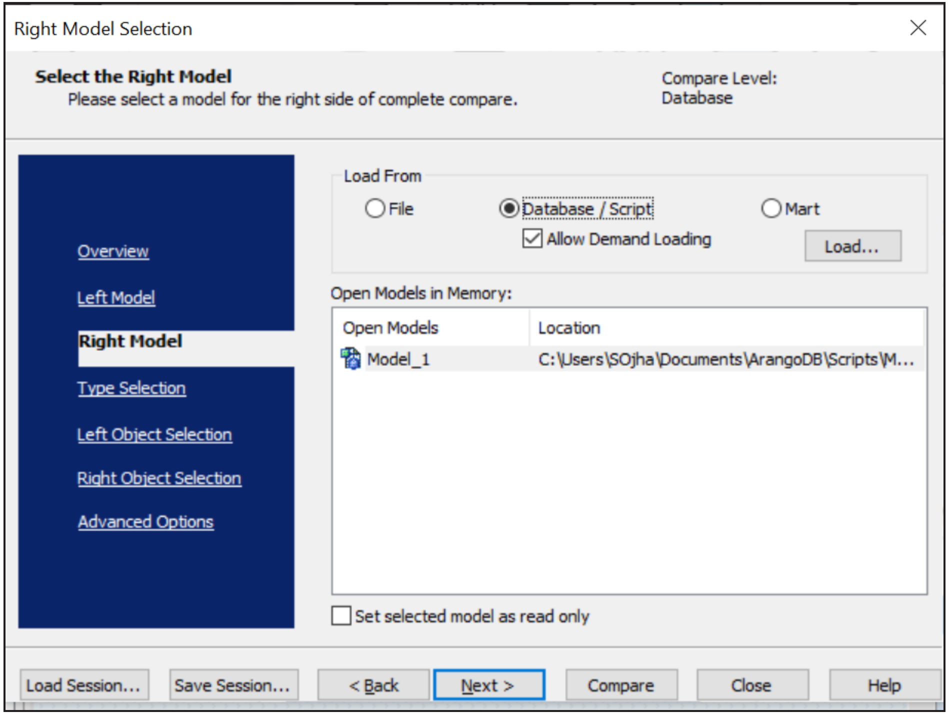
● File ○ Database / Script ○ Mart

Open Models in Memory:

Open Models	Location
Model_1	C:\Users\SOjha\Documents\ArangoDB\Scripts\M...

□ Set selected model as read only

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



Right Model Selection

Select the Right Model
Please select a model for the right side of complete compare.

Compare Level: Database

Load From: ○ File ● Database / Script ○ Mart

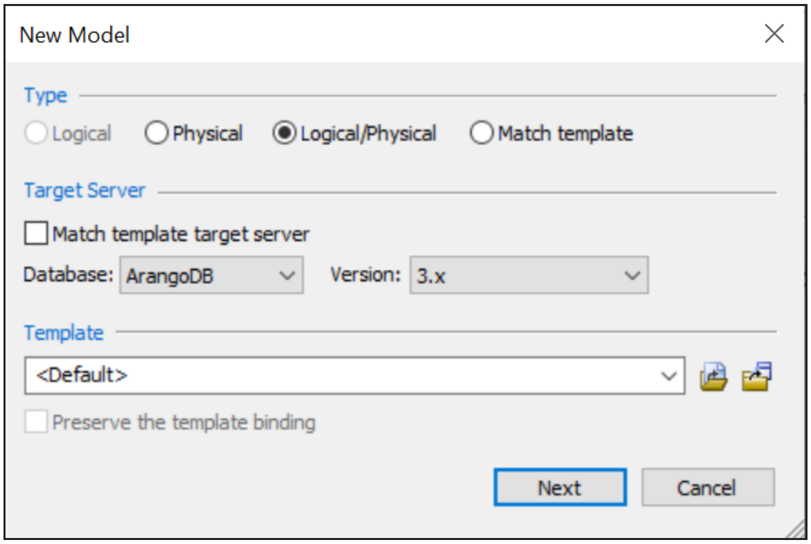
☑ Allow Demand Loading

Open Models in Memory:

Open Models	Location
Model_1	C:\Users\SOjha\Documents\ArangoDB\Scripts\M...

□ Set selected model as read only

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



New Model

Type: ○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

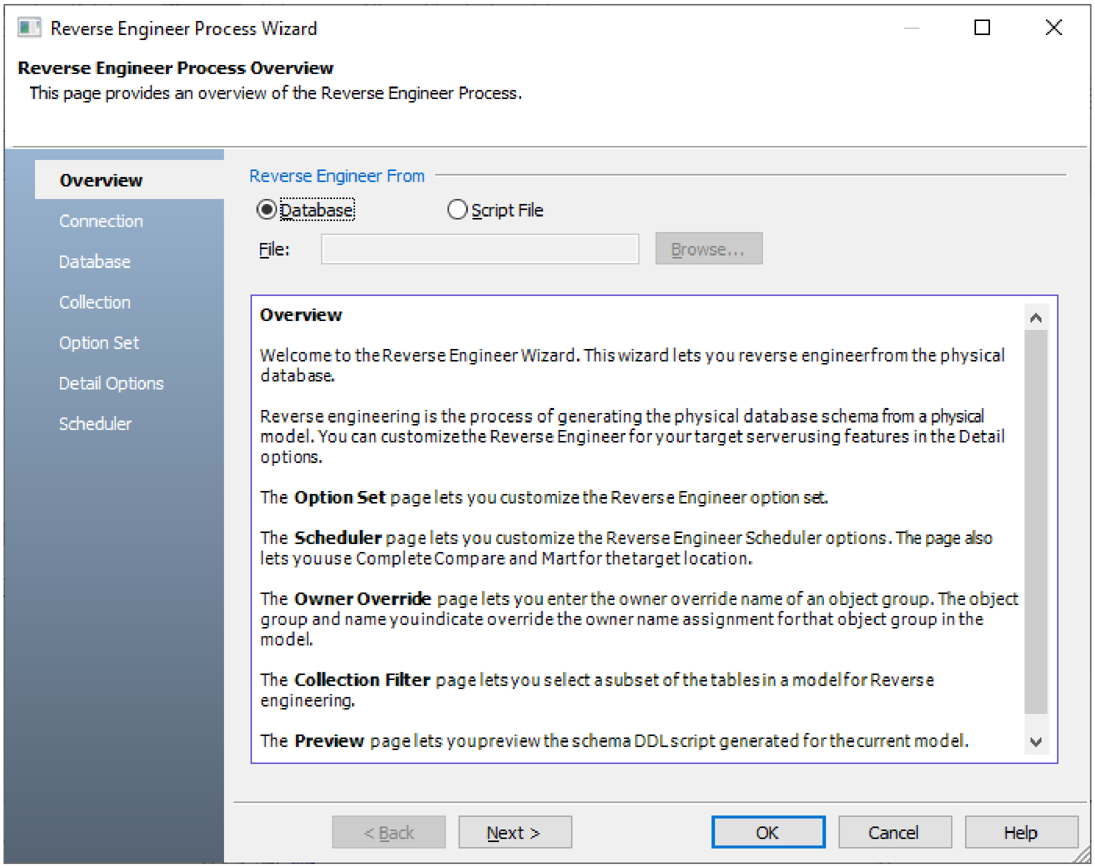
Database: ArangoDB Version: 3.x

Template: <Default>

□ Preserve the template binding

5. Ensure that the Database is set to the correct one. In this case, ArangoDB. Then, click **Next**.

   The Reverse Engineer Process Wizard appears.



Reverse Engineer Process Overview

This page provides an overview of the Reverse Engineer Process.

● Database ○ Script File

Overview

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

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

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

The **Scheduler** page lets you customize the Reverse Engineer Scheduler options. The page also lets youuse Complete Compare and Martfor thetarget location.

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

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

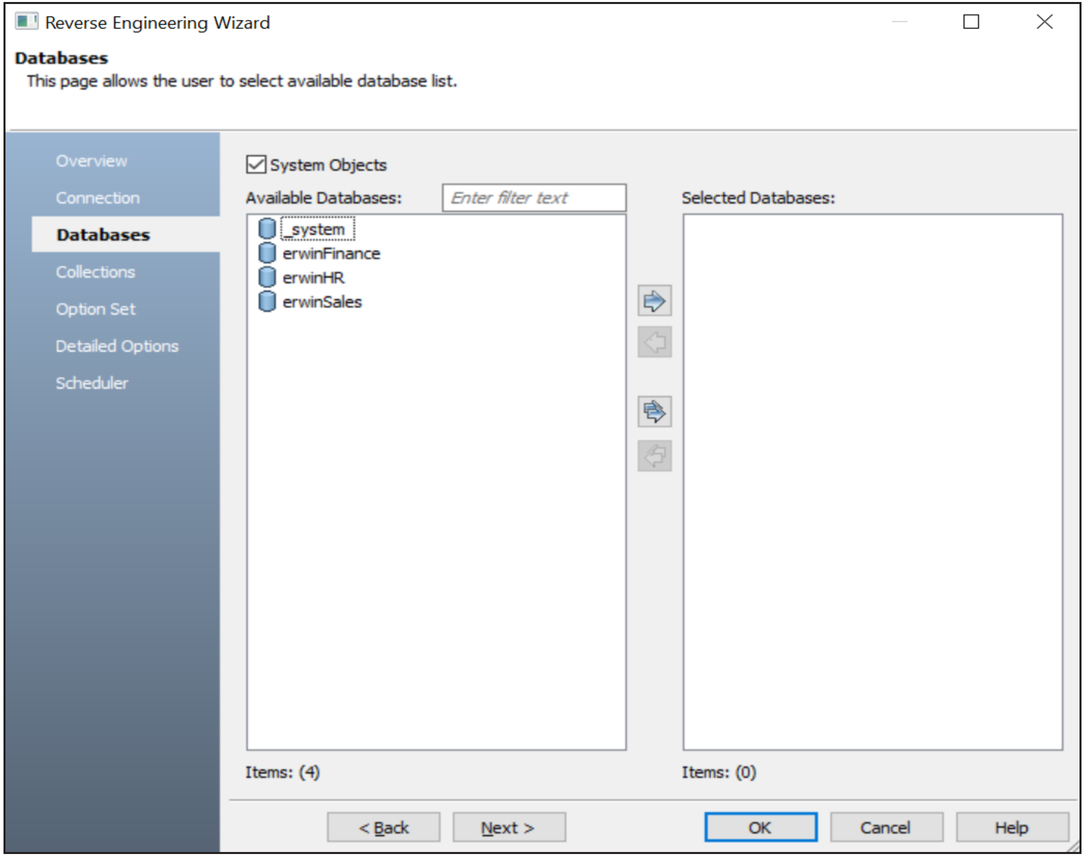
The **Preview** page lets youpreview the schema DDL script generated for the current model.

6. Click **Database**. Then, click **Next**.

   The Connection section appears. Use this section to connect to the database from which you want to reverse engineer the model.

7. After connection is established, click **Next**.

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



Reverse Engineering Wizard

Databases

This page allows the user to select available database list.

☑ System Objects

Available Databases: Enter filter text

- _system
- erwinFinance
- erwinHR
- erwinSales

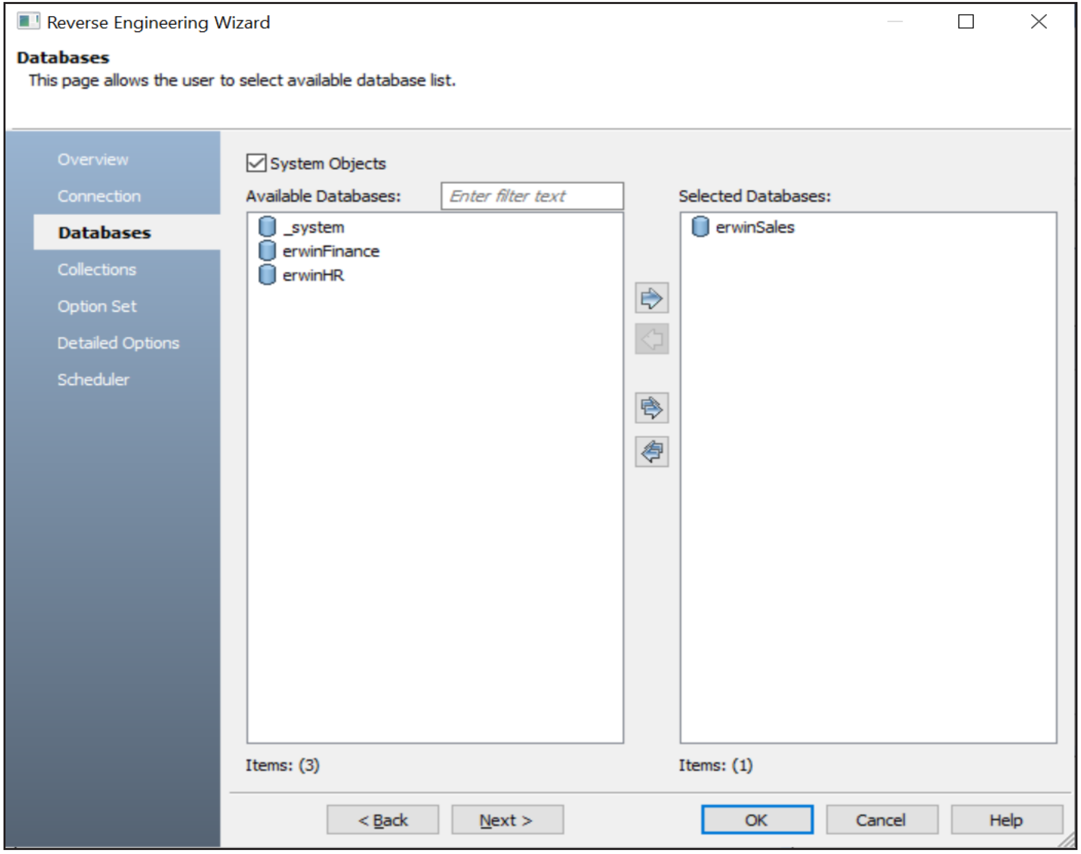
Selected Databases:

Items: (4)

Items: (0)

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

   This moves the selected databases under Selected Databases.



Databases

This page allows the user to select available database list.

☑ System Objects

Available Databases: Enter filter text

- _system
- erwinFinance
- erwinHR

Items: (3)

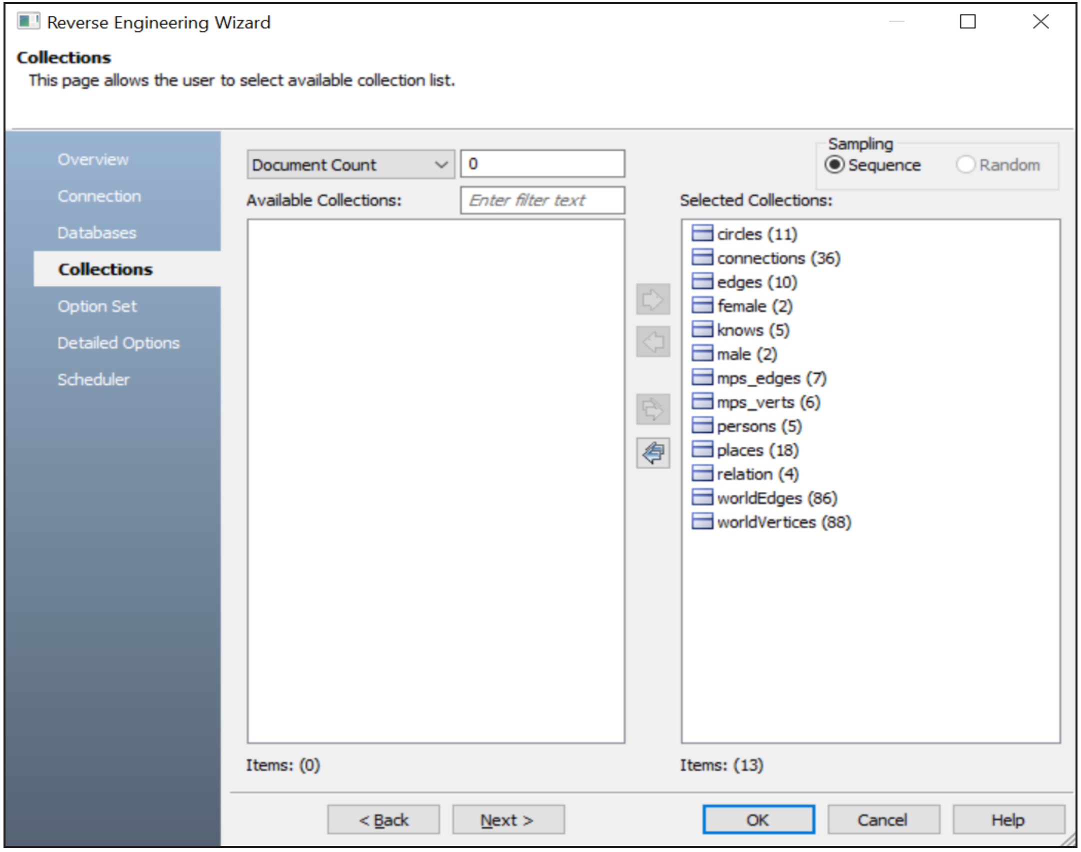
Selected Databases:

- erwinSales

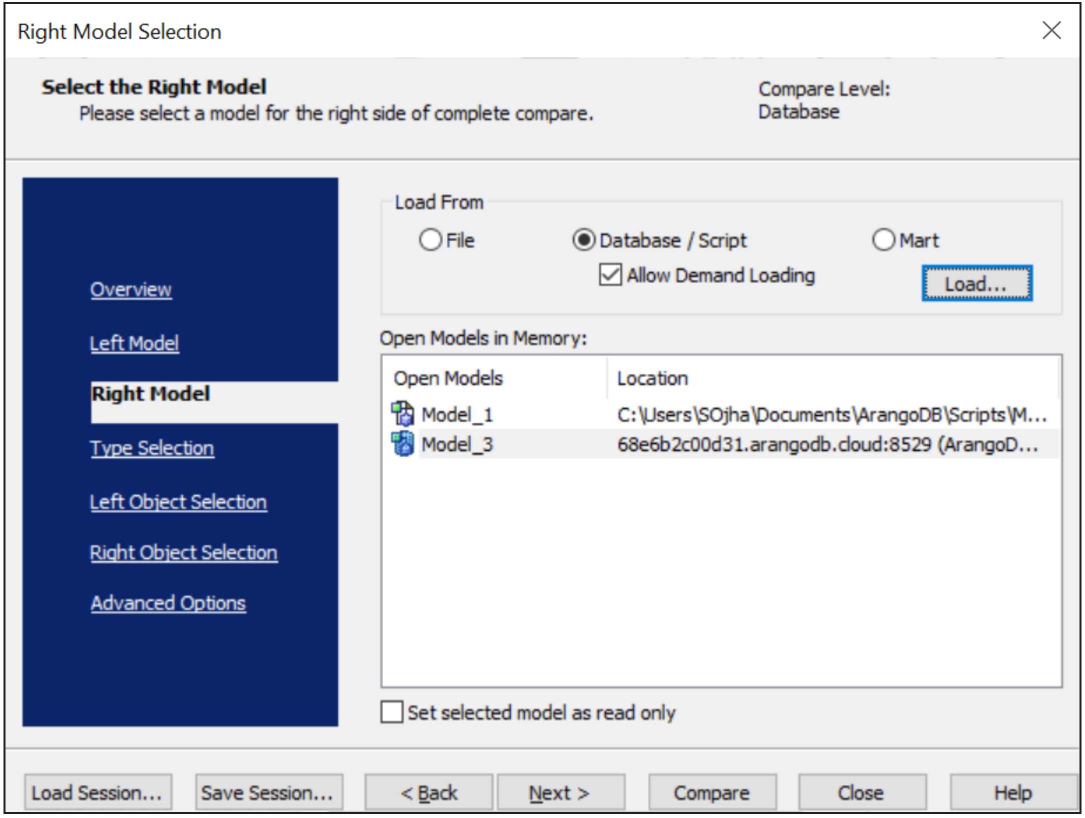
Items: (1)

9. Click **Next** and in the Collection section, click ⇛.

   This selects all the available collections. Also, ensure that the Document Count/Document % is not set to zero (0).



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



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level: Database

Load From

○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

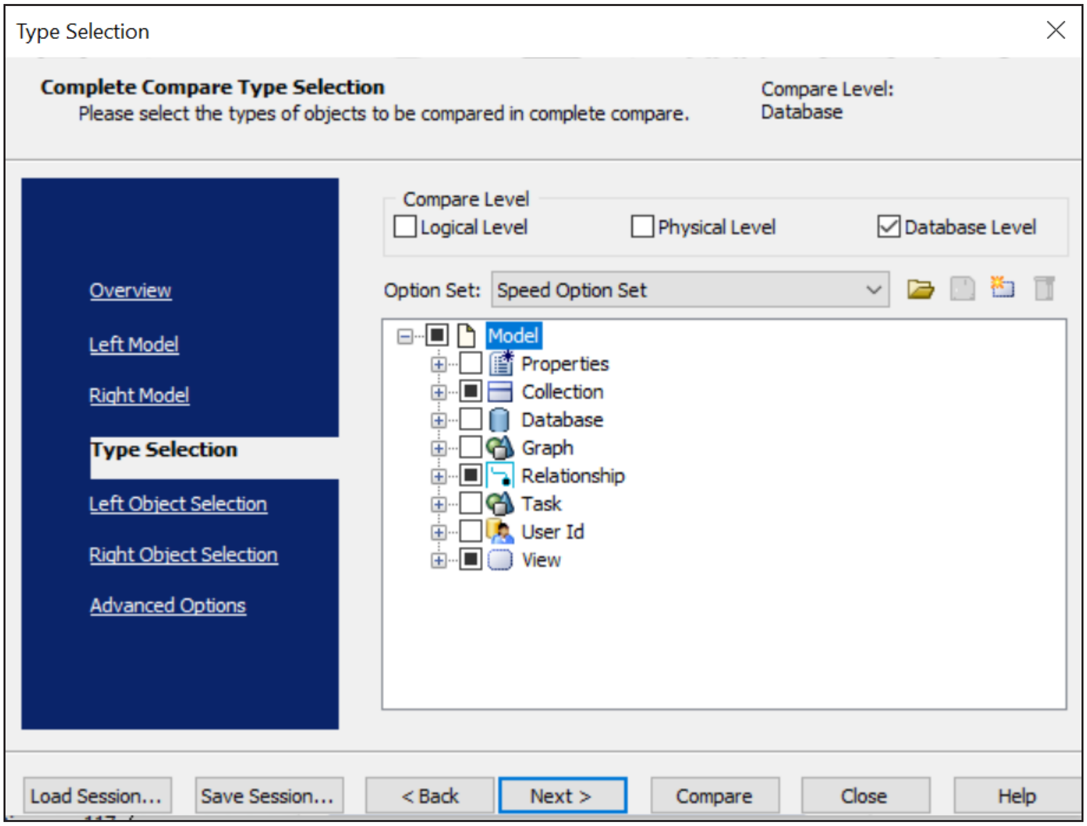
Open Models in Memory:

Open Models	Location
Model_1	C:\Users\SOjha\Documents\ArangoDB\Scripts\M...
Model_3	68e6b2c00d31.arangodb.cloud:8529 (ArangoD...

□ Set selected model as read only

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

For example, the following image shows the default options.



Complete Compare Type Selection

Please select the types of objects to be compared in complete compare.

Compare Level:
Database

Compare Level

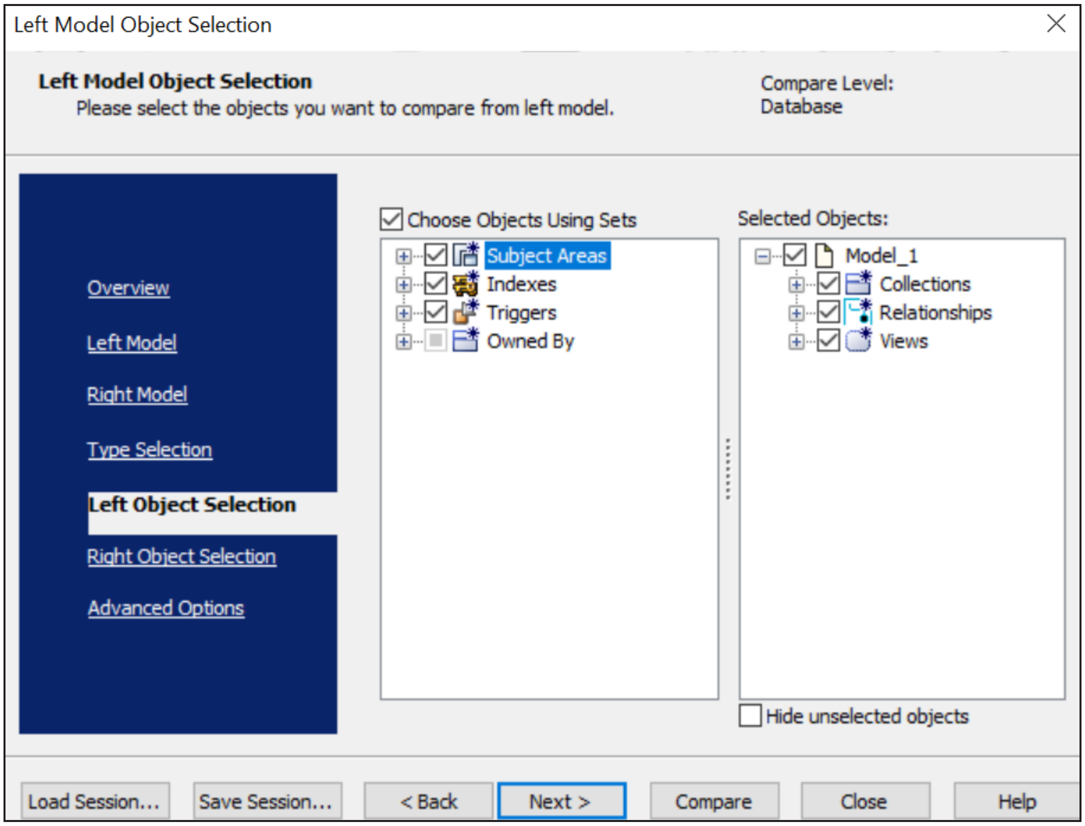
□ Logical Level □ Physical Level ☑ Database Level

Option Set: Speed Option Set

- ■ Model
  - □ Properties
  - ■ Collection
  - □ Database
  - □ Graph
  - ■ Relationship
  - □ Task
  - □ User Id
  - ■ View

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

For example, the following image shows the default options.



Left Model Object Selection

Left Model Object Selection
Please select the objects you want to compare from left model.

Compare Level:
Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

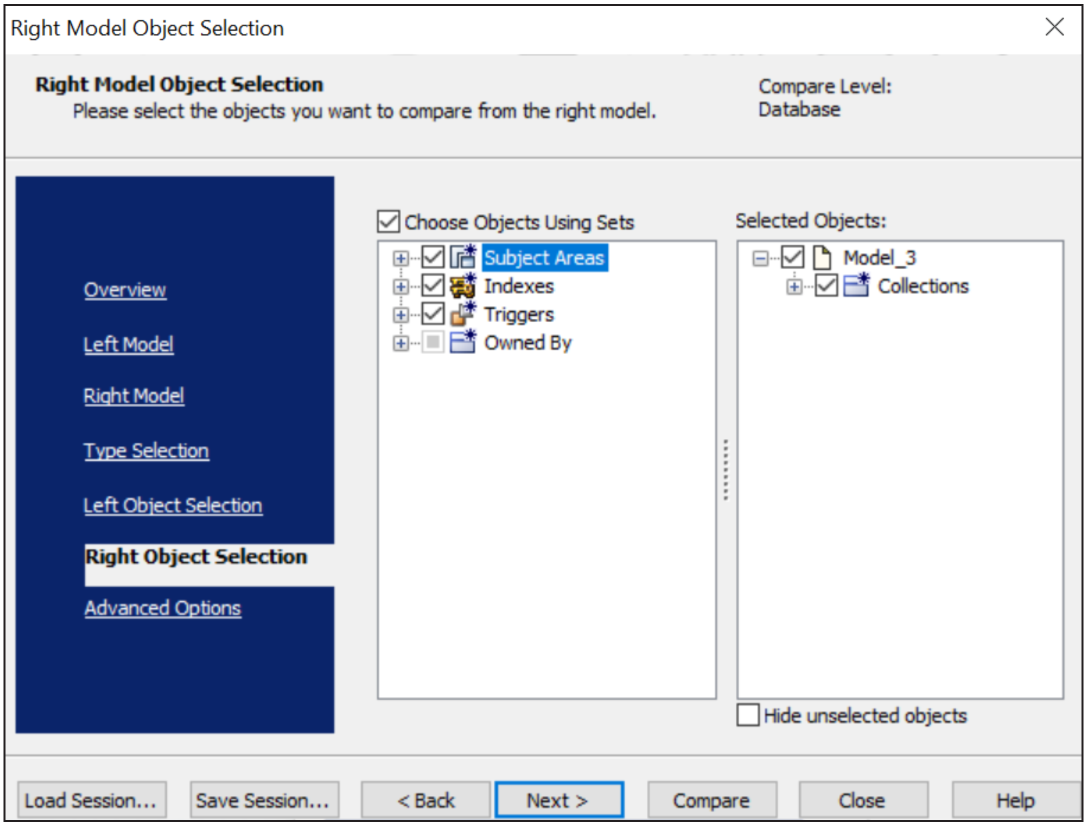
Selected Objects:

- ☑ Model_1
  - ☑ Collections
  - ☑ Relationships
  - ☑ Views

□ Hide unselected objects

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

For example, the following image shows the default options.



Right Model Object Selection

Right Model Object Selection
Please select the objects you want to compare from the right model.

Compare Level:
Database

☑ Choose Objects Using Sets

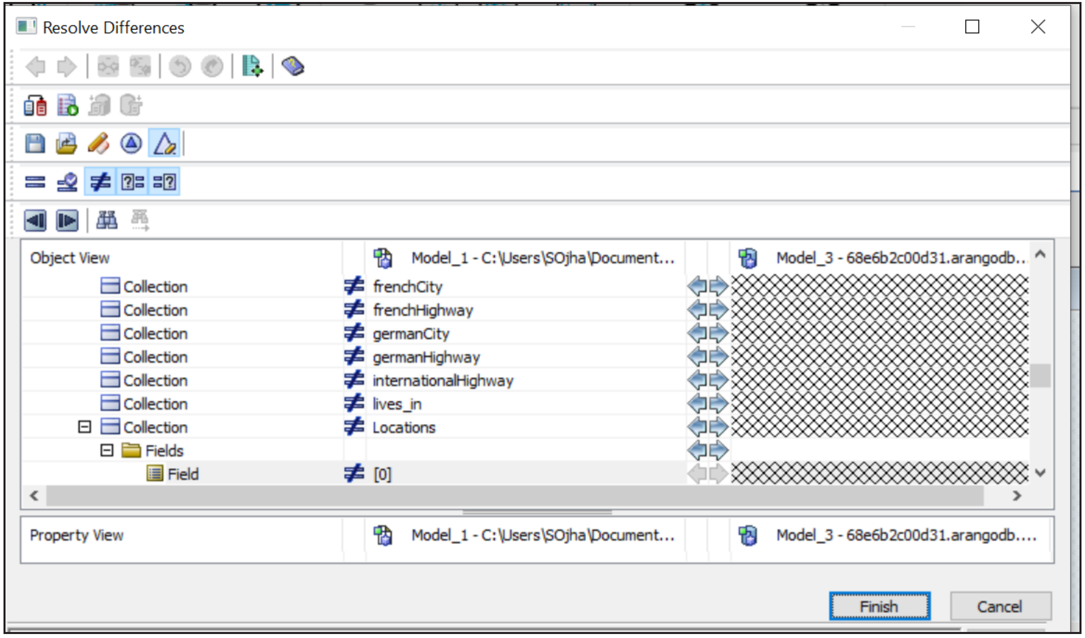
Selected Objects:

☐ Hide unselected objects

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.

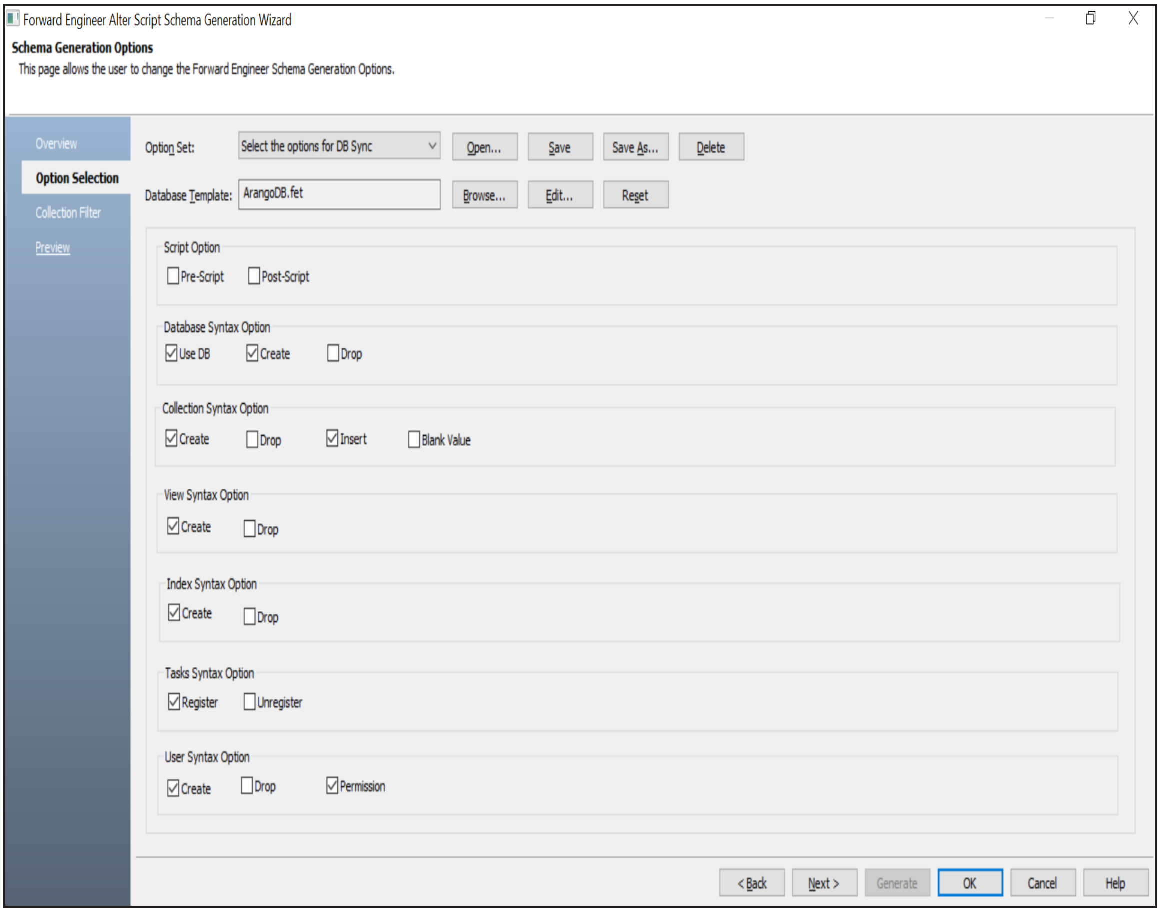

Resolve Differences
Object View
Model_1 - C:\Users\SOjha\Document...
Model_3 - 68e6b2c00d31.arangodb...
Collection
Collection
Collection
Collection
Collection
Collection
Collection
Fields
Field
Property View
Model_1 - C:\Users\SOjha\Document...
Model_3 - 68e6b2c00d31.arangodb....
Finish
Cancel

Object View	Model_1 - C:\Users\SOjha\Document...	Model_3 - 68e6b2c00d31.arangodb...
Collection	≠ frenchCity	▨
Collection	≠ frenchHighway	▨
Collection	≠ germanCity	▨
Collection	≠ germanHighway	▨
Collection	≠ internationalHighway	▨
Collection	≠ lives_in	▨
Collection	≠ Locations	▨
Fields		
Field	≠ [0]	▨

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.



Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Select the options for DB Sync

Database Template: ArangoDB.fet

Script Option

□ Pre-Script □ Post-Script

Database Syntax Option

☑ Use DB ☑ Create □ Drop

Collection Syntax Option

☑ Create □ Drop ☑ Insert □ Blank Value

View Syntax Option

☑ Create □ Drop

Index Syntax Option

☑ Create □ Drop

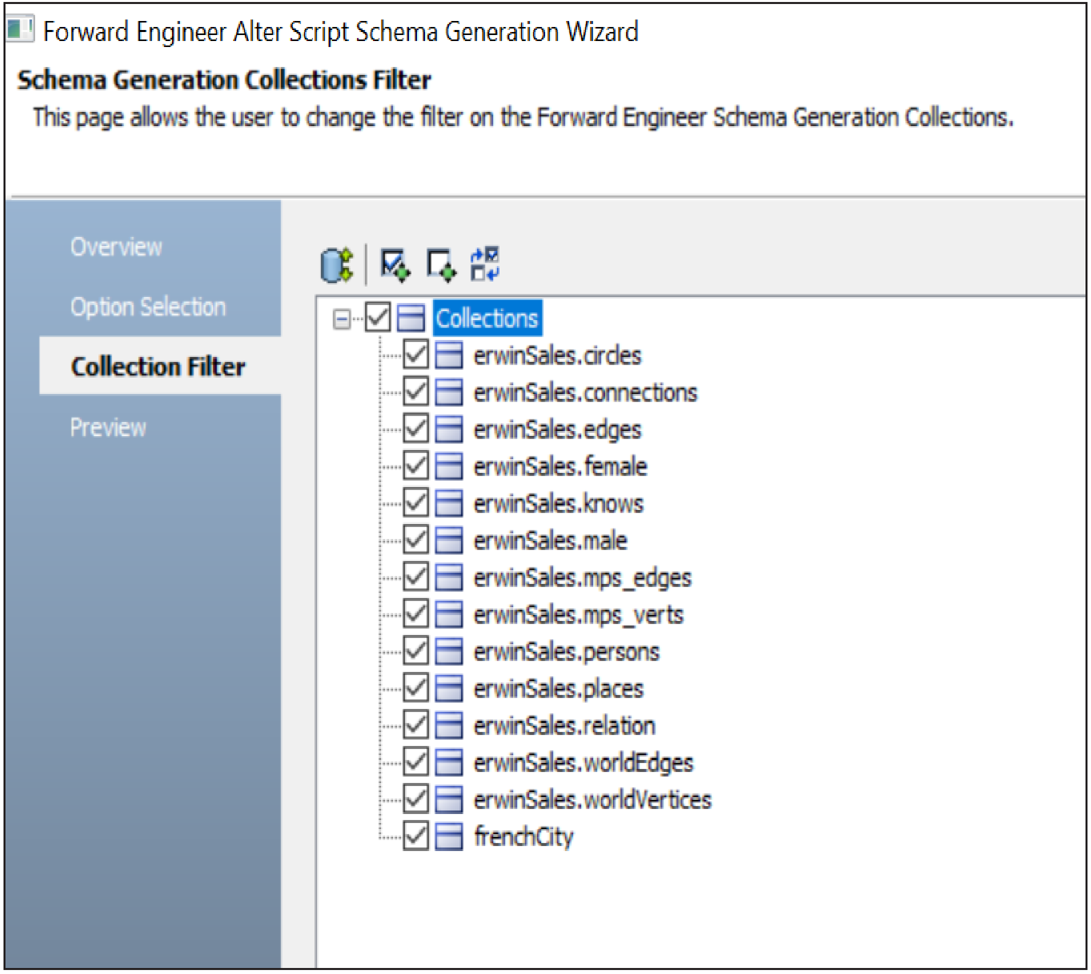
Tasks Syntax Option

☑ Register □ Unregister

User Syntax Option

☑ Create □ Drop ☑ Permission

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



Forward Engineer Alter Script Schema Generation Wizard

Schema Generation Collections Filter

This page allows the user to change the filter on the Forward Engineer Schema Generation Collections.

- ☑ Collections
  - ☑ erwinSales.circles
  - ☑ erwinSales.connections
  - ☑ erwinSales.edges
  - ☑ erwinSales.female
  - ☑ erwinSales.knows
  - ☑ erwinSales.male
  - ☑ erwinSales.mps_edges
  - ☑ erwinSales.mps_verts
  - ☑ erwinSales.persons
  - ☑ erwinSales.places
  - ☑ erwinSales.relation
  - ☑ erwinSales.worldEdges
  - ☑ erwinSales.worldVertices
  - ☑ frenchCity

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.

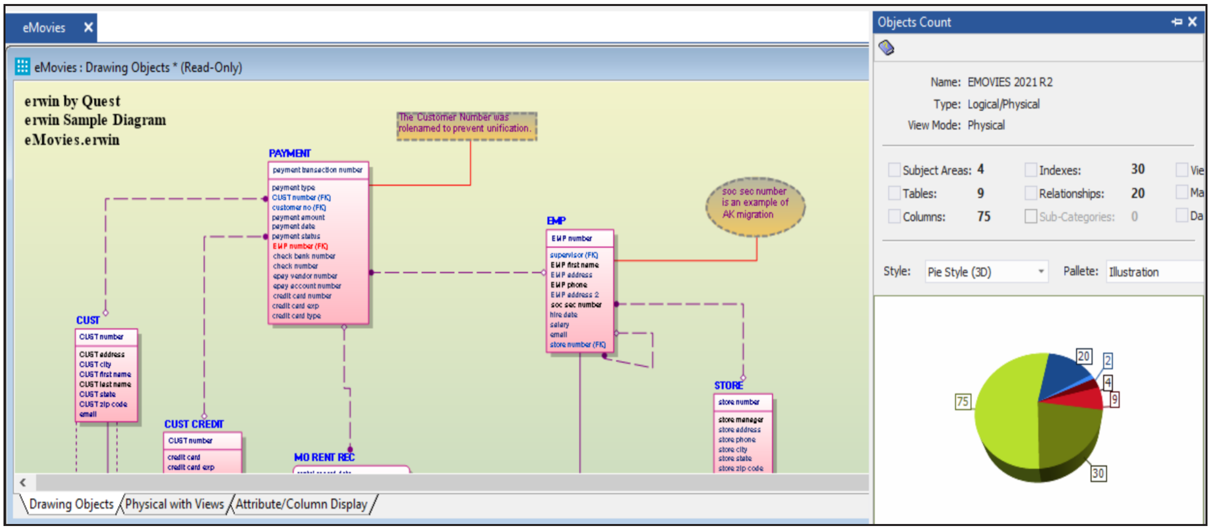
Migration by Changing the Target Database

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.



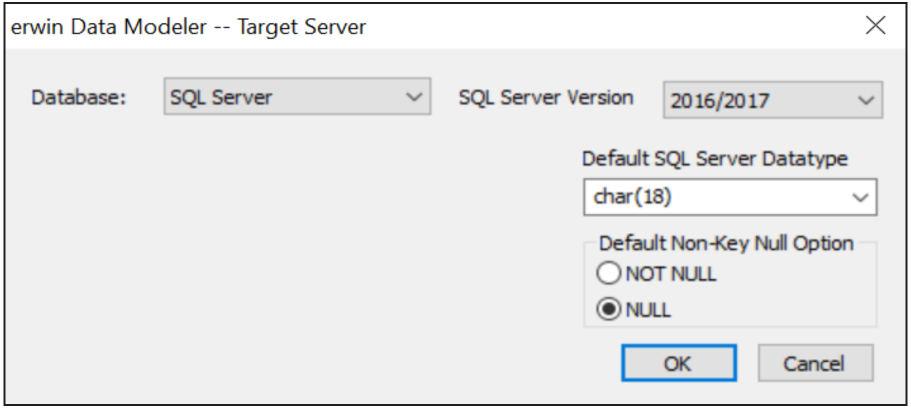
Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

☐ Subject Areas: 4 ☐ Indexes: 30 ☐ Vie
☐ Tables: 9 ☐ Relationships: 20 ☐ Ma
☐ Columns: 75 ☐ Sub-Categories: 0 ☐ Da

Style: Pie Style (3D) Pallete: Illustration

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.



erwin Data Modeler -- Target Server

Database: SQL Server

SQL Server Version: 2016/2017

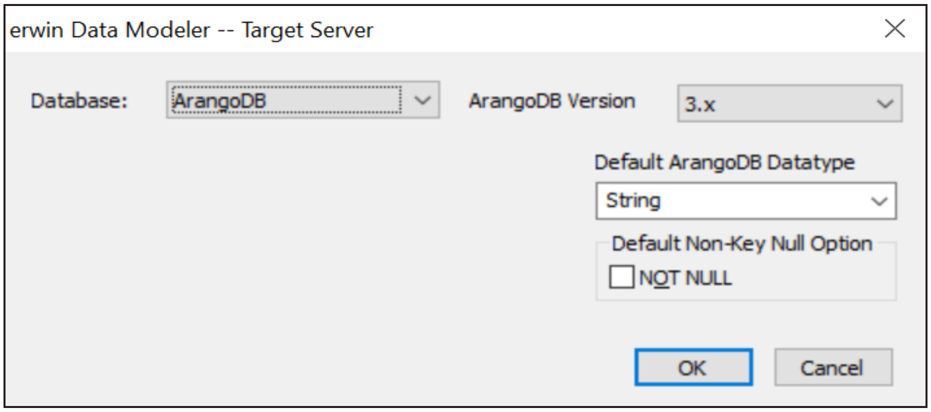
Default SQL Server Datatype: char(18)

Default Non-Key Null Option

○ NOT NULL

● NULL

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



erwin Data Modeler -- Target Server

Database: ArangoDB

ArangoDB Version: 3.x

Default ArangoDB Datatype: String

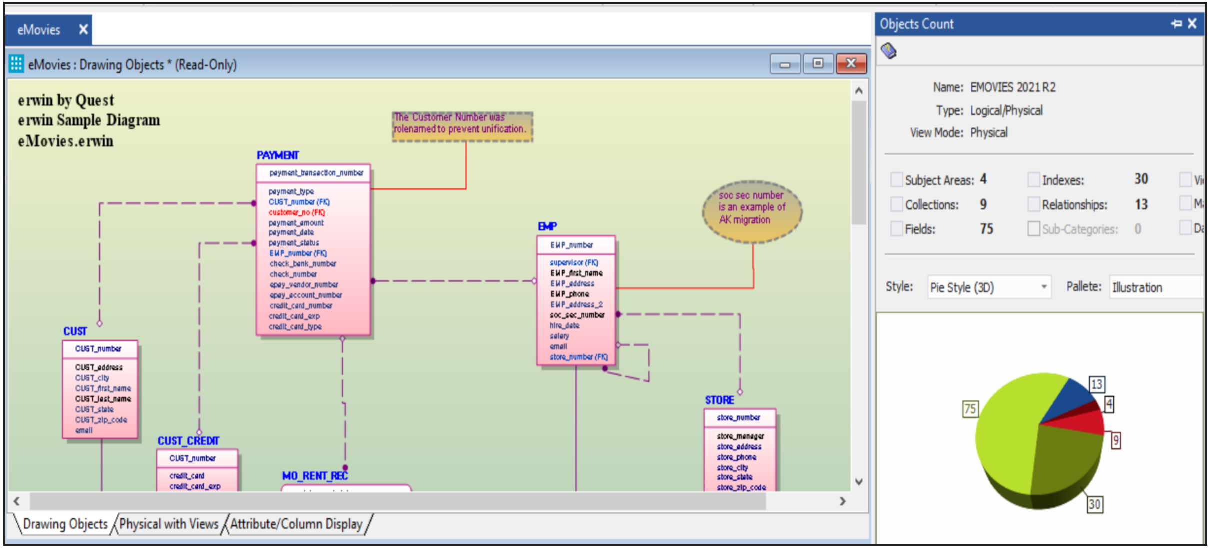
Default Non-Key Null Option

□ NOT NULL

4. Click **OK**.

   The conversion process starts.

   Once the conversion is complete, the existing model in migrated to an ArangoDB database.



Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 4
□ Collections: 9
□ Fields: 75
□ Indexes: 30
□ Relationships: 13
□ Sub-Categories: 0

Style: Pie Style (3D)
Pallete: Illustration

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.

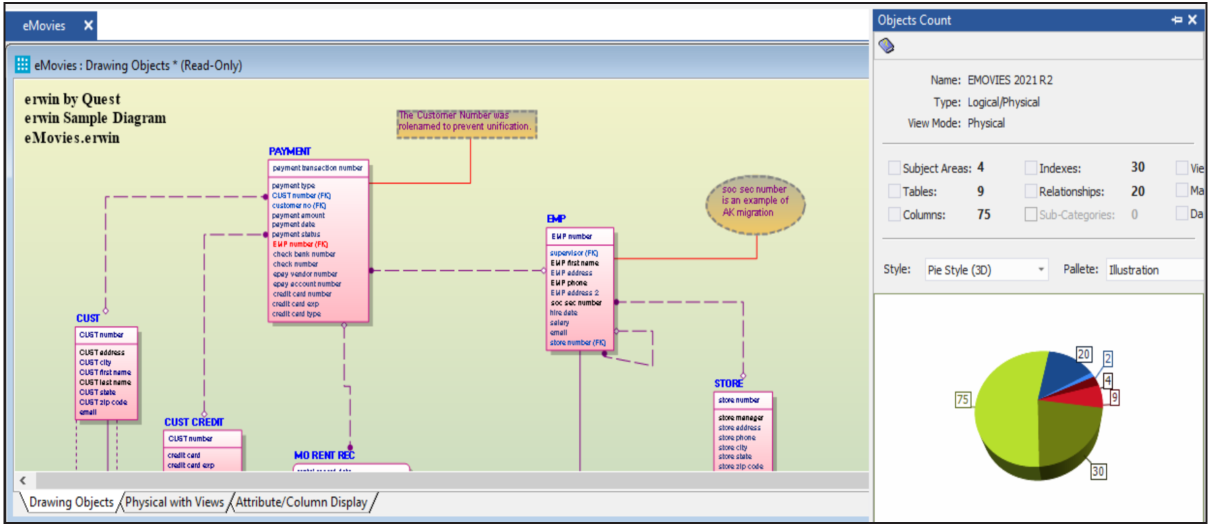
Migration by Deriving a Model

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.

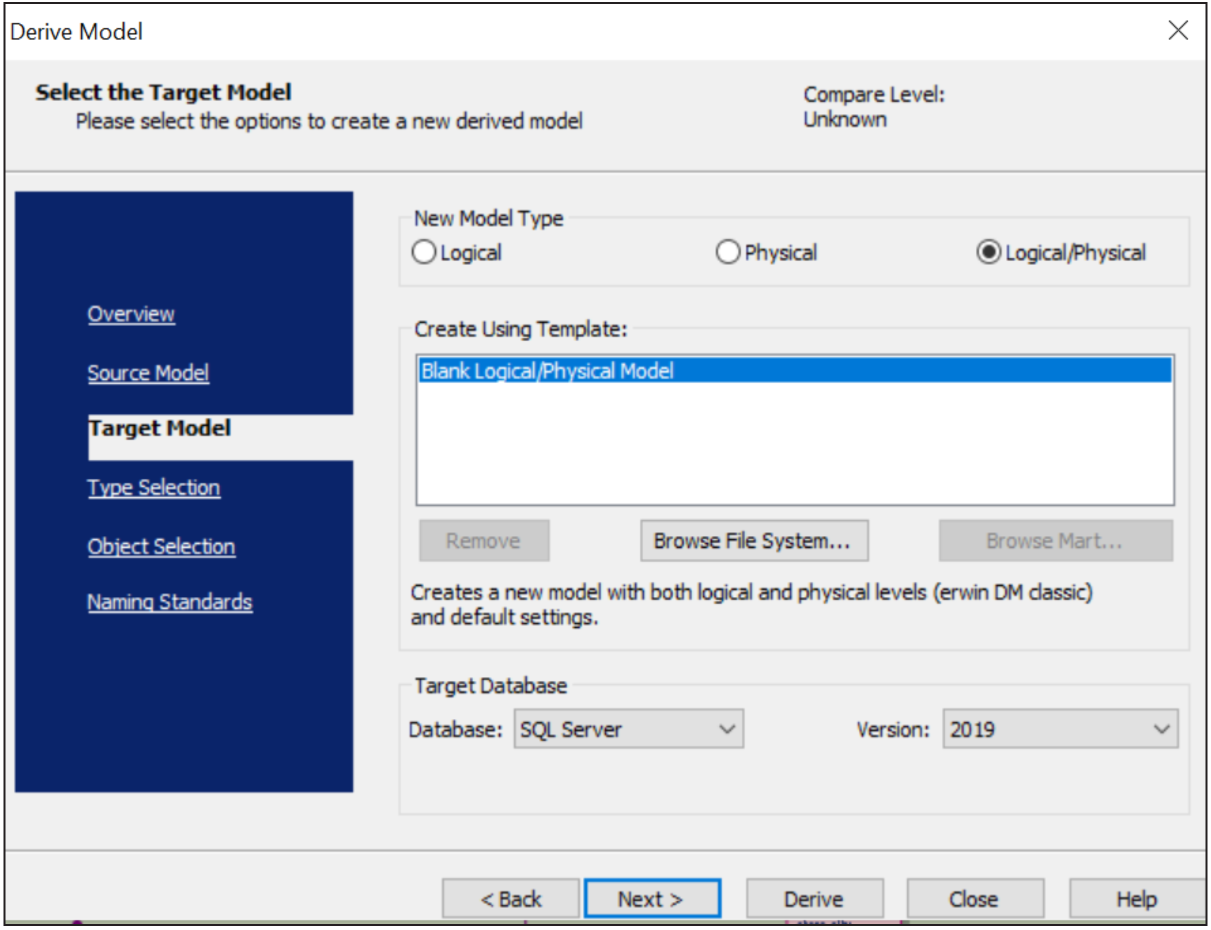


Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 4	□ Indexes: 30	□ Vie
□ Tables: 9	□ Relationships: 20	□ Ma
□ Columns: 75	□ Sub-Categories: 0	□ Da

Style: Pie Style (3D)   Pallete: Illustration

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.



Select the Target Model

Please select the options to create a new derived model

Compare Level:
Unknown

New Model Type

○ Logical ○ Physical ● Logical/Physical

Create Using Template:

Blank Logical/Physical Model

Creates a new model with both logical and physical levels (erwin DM classic) and default settings.

Target Database

Database: SQL Server

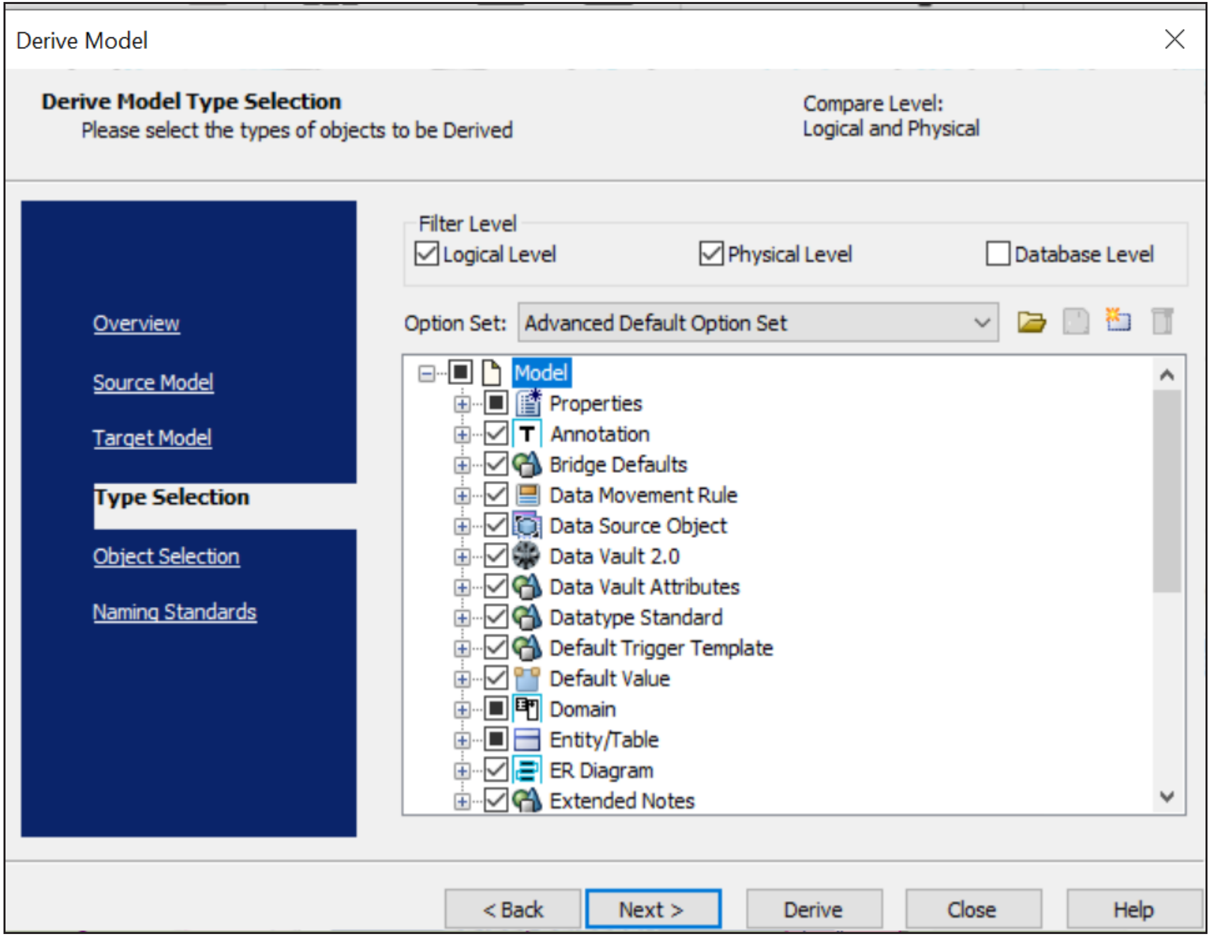
Version: 2019

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.



Derive Model

Derive Model Type Selection
Please select the types of objects to be Derived

Compare Level:
Logical and Physical

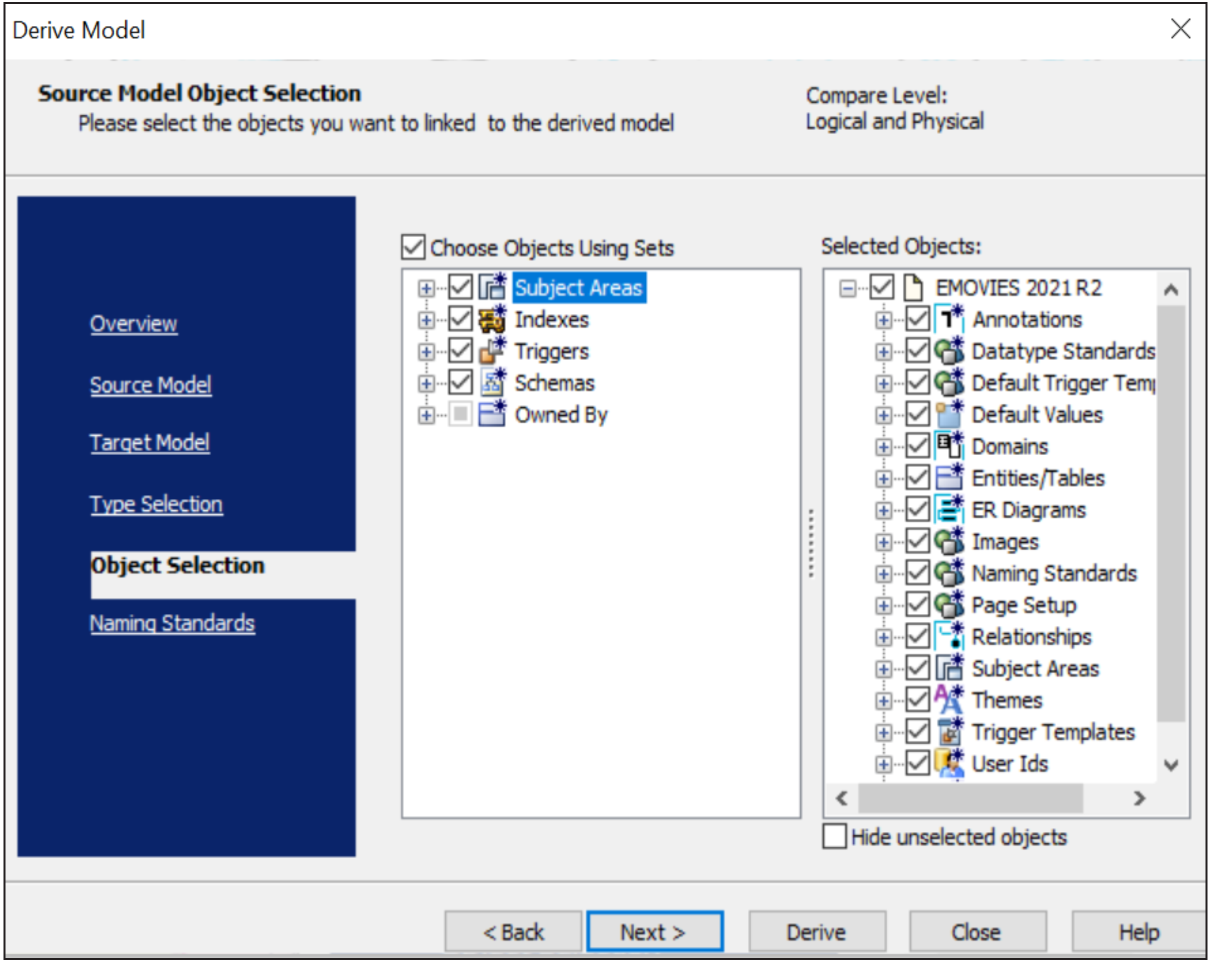
Filter Level
☑ Logical Level ☑ Physical Level □ Database Level

Option Set: Advanced Default Option Set

- ■ Model
  - ■ Properties
  - ☑ Annotation
  - ☑ Bridge Defaults
  - ☑ Data Movement Rule
  - ☑ Data Source Object
  - ☑ Data Vault 2.0
  - ☑ Data Vault Attributes
  - ☑ Datatype Standard
  - ☑ Default Trigger Template
  - ☑ Default Value
  - ■ Domain
  - ■ Entity/Table
  - ☑ ER Diagram
  - ☑ Extended Notes

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.


Derive Model

Source Model Object Selection
Please select the objects you want to linked to the derived model

Compare Level:
Logical and Physical

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- ☑ Schemas
- □ Owned By

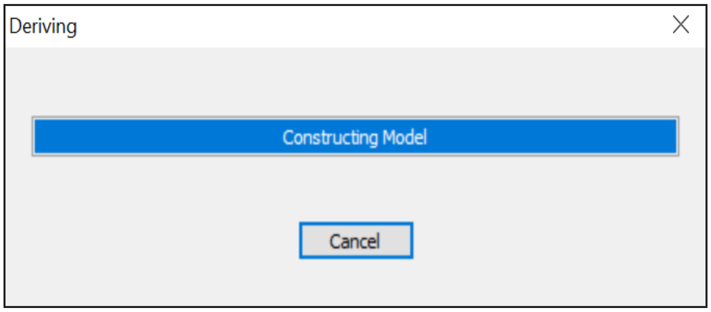
Selected Objects:

- ☑ EMOVIES 2021 R2
  - ☑ Annotations
  - ☑ Datatype Standards
  - ☑ Default Trigger Temp
  - ☑ Default Values
  - ☑ Domains
  - ☑ Entities/Tables
  - ☑ ER Diagrams
  - ☑ Images
  - ☑ Naming Standards
  - ☑ Page Setup
  - ☑ Relationships
  - ☑ Subject Areas
  - ☑ Themes
  - ☑ Trigger Templates
  - ☑ User Ids

□ Hide unselected objects

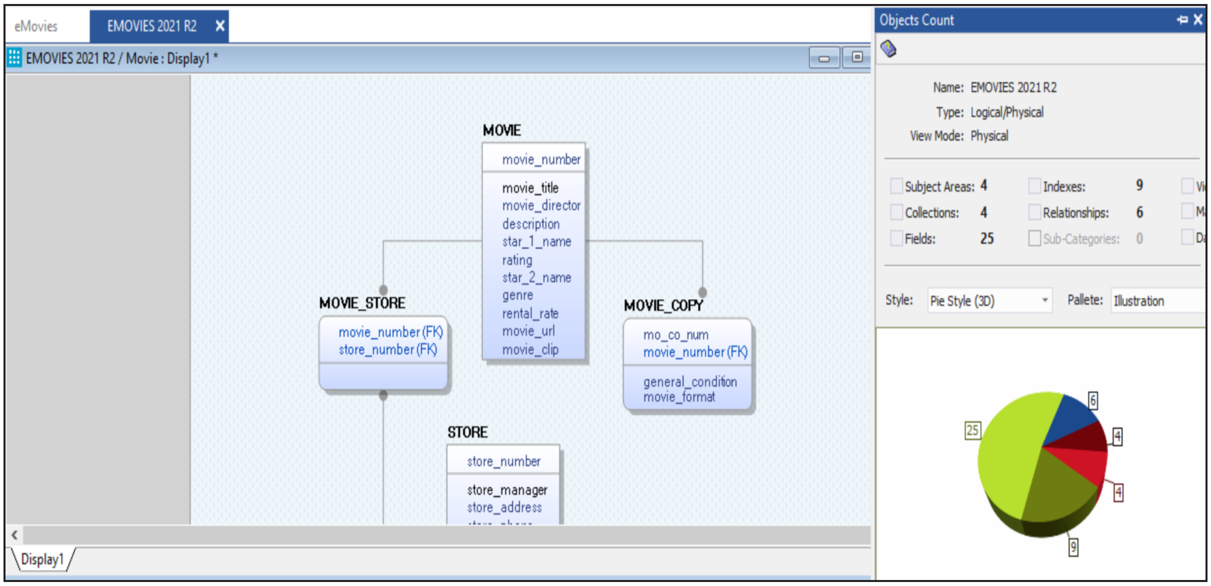
7. Select the objects that you want to derive into the target ArangoDB model.

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


Deriving

Constructing Model

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



Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	4	□ Indexes:	9
□ Collections:	4	□ Relationships:	6
□ Fields:	25	□ Sub-Categories:	0

Style: Pie Style (3D)
Pallete: Illustration

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

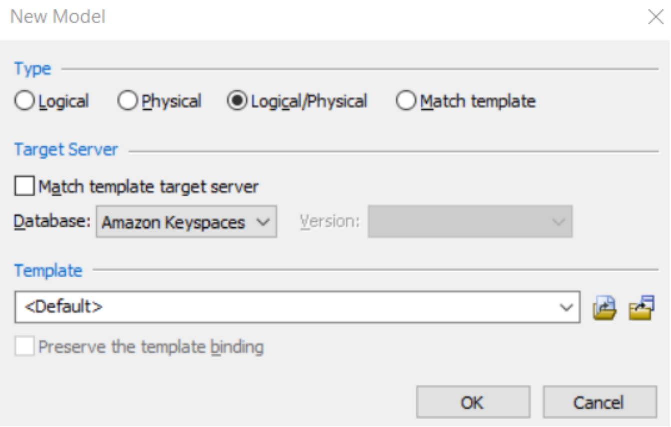
Reverse Engineering Models

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.

   New Model

   Type

   ○ Logical ○ Physical ● Logical/Physical ○ Match template

   Target Server

   □ Match template target server

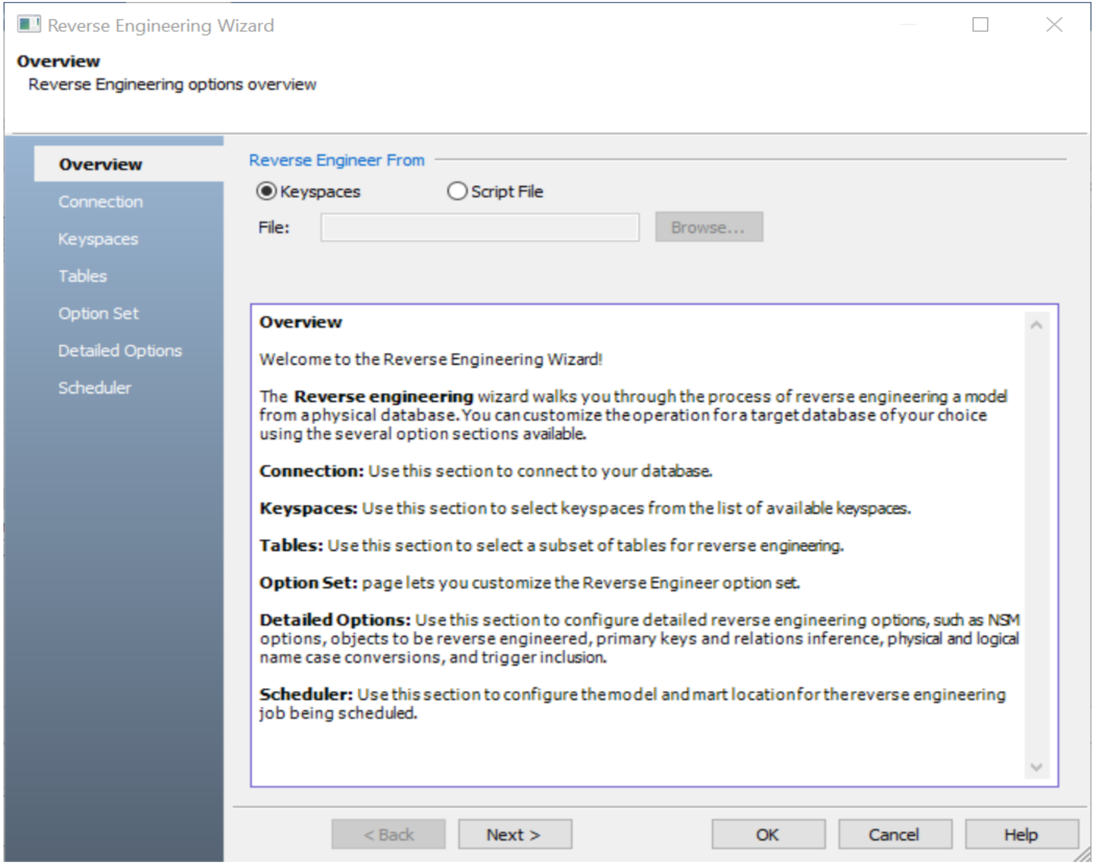
   Database: Amazon Keyspaces Version:

   Template

   <Default>

   □ Preserve the template binding

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



Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Keyspaces: Use this section to select keyspaces from the list of available keyspaces.

Tables: Use this section to select a subset of tables for reverse engineering.

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

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

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

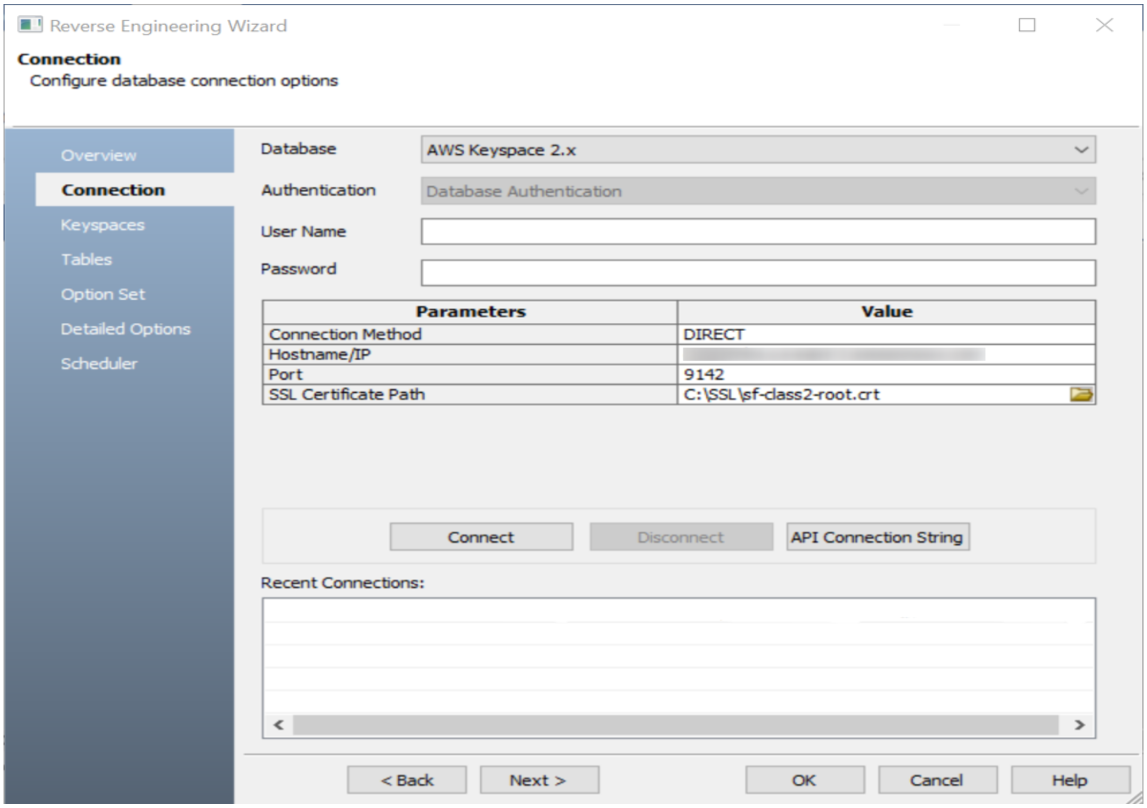
     > If you click **Keyspaces**, continue to step 5.

   - **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 section appears.



Connection
Configure database connection options

Database: AWS Keyspace 2.x

Authentication: Database Authentication

User Name:

Password:

Parameters	Value
Connection Method	DIRECT
Hostname/IP	
Port	9142
SSL Certificate Path	C:\SSL\sf-class2-root.crt

Recent Connections:

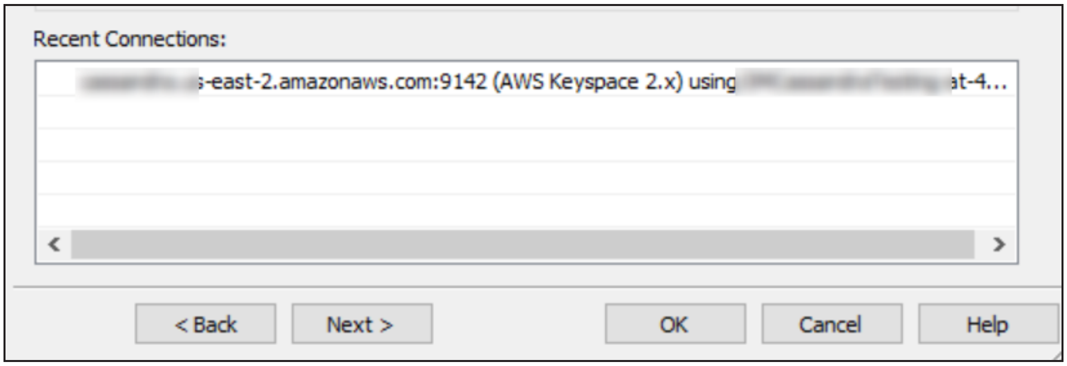
6. Enter your **User Name** and **Password**.

   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.

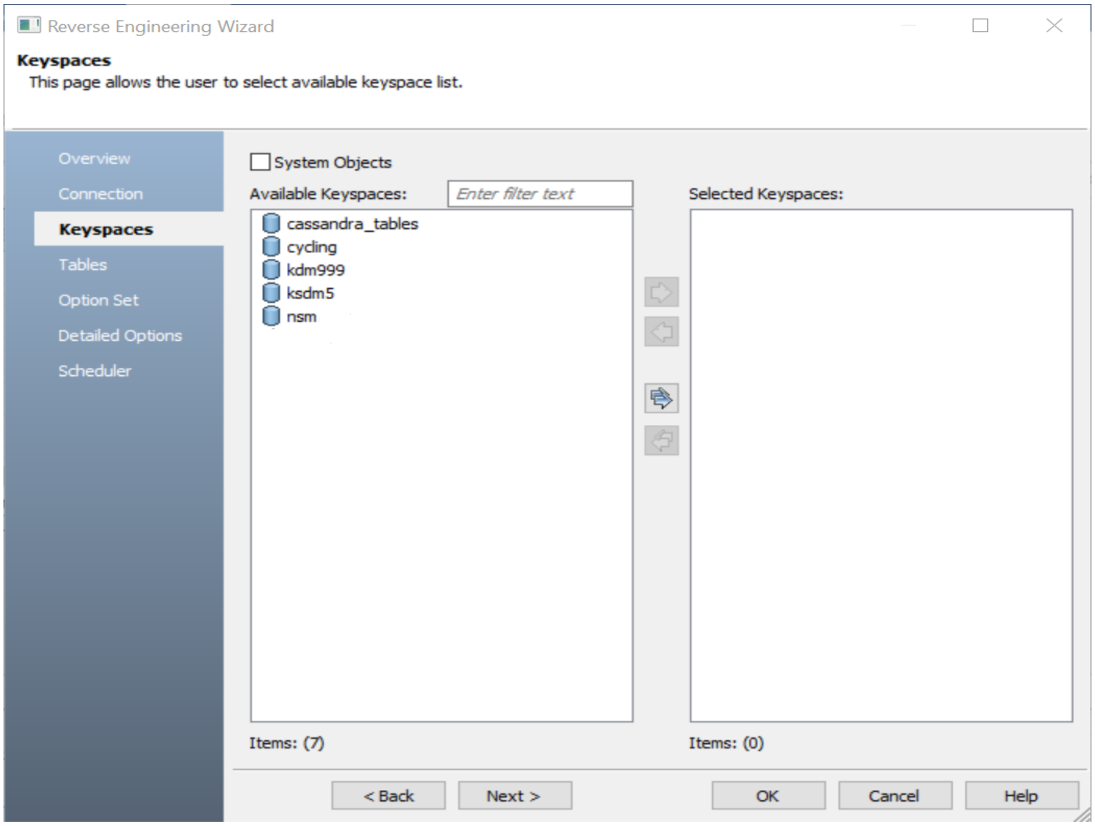


Recent Connections:

[redacted]s-east-2.amazonaws.com:9142 (AWS Keyspace 2.x) using [redacted]at-4...

8. Click **Next**.

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



Keyspaces

This page allows the user to select available keyspace list.

□ System Objects

Available Keyspaces: Enter filter text

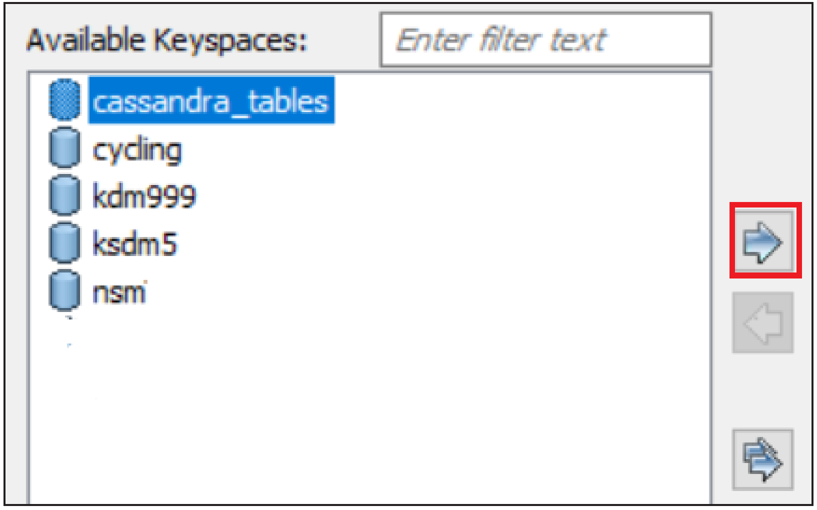
- cassandra_tables
- cycling
- kdm999
- ksdm5
- nsm

Items: (7)

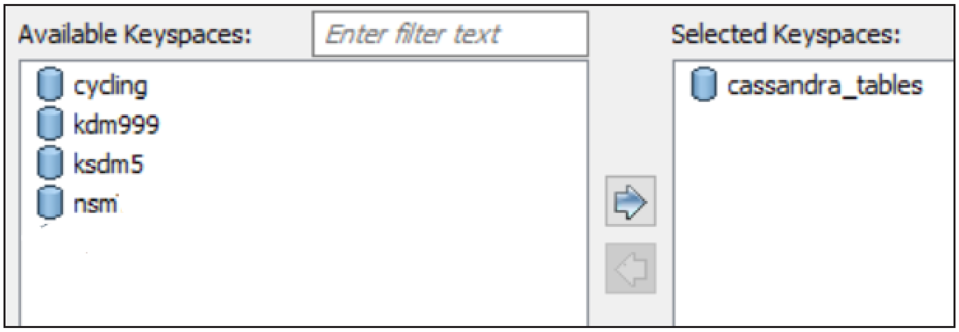
Selected Keyspaces:

Items: (0)

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

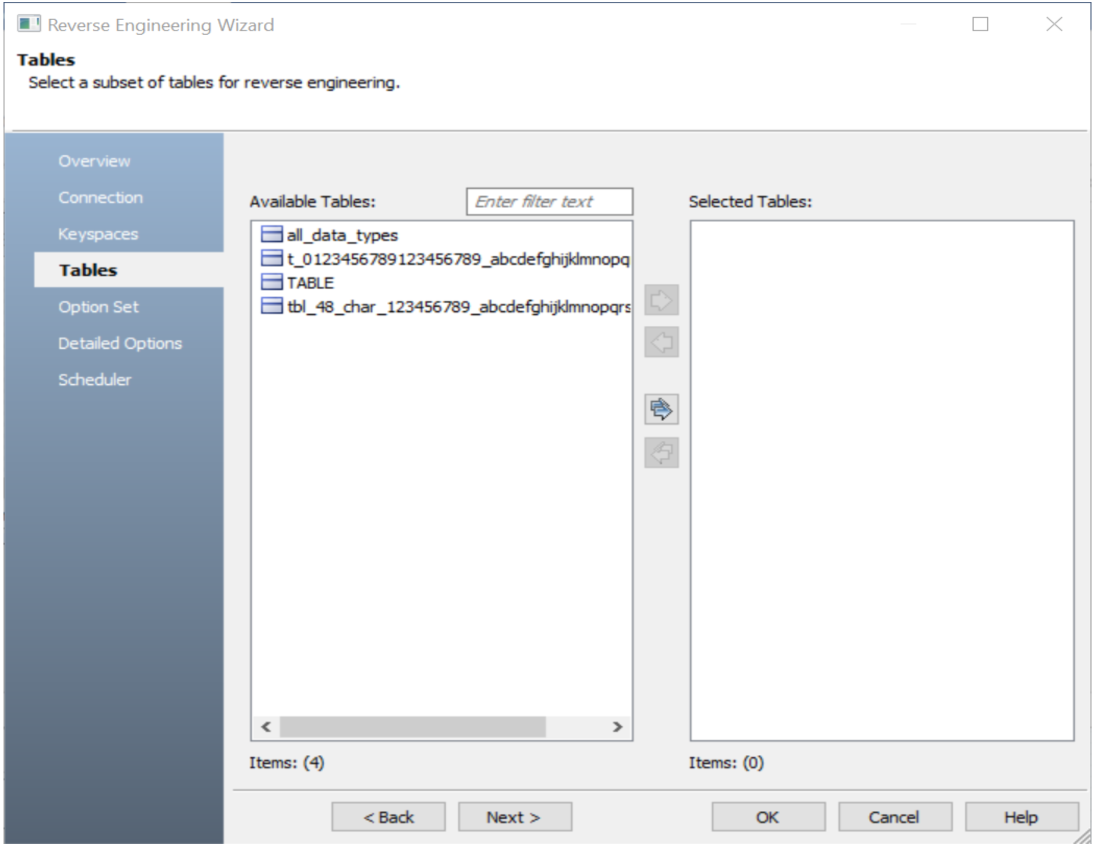


This moves the selected databases under Selected Keyspaces.



10. Click **Next**.

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



Tables

Select a subset of tables for reverse engineering.

Available Tables: Enter filter text

all_data_types

t_0123456789123456789_abcdefghijklmnopq

TABLE

tbl_48_char_123456789_abcdefghijklmnopqrs

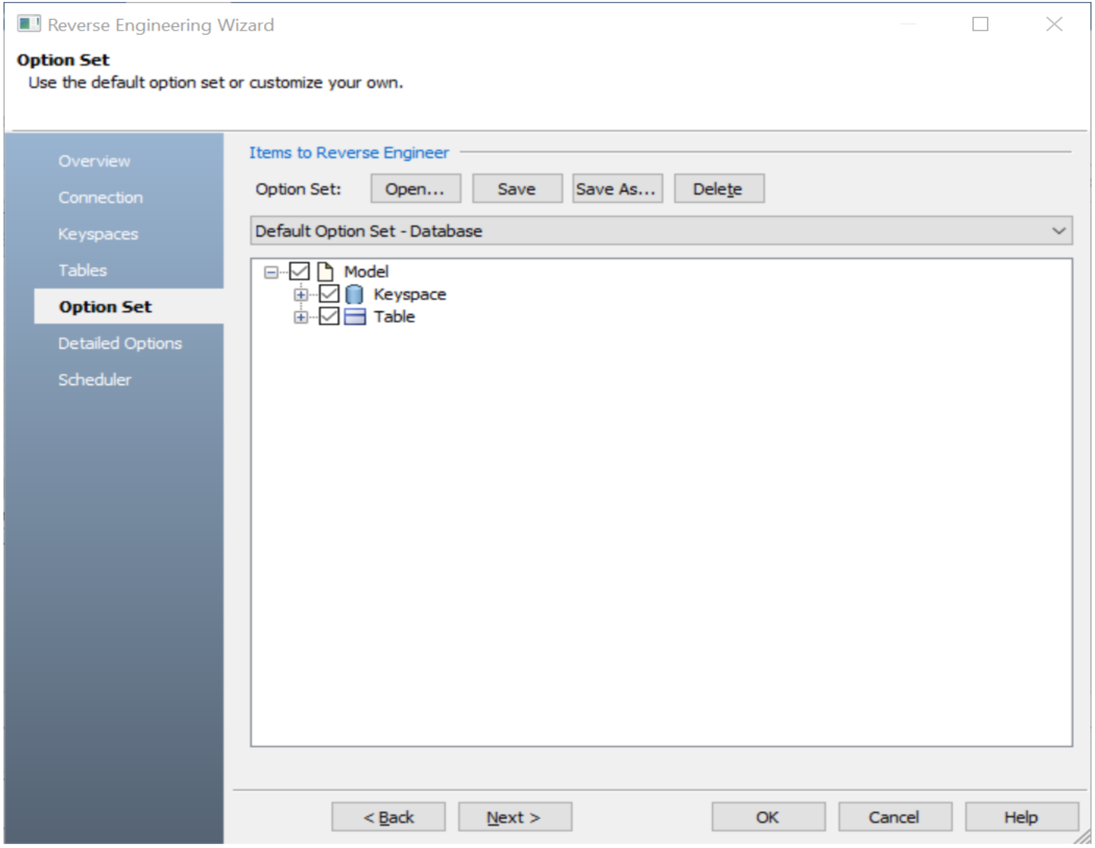
Items: (4)

Selected Tables:

Items: (0)

11. Click **Next**.

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



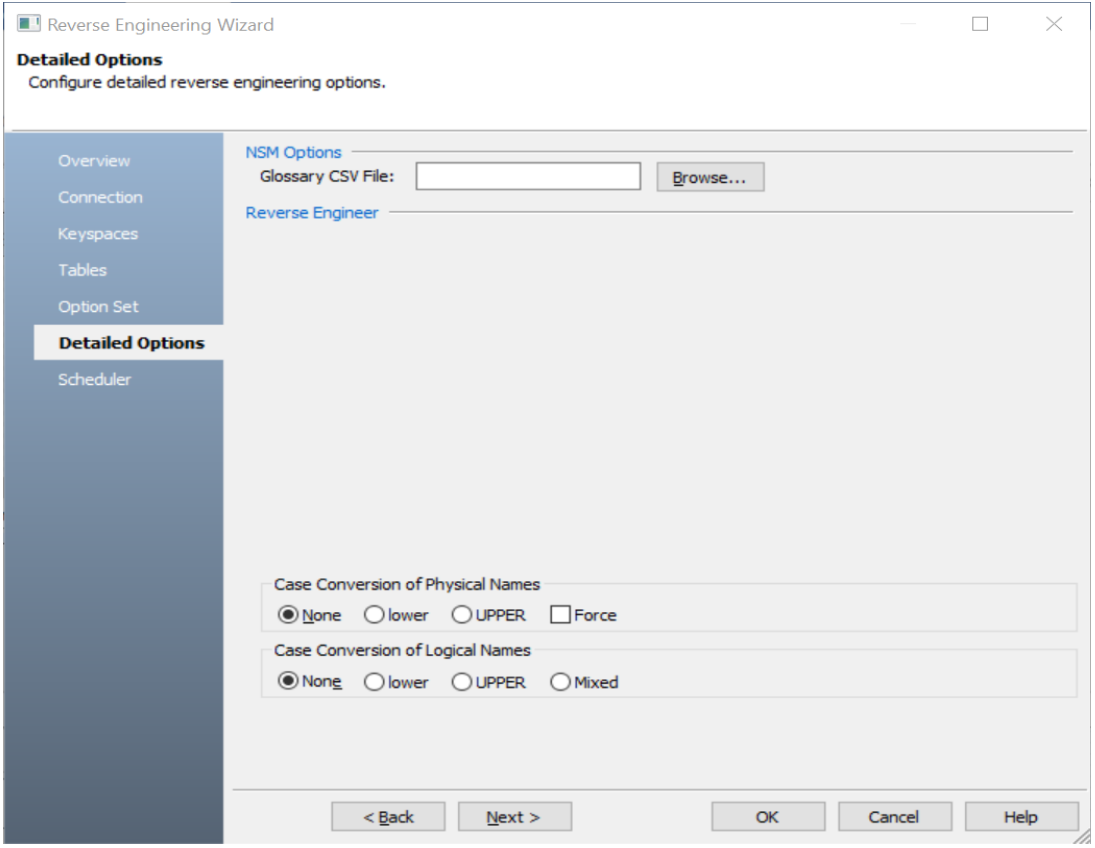
Option Set
Use the default option set or customize your own.

Option Set: Default Option Set - Database

- ☑ Model
  - ☑ Keyspace
  - ☑ Table

12. Click **Next**.

    The Detail Options section appears. Set up appropriate options based on your requirement.



Detailed Options

Configure detailed reverse engineering options.

NSM Options

Glossary CSV File:

Reverse Engineer

Case Conversion of Physical Names

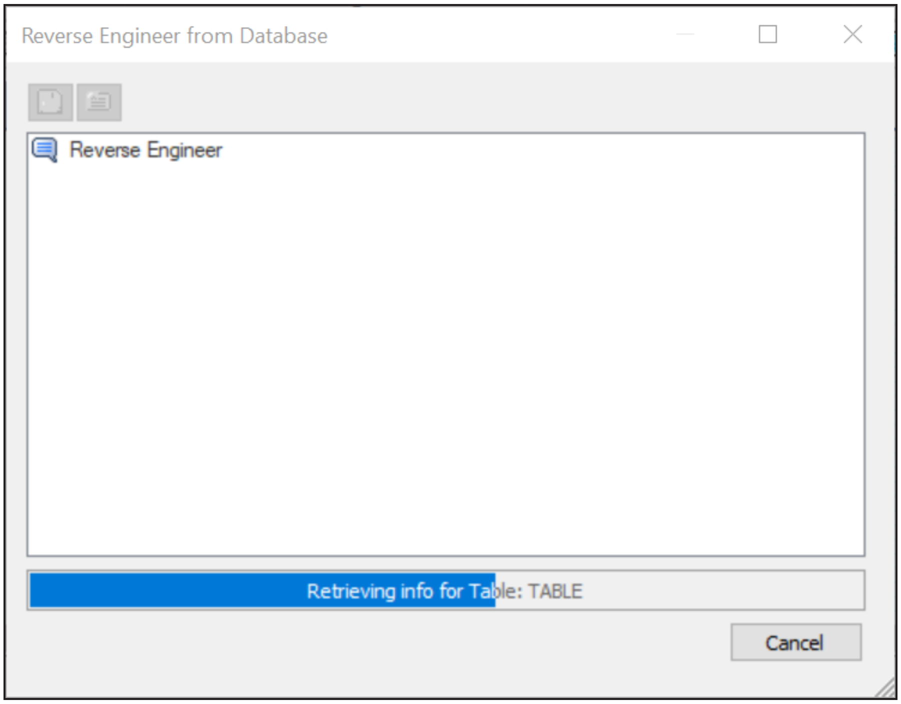
● None ○ lower ○ UPPER □ Force

Case Conversion of Logical Names

● None ○ lower ○ UPPER ○ Mixed

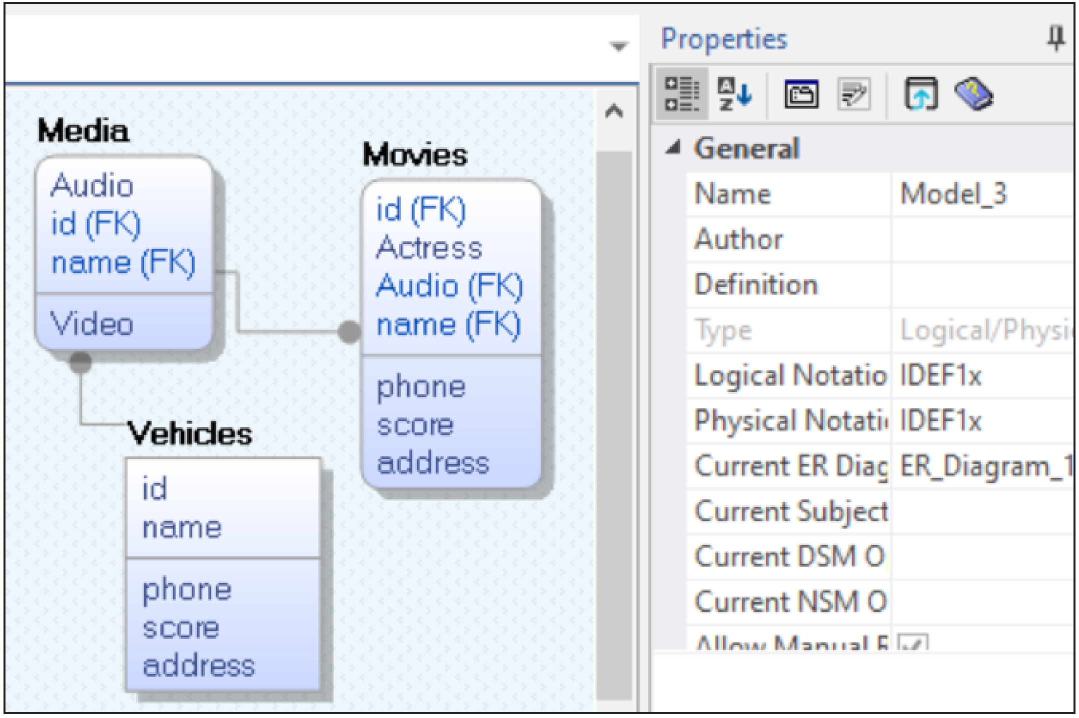
13. Click **OK**.

The reverse engineering process starts.



Retrieving info for Table: TABLE

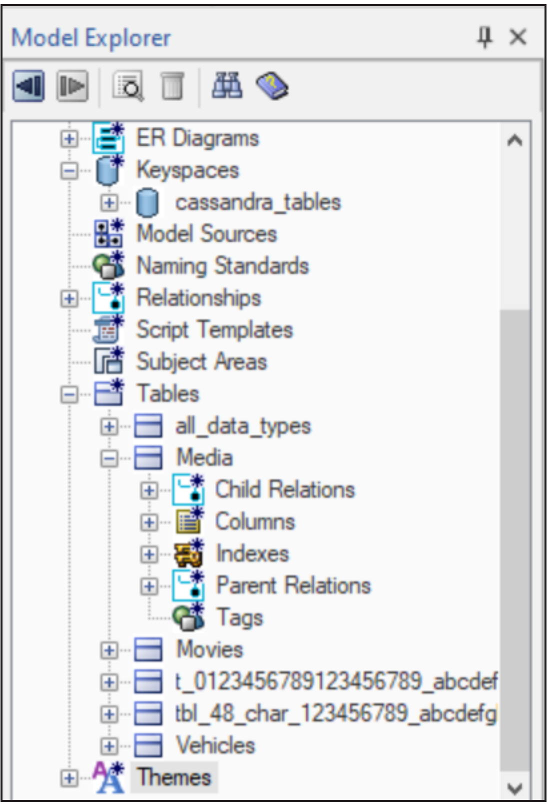
Once the process is complete, based on your selections, a schema is generated and a model is created.



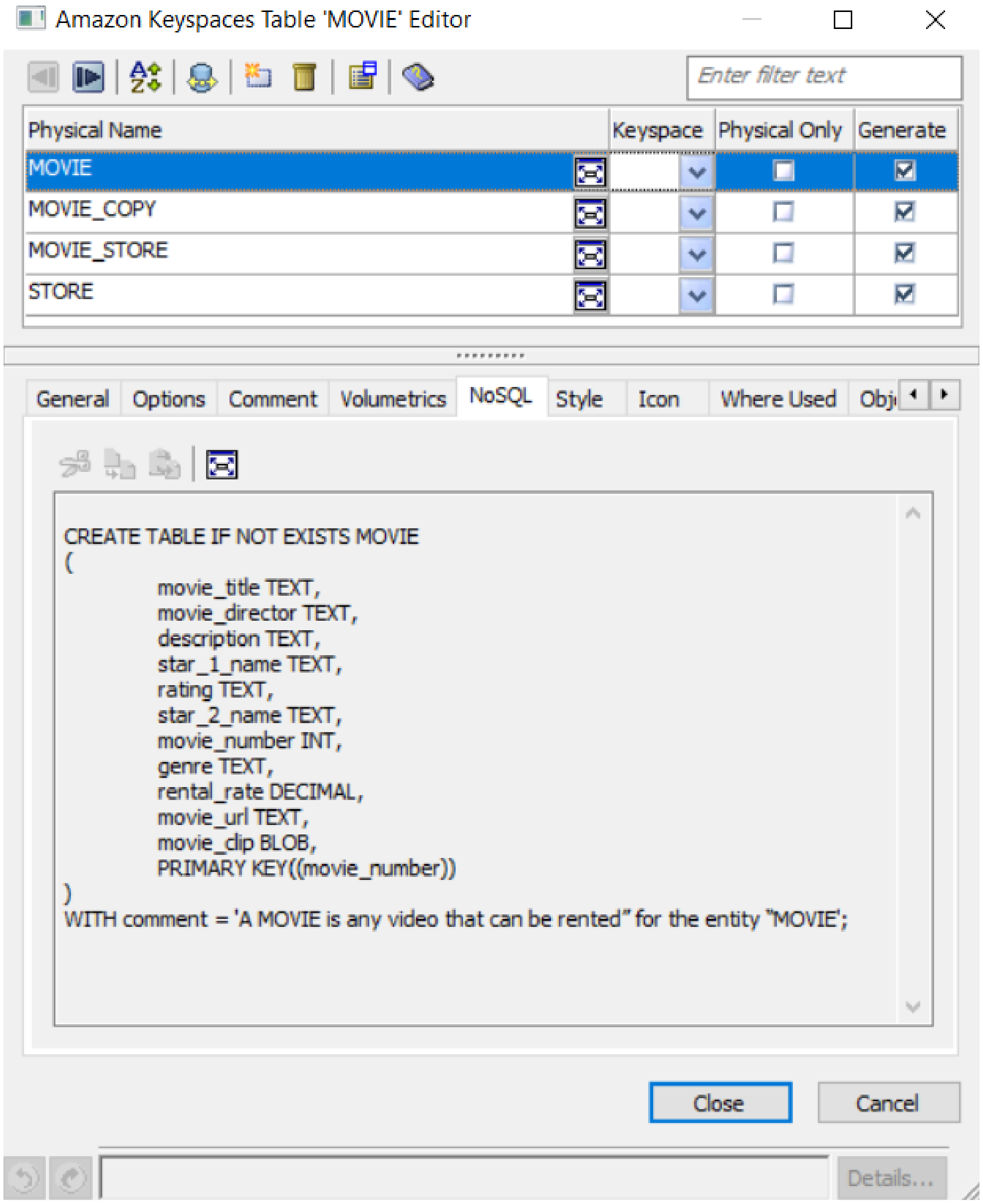
Name	Model_3
Author	
Definition	
Type	Logical/Physi
Logical Notatio	IDEF1x
Physical Notati	IDEF1x
Current ER Diag	ER_Diagram_1
Current Subject	
Current DSM O	
Current NSM O	
Allow Manual F	☑

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 retrived.



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.



Physical Name	Keyspace	Physical Only	Generate
MOVIE		□	☑
MOVIE_COPY		□	☑
MOVIE_STORE		□	☑
STORE		□	☑

```
CREATE TABLE IF NOT EXISTS MOVIE
(
            movie_title TEXT,
            movie_director TEXT,
            description TEXT,
            star_1_name TEXT,
            rating TEXT,
            star_2_name TEXT,
            movie_number INT,
            genre TEXT,
            rental_rate DECIMAL,
            movie_url TEXT,
            movie_clip BLOB,
            PRIMARY KEY((movie_number))
)
WITH comment = 'A MOVIE is any video that can be rented'' for the entity ''MOVIE';
```

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 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:	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 where the reverse engineered model should be saved and its name	When you schedule a job on a remote server, ensure the model path is same for remote and local server. For example: C:\Scheduler\<Model Name>.erwin
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 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.

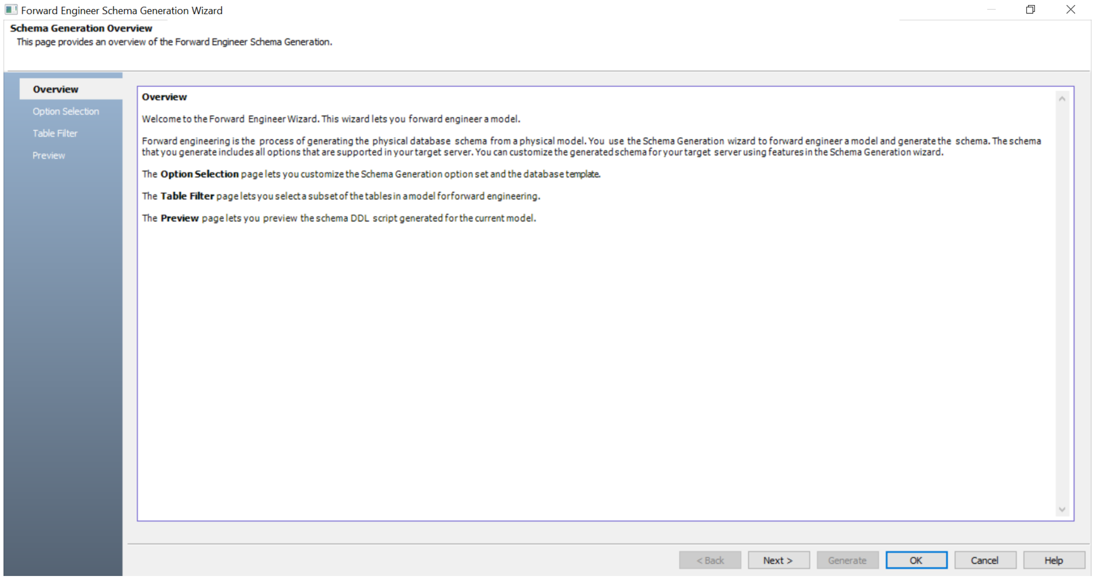
This topic walks you through the steps to forward engineer an Amazon Keyspaces model. For detailed description of forward engineering options, refer to the Forward Engineering Options topic.

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.



Forward Engineer Schema Generation Wizard

Schema Generation Overview

This page provides an overview of the Forward Engineer Schema Generation.

Overview

Welcome to the Forward Engineer Wizard. This wizard lets you forward engineer a model.

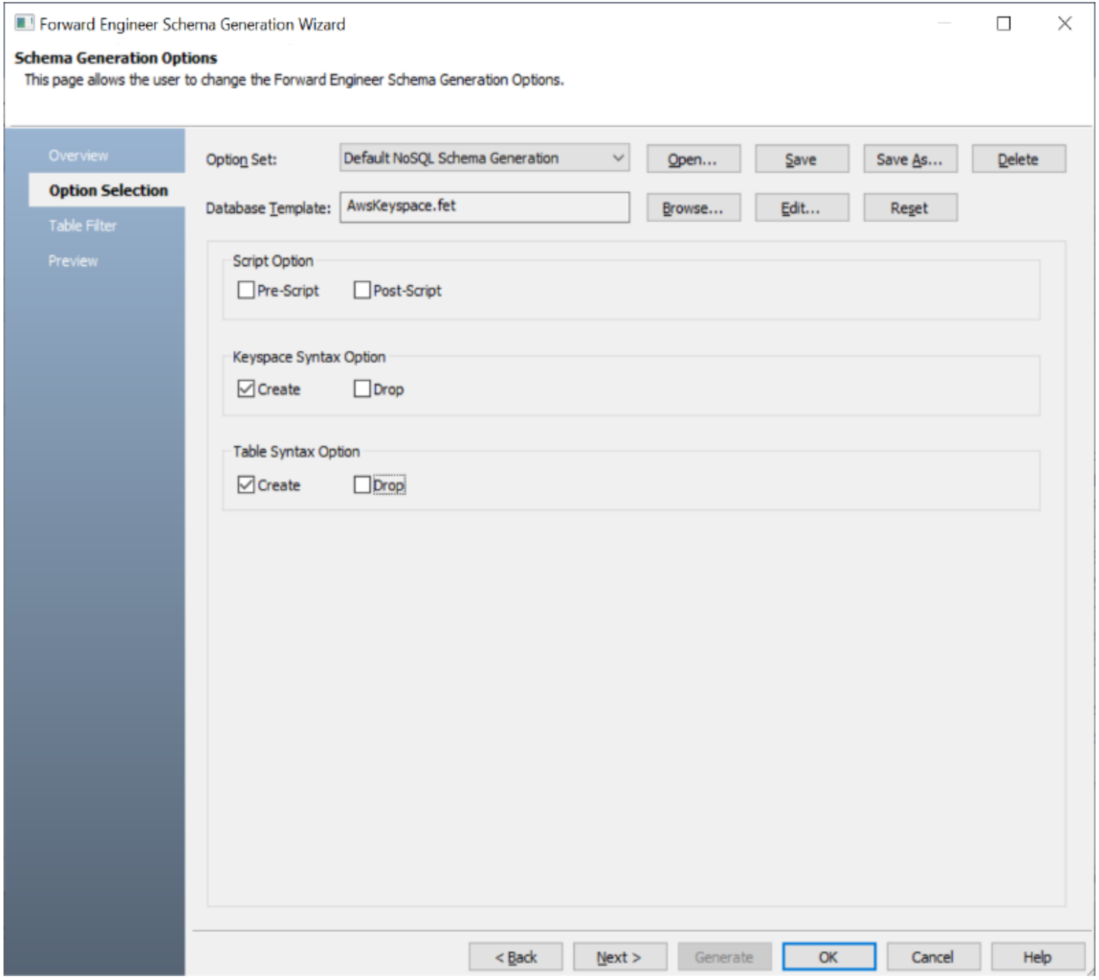
Forward engineering is the process of generating the physical database schema from a physical model. You use the Schema Generation wizard to forward engineer a model and generate the schema. The schema that you generate includes all options that are supported in your target server. You can customize the generated schema for your target server using features in the Schema Generation wizard.

The **Option Selection** page lets you customize the Schema Generation option set and the database template.

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

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

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



Forward Engineer Schema Generation Wizard

Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default NoSQL Schema Generation

Database Template: AwsKeyspace.fet

Script Option

□ Pre-Script □ Post-Script

Keyspace Syntax Option

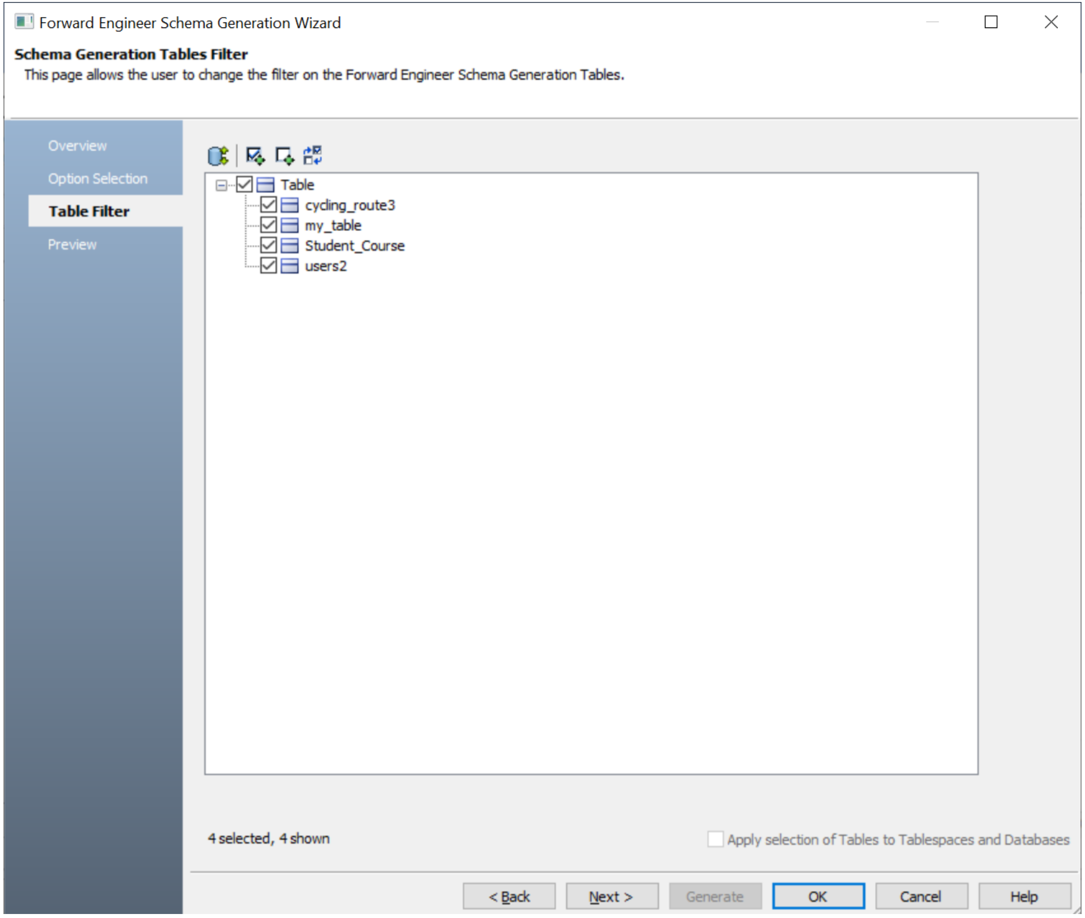
☑ Create □ Drop

Table Syntax Option

☑ Create □ Drop

4. Click **Next**.

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



Forward Engineer Schema Generation Wizard

Schema Generation Tables Filter

This page allows the user to change the filter on the Forward Engineer Schema Generation Tables.

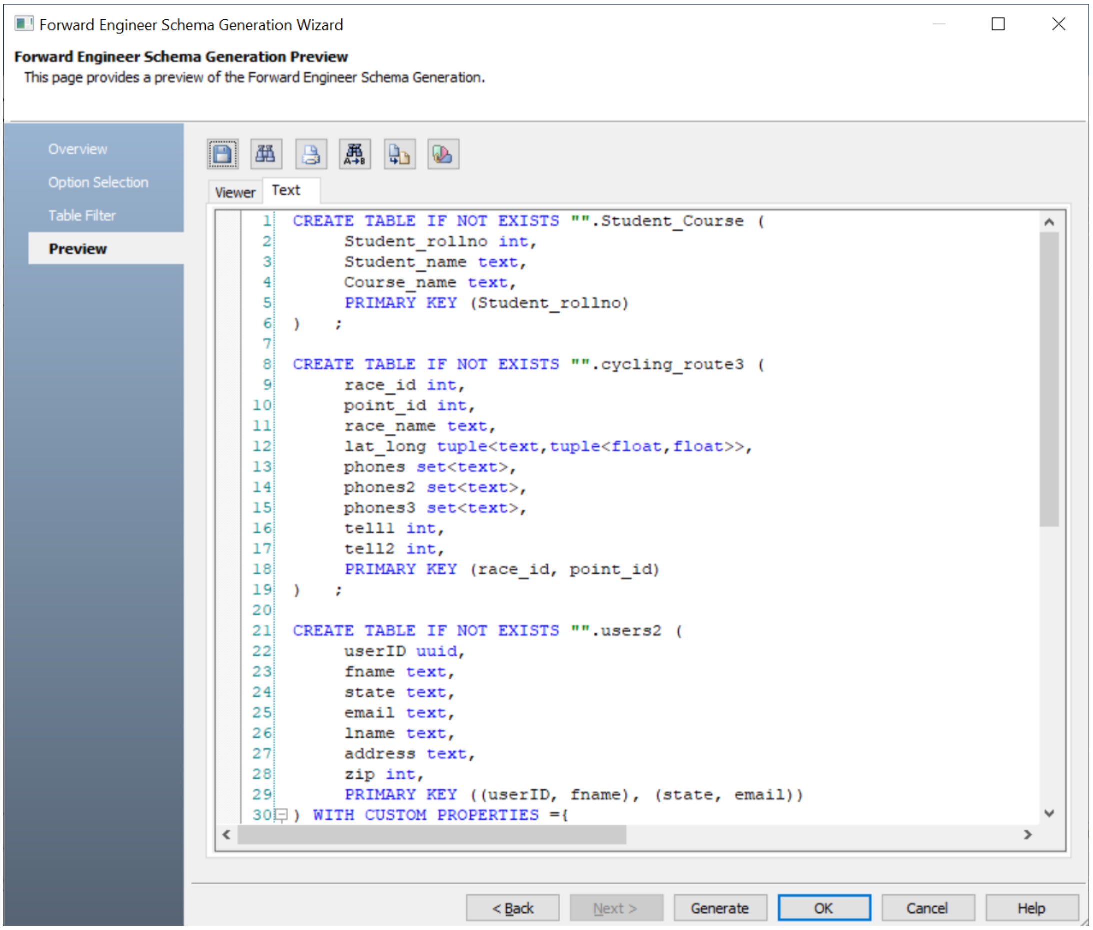
- ☑ Table
  - ☑ cycling_route3
  - ☑ my_table
  - ☑ Student_Course
  - ☑ users2

4 selected, 4 shown

□ Apply selection of Tables to Tablespaces and Databases

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

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



Forward Engineer Schema Generation Preview

This page provides a preview of the Forward Engineer Schema Generation.

```
CREATE TABLE IF NOT EXISTS "".Student_Course (
     Student_rollno int,
     Student_name text,
     Course_name text,
     PRIMARY KEY (Student_rollno)
)   ;

CREATE TABLE IF NOT EXISTS "".cycling_route3 (
     race_id int,
     point_id int,
     race_name text,
     lat_long tuple<text,tuple<float,float>>,
     phones set<text>,
     phones2 set<text>,
     phones3 set<text>,
     tell1 int,
     tell2 int,
     PRIMARY KEY (race_id, point_id)
)   ;

CREATE TABLE IF NOT EXISTS "".users2 (
     userID uuid,
     fname text,
     state text,
     email text,
     lname text,
     address text,
     zip int,
     PRIMARY KEY ((userID, fname), (state, email))
) WITH CUSTOM PROPERTIES ={
```

Use the following options:

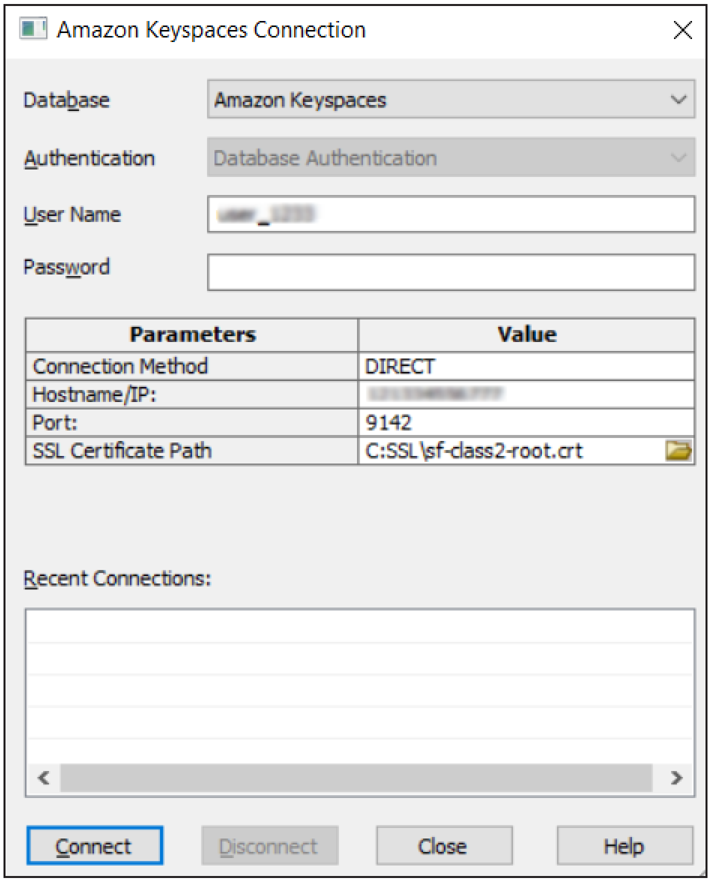
- **Copy** (⎘): Use this option to copy the script.
- **Save** (💾): Use this option to save the generated script in the CQL or SQL format.
- **Search** (🔍): Use this option to search through the generated schema.
- **Print** (🖶): Use this option to print the generated schema.
- **Replace** (🖶): Use this 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 editor appears.



Amazon Keyspaces Connection

Database: Amazon Keyspaces

Authentication: Database Authentication

User Name: [redacted]

Password:

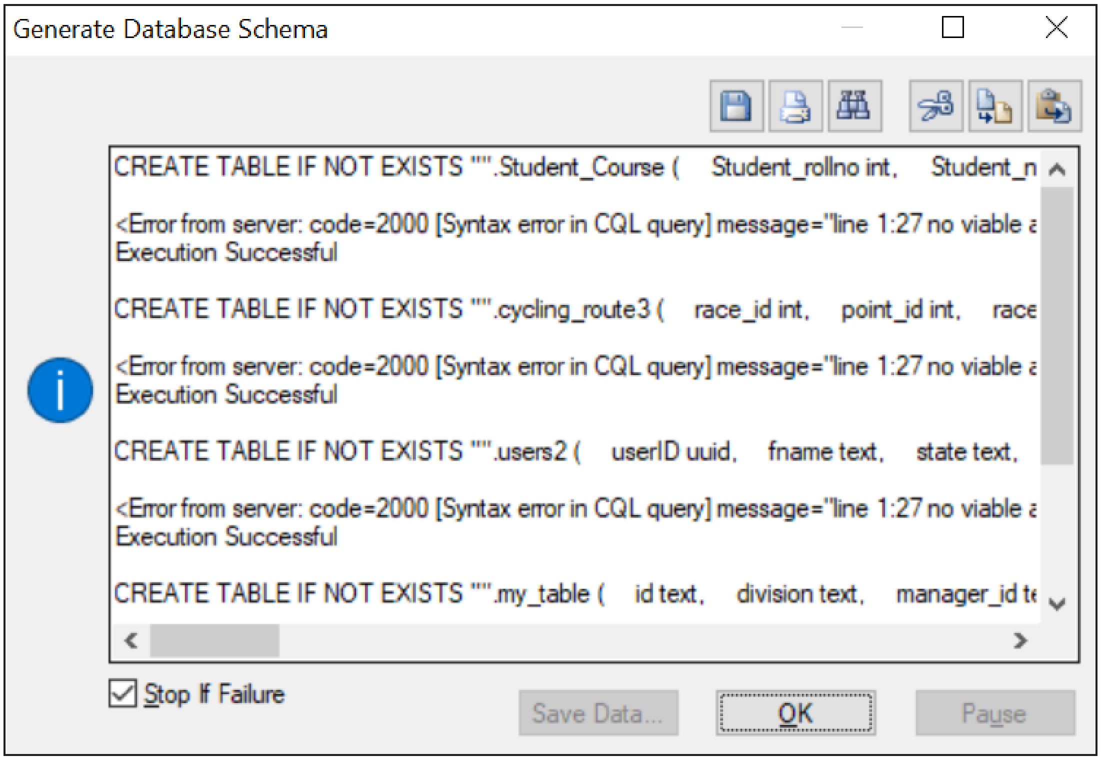
Parameters	Value
Connection Method	DIRECT
Hostname/IP:	[redacted]
Port:	9142
SSL Certificate Path	C:SSL\sf-class2-root.crt

Recent Connections:

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.



```
CREATE TABLE IF NOT EXISTS "".Student_Course (    Student_rollno int,    Student_n

<Error from server: code=2000 [Syntax error in CQL query] message="line 1:27 no viable a
Execution Successful

CREATE TABLE IF NOT EXISTS "".cycling_route3 (    race_id int,    point_id int,    race

<Error from server: code=2000 [Syntax error in CQL query] message="line 1:27 no viable a
Execution Successful

CREATE TABLE IF NOT EXISTS "".users2 (    userID uuid,    fname text,    state text,

<Error from server: code=2000 [Syntax error in CQL query] message="line 1:27 no viable a
Execution Successful

CREATE TABLE IF NOT EXISTS "".my_table (    id text,    division text,    manager_id te
```

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.

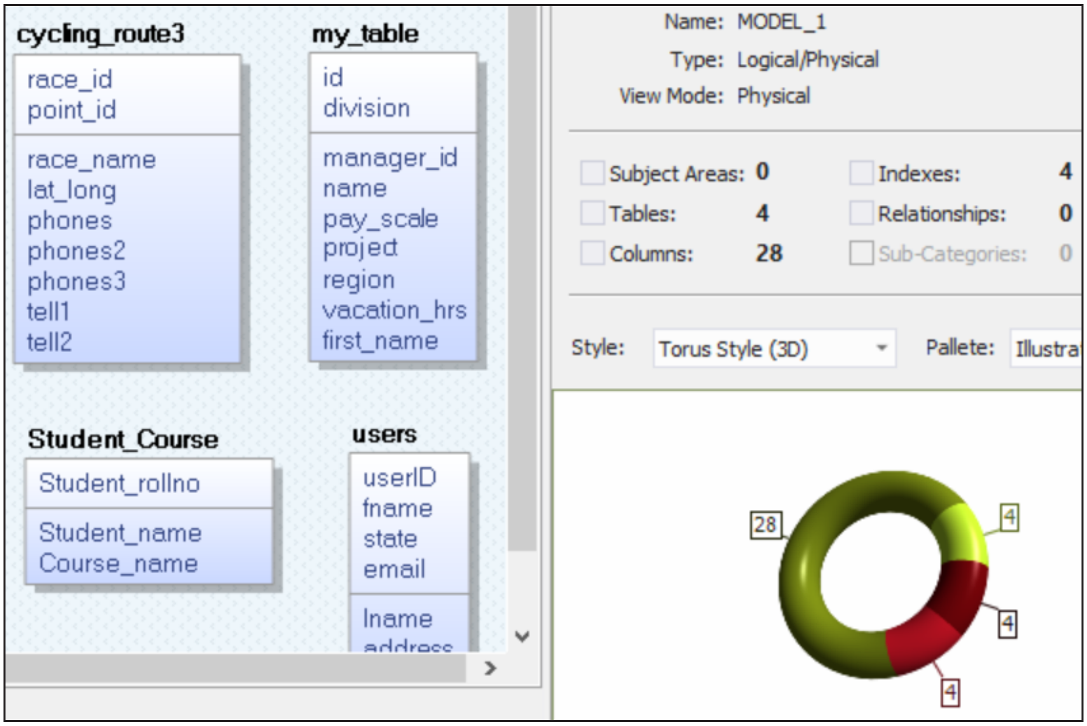
This topic walks you through the steps to compare an Amazon Keyspaces model with database.

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.



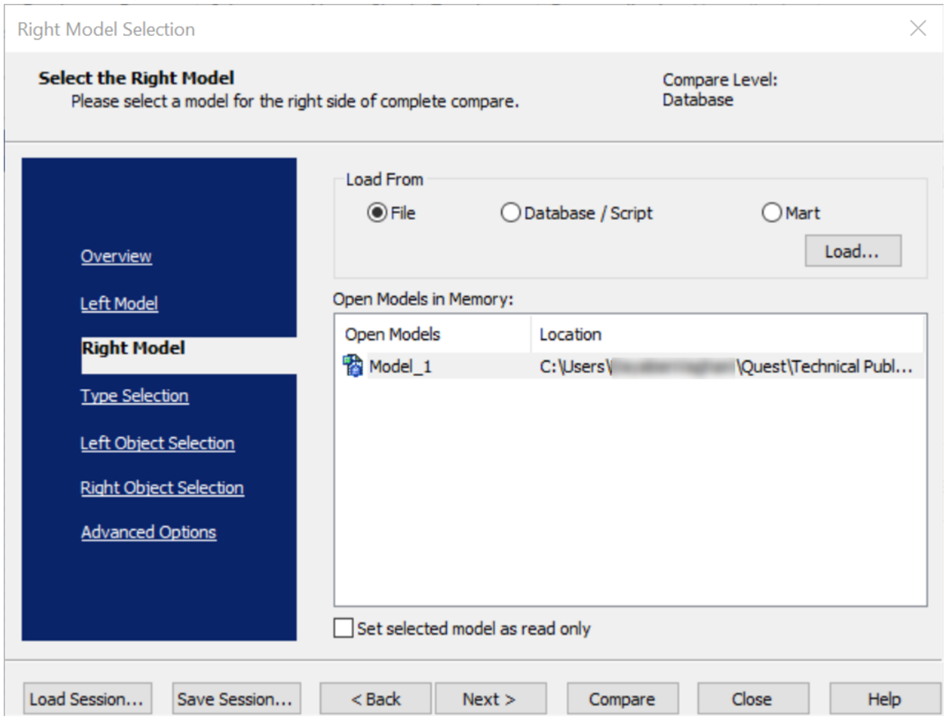
Name: MODEL_1
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 0
□ Tables: 4
□ Columns: 28
□ Indexes: 4
□ Relationships: 0
□ Sub-Categories: 0

Style: Torus Style (3D)
Pallete: Illustra

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.



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

● File ○ Database / Script ○ Mart

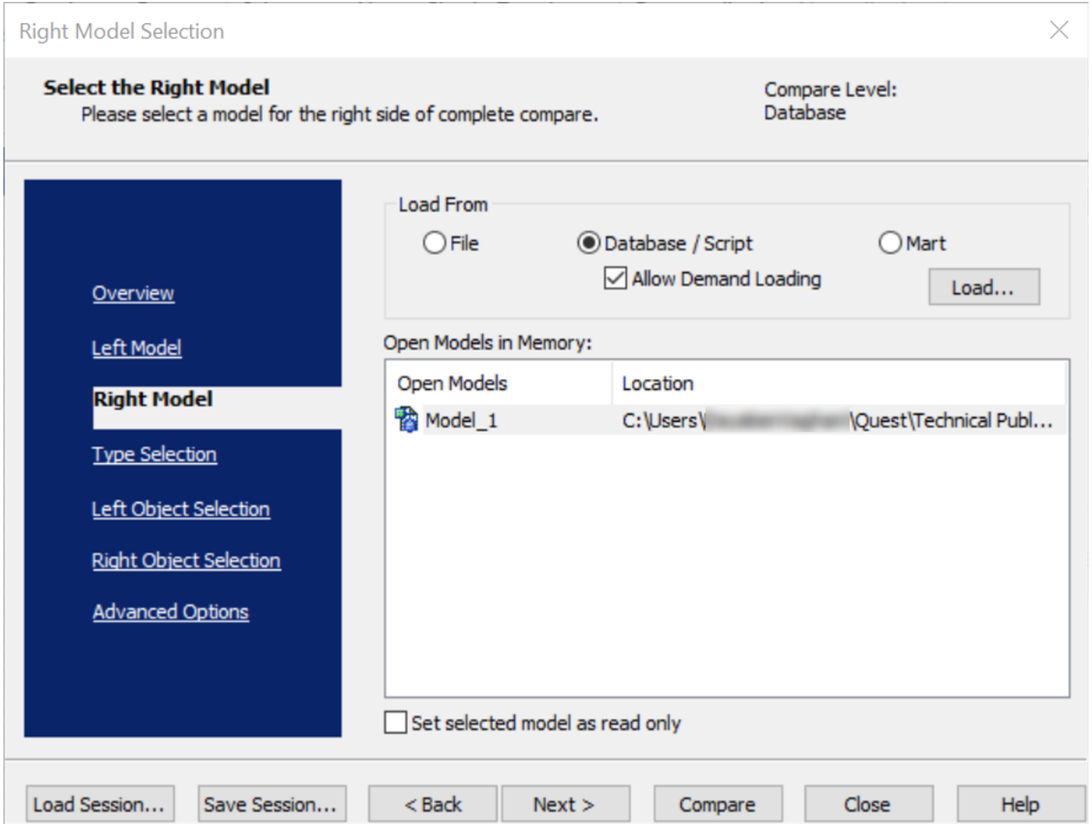
Open Models in Memory:

Open Models	Location
Model_1	C:\Users\[redacted]\Quest\Technical Publ...

□ Set selected model as read only

3. Click **Database/Script**.

   By default, the Allow Demand Loading option is selected.



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level: Database

Load From: ○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

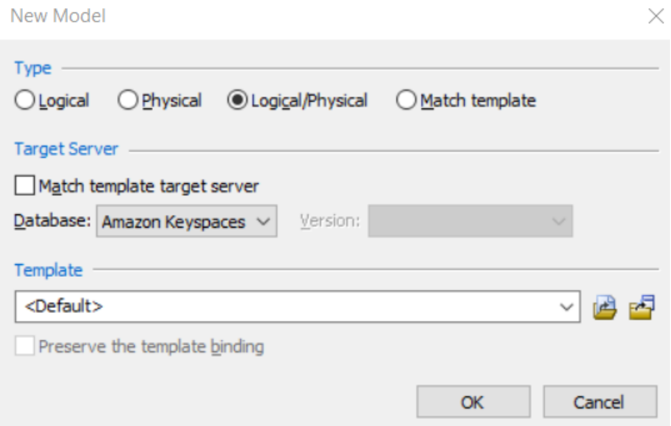
Open Models in Memory:

Open Models	Location
Model_1	C:\Users\[redacted]\Quest\Technical Publ...

☐ Set selected model as read only

4. Click **Load**.

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



New Model

Type: ○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

☐ Match template target server

Database: Amazon Keyspaces

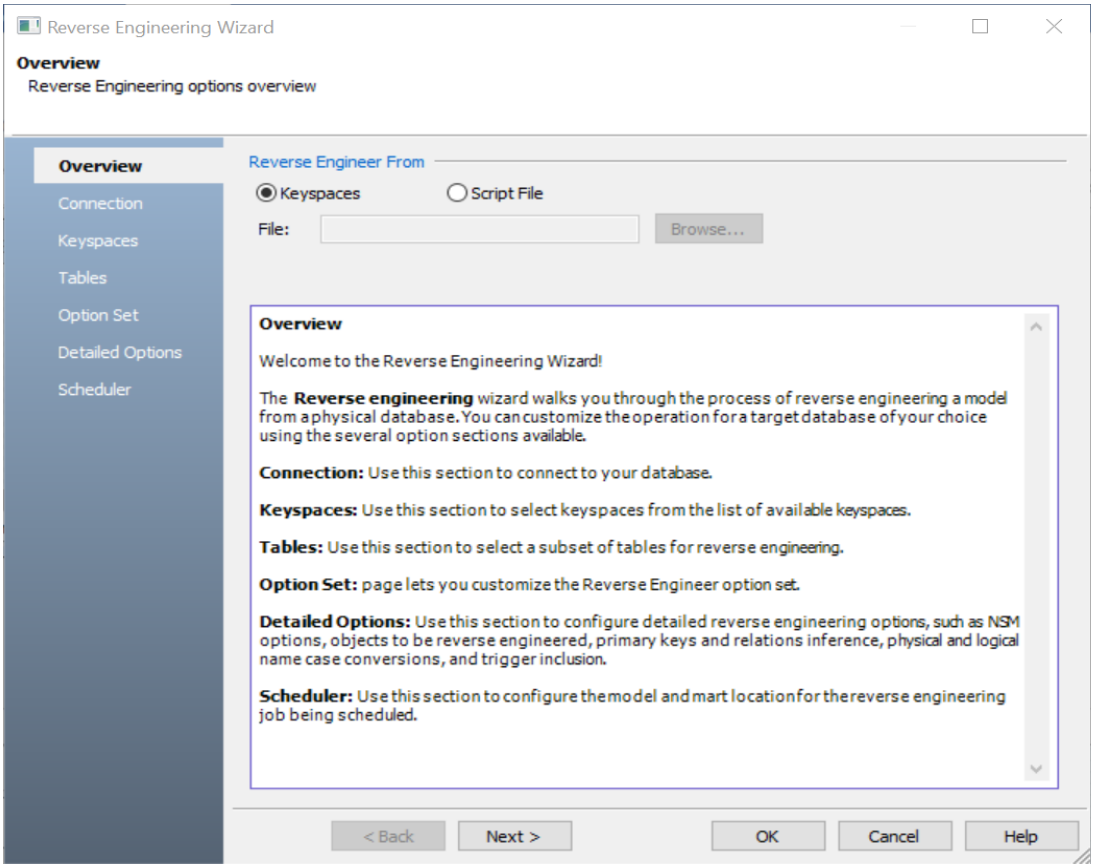
Version:

Template: <Default>

☐ Preserve the template binding

5. Ensure that the Database is set to the correct one. In this case, Amazon Keyspaces. Then, click **Next**.

   The Reverse Engineering Wizard appears.



Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Keyspaces: Use this section to select keyspaces from the list of available keyspaces.

Tables: Use this section to select a subset of tables for reverse engineering.

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

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

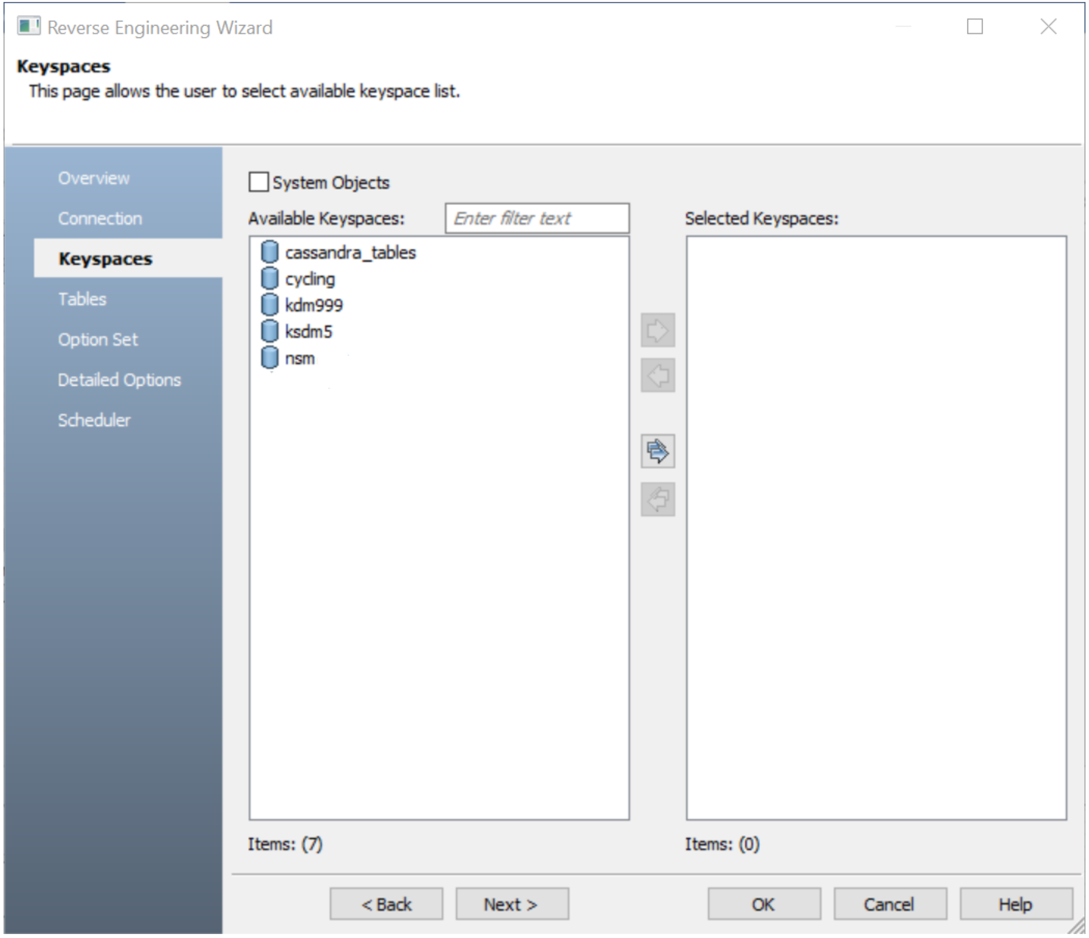
Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

6. Click **Keyspaces**. Then, click **Next**.

   The Connection section appears. Use this section to connect to the database from which you want to reverse engineer the model.

7. After connection is established, click **Next**.

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



Reverse Engineering Wizard

Keyspaces

This page allows the user to select available keyspace list.

□ System Objects

Available Keyspaces: Enter filter text

- cassandra_tables
- cycling
- kdm999
- ksdm5
- nsm

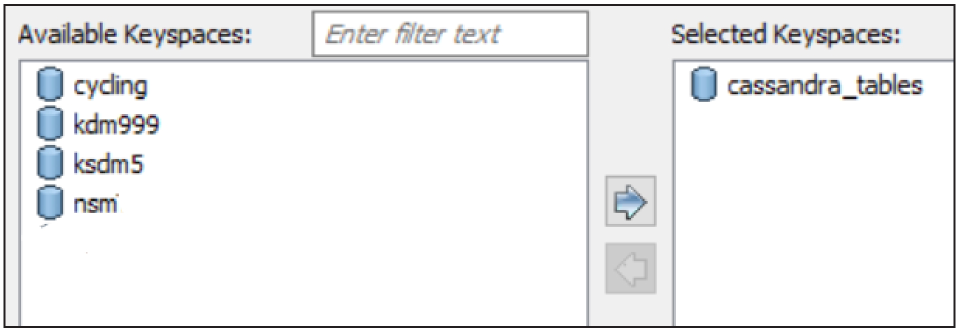
Items: (7)

Selected Keyspaces:

Items: (0)

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

   This moves the selected keyspaces under Selected Keyspaces.



Available Keyspaces: Enter filter text

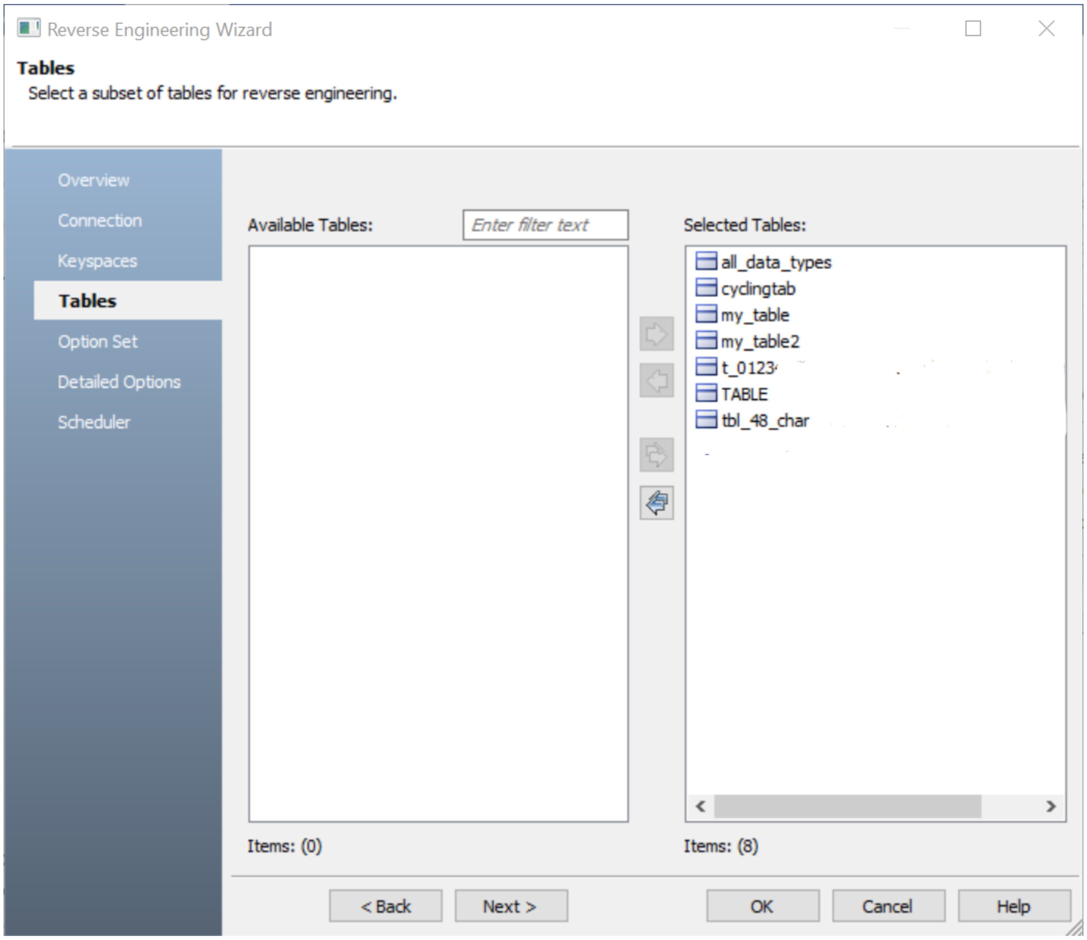
- cycling
- kdm999
- ksdm5
- nsm

Selected Keyspaces:

- cassandra_tables

9. Click **Next** and in the Tables section, click ⇒.

   This selects all the available tables.



Reverse Engineering Wizard

Tables

Select a subset of tables for reverse engineering.

Available Tables: Enter filter text

Selected Tables:

- all_data_types
- cyclingtab
- my_table
- my_table2
- t_0123
- TABLE
- tbl_48_char

Items: (0)

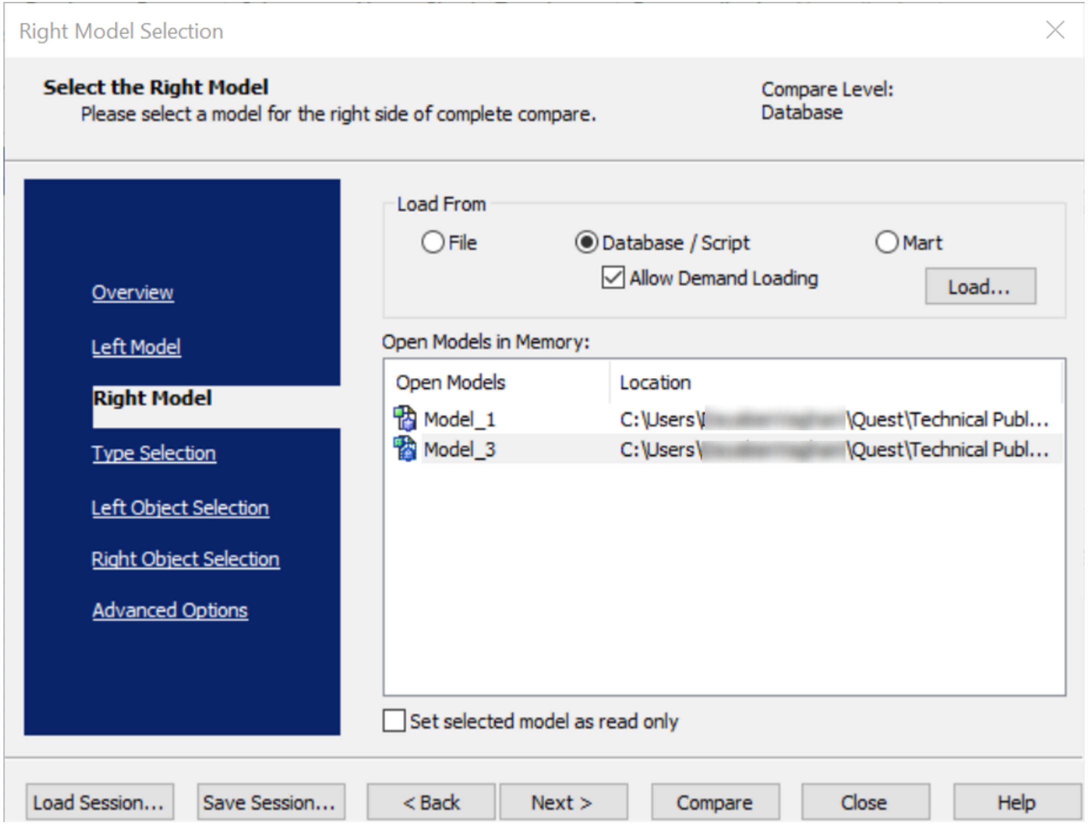
Items: (8)

10. Click **Next** and in the Option Set section, keep the default configuration.

11. Click **Next** and in the Detail Options section, keep the default configuration.

12. Click **OK**.

    The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse engineered.



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

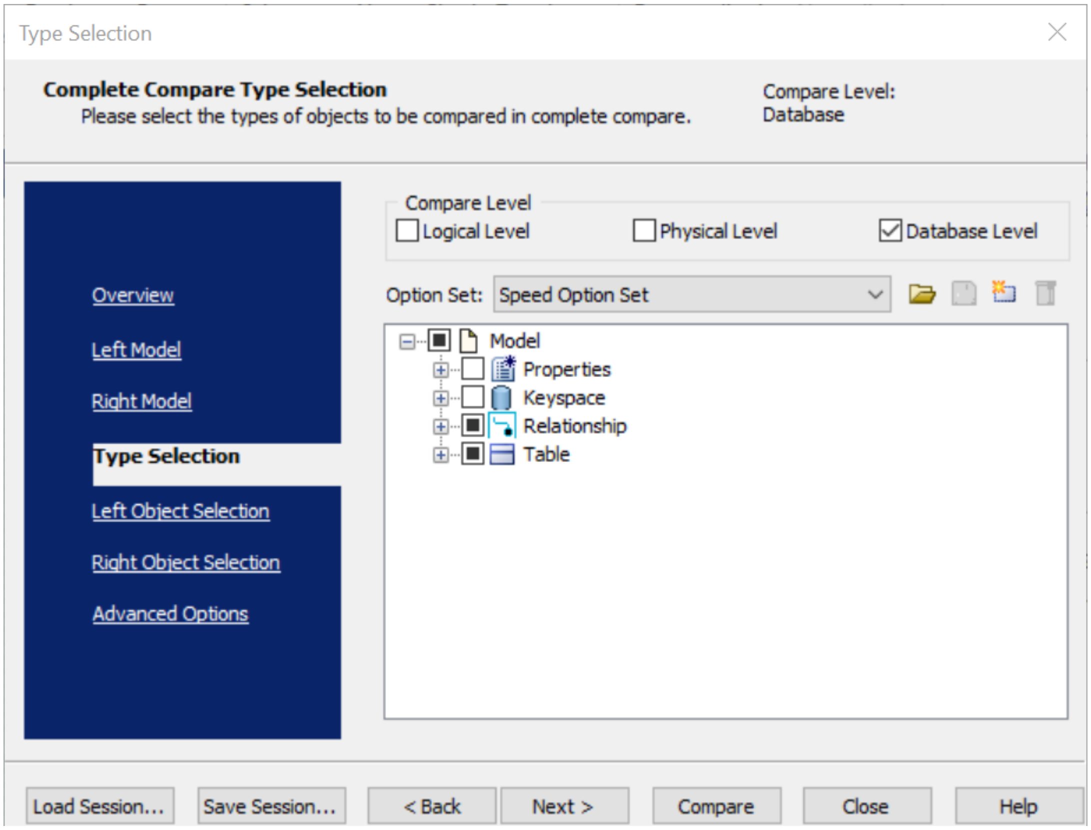
Open Models in Memory:

Open Models	Location
Model_1	C:\Users\[redacted]\Quest\Technical Publ...
Model_3	C:\Users\[redacted]\Quest\Technical Publ...

□ Set selected model as read only

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

For example, the following image shows the default options.



Type Selection

Complete Compare Type Selection

Please select the types of objects to be compared in complete compare.

Compare Level:
Database

Compare Level

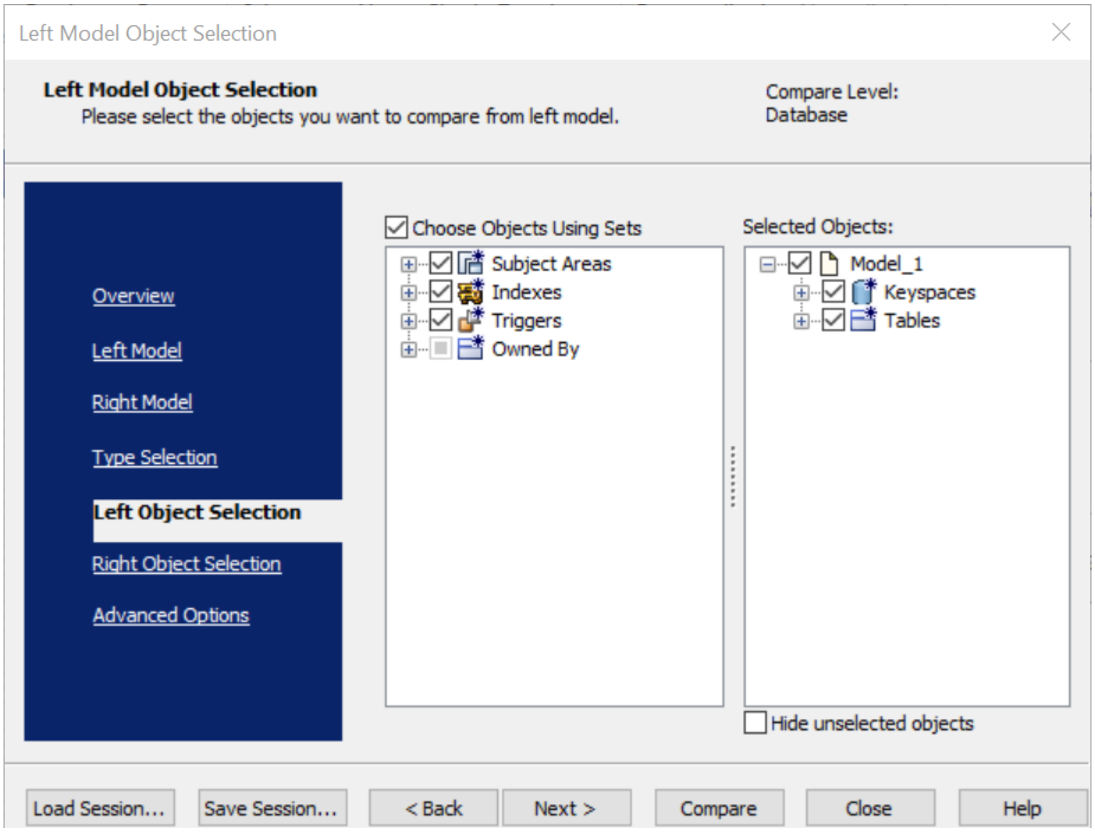
□ Logical Level □ Physical Level ☑ Database Level

Option Set: Speed Option Set

- ■ Model
  - □ Properties
  - □ Keyspace
  - ■ Relationship
  - ■ Table

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

For example, the following image shows the default options.



Left Model Object Selection

Left Model Object Selection

Please select the objects you want to compare from left model.

Compare Level:
Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

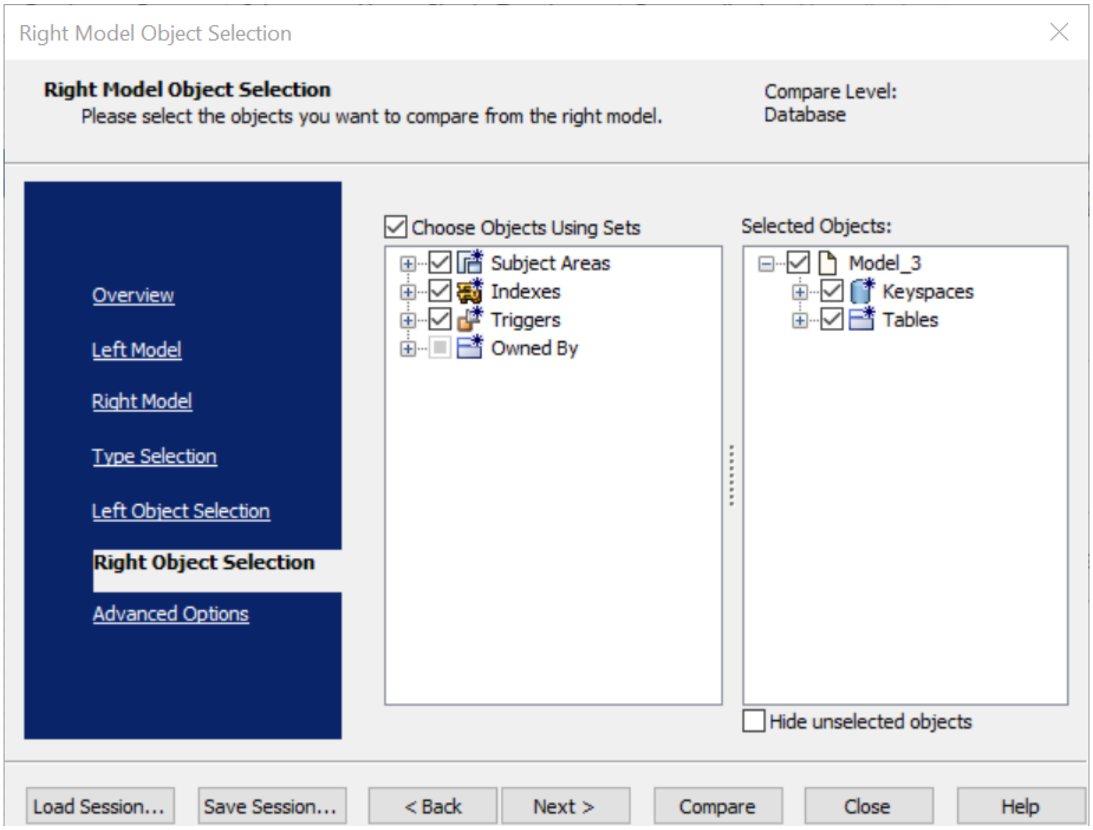
Selected Objects:

- ☑ Model_1
  - ☑ Keyspaces
  - ☑ Tables

□ Hide unselected objects

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

For example, the following image shows the default options.



Right Model Object Selection

Right Model Object Selection
Please select the objects you want to compare from the right model.

Compare Level:
Database

☑ Choose Objects Using Sets

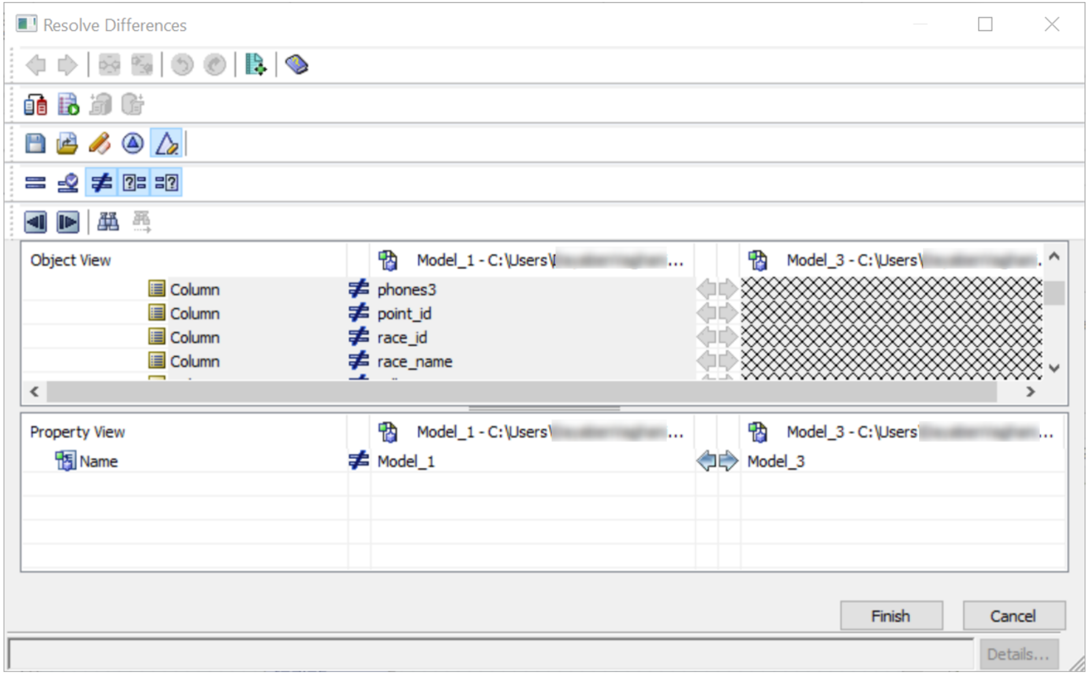
- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

Selected Objects:

- ☑ Model_3
  - ☑ Keyspaces
  - ☑ Tables

□ Hide unselected objects

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.



Resolve Differences

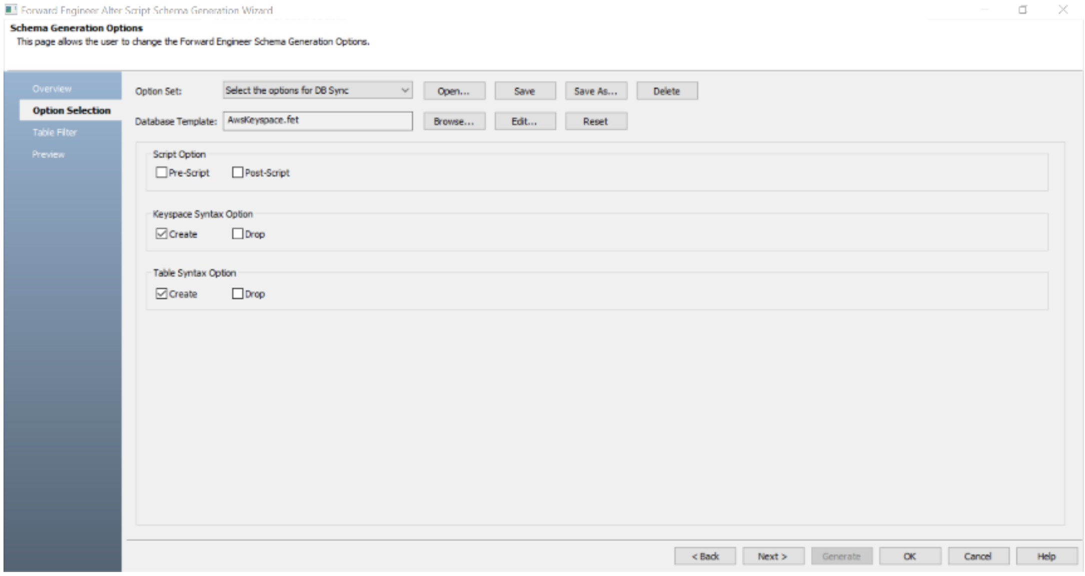
Object View	Model_1 - C:\Users\...	Model_3 - C:\Users\...
Column	≠ phones3	
Column	≠ point_id	
Column	≠ race_id	
Column	≠ race_name	

Property View	Model_1 - C:\Users\...	Model_3 - C:\Users\...
Name	≠ Model_1	Model_3

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 launches the Forward Engineering Alter Script Generation Wizard.

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



Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Select the options for DB Sync

Database Template: AwsKeyspace.fet

Script Option

□ Pre-Script □ Post-Script

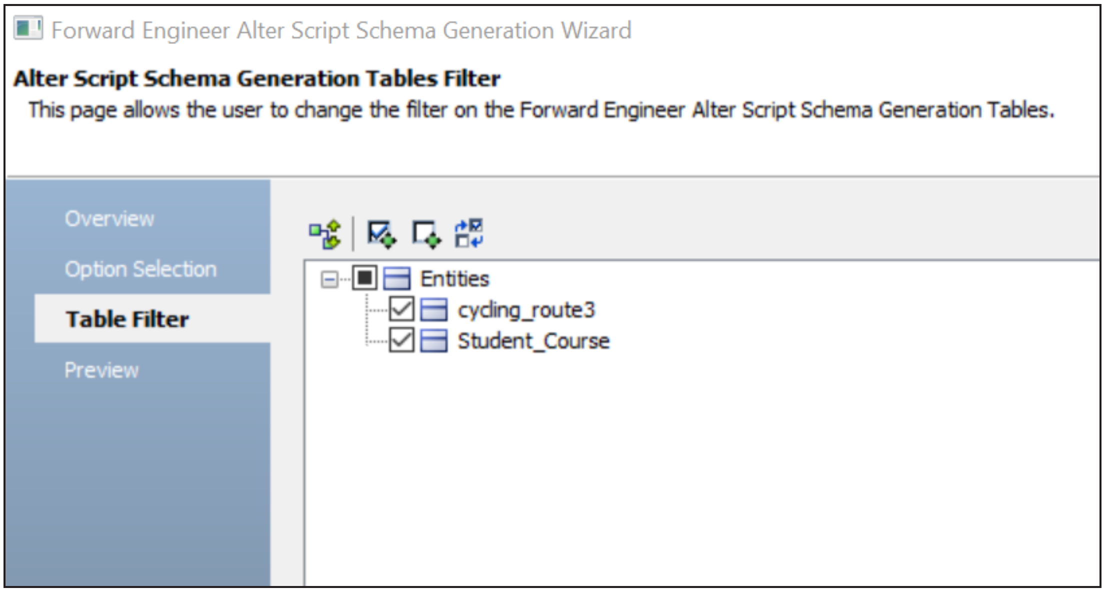
Keyspace Syntax Option

☑ Create □ Drop

Table Syntax Option

☑ Create □ Drop

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



Alter Script Schema Generation Tables Filter

This page allows the user to change the filter on the Forward Engineer Alter Script Schema Generation Tables.

■ Entities

☑ cycling_route3

☑ Student_Course

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.

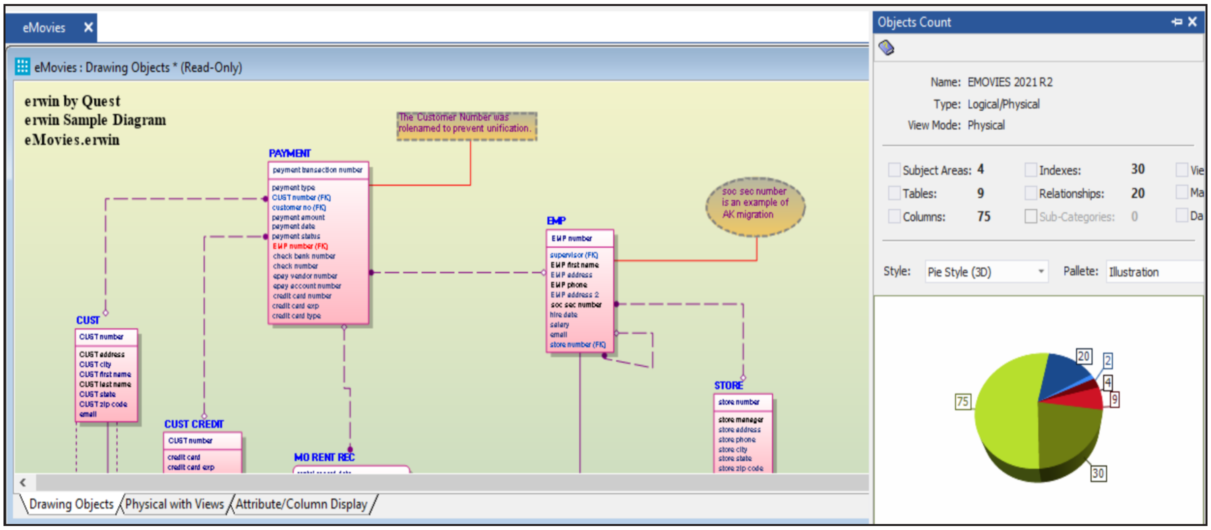
Migration by Changing the Target Database

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.

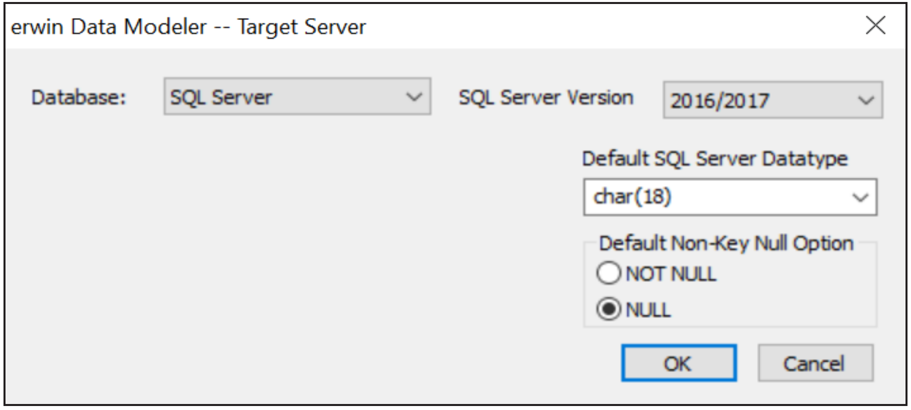


Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 4 □ Indexes: 30
□ Tables: 9 □ Relationships: 20
□ Columns: 75 □ Sub-Categories: 0

Style: Pie Style (3D) Pallete: Illustration

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.



erwin Data Modeler -- Target Server

Database: SQL Server

SQL Server Version 2016/2017

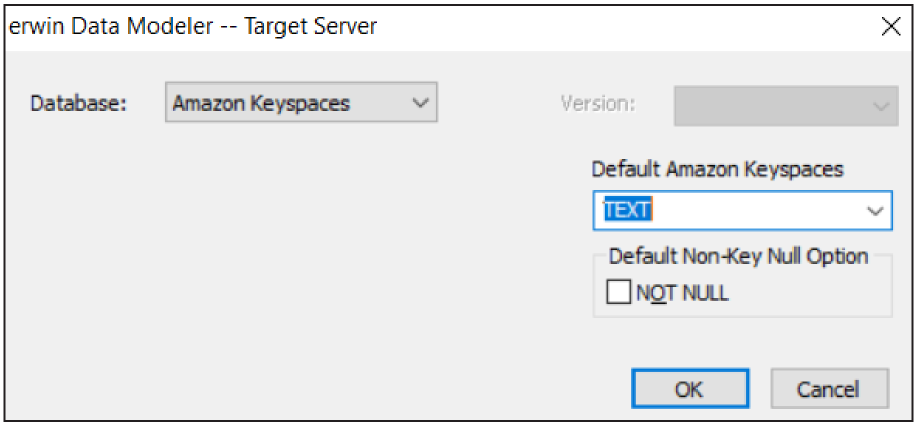
Default SQL Server Datatype: char(18)

Default Non-Key Null Option

○ NOT NULL

● NULL

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



erwin Data Modeler -- Target Server

Database: Amazon Keyspaces

Version:

Default Amazon Keyspaces: TEXT

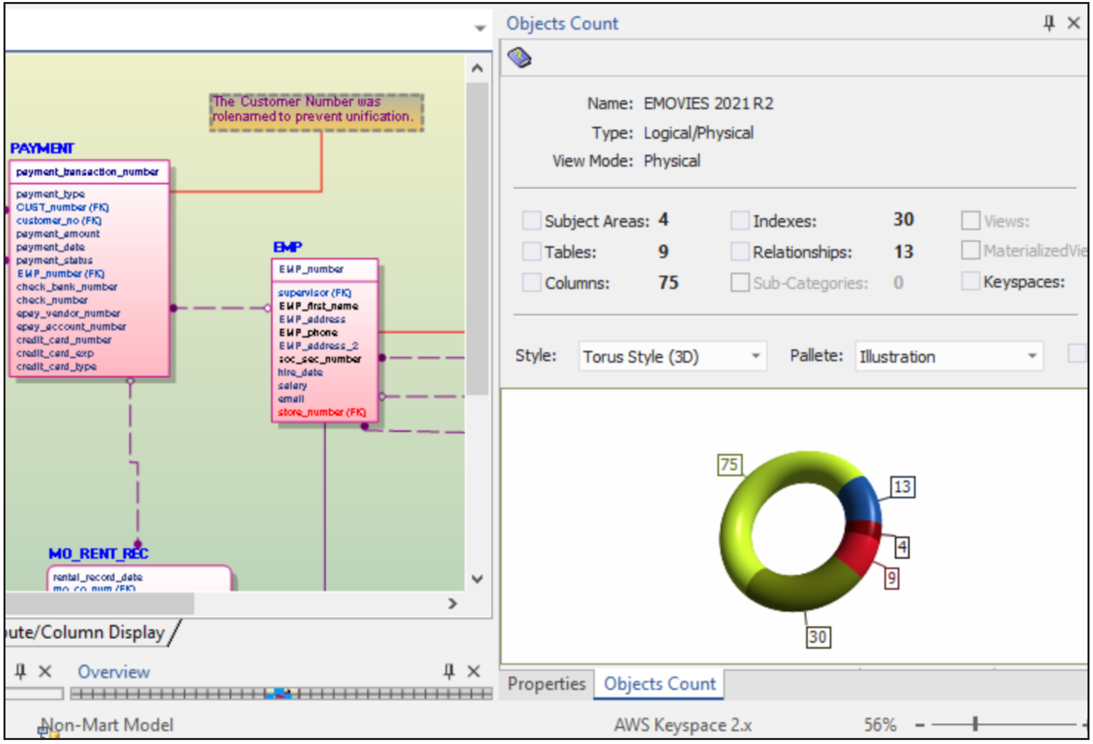
Default Non-Key Null Option

□ NOT NULL

4. Click **OK**.

   The conversion process starts.

   Once the conversion is complete, the existing model is migrated to an Amazon Keyspaces database.



Objects Count

Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 4	□ Indexes:	30	□ Views:	
□ Tables: 9	□ Relationships:	13	□ MaterializedVie	
□ Columns: 75	□ Sub-Categories:	0	□ Keyspaces:	

Style: Torus Style (3D) Pallete: Illustration

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.

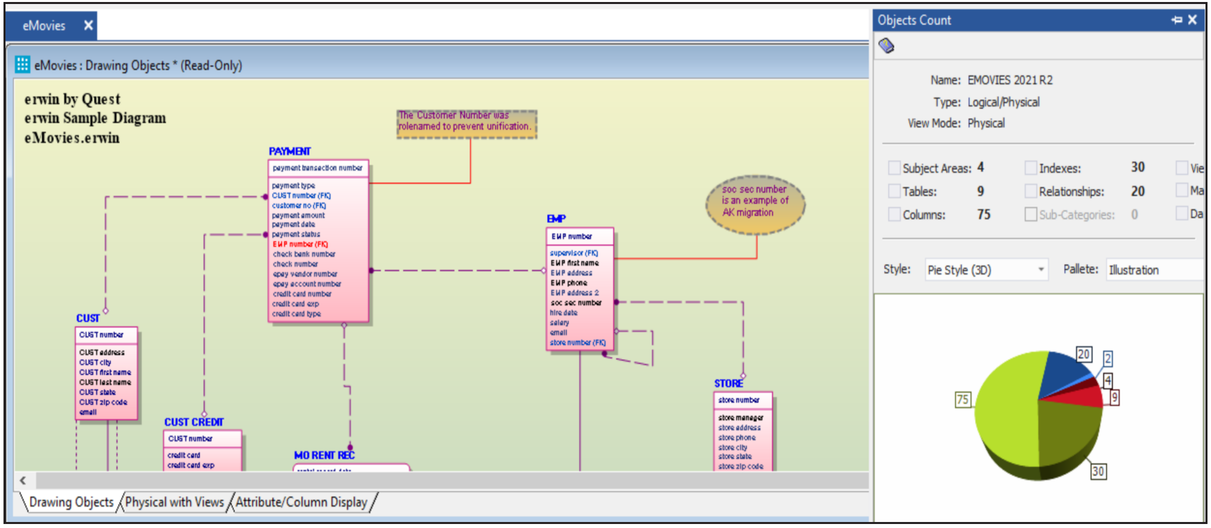
Migration by Deriving a Model

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.

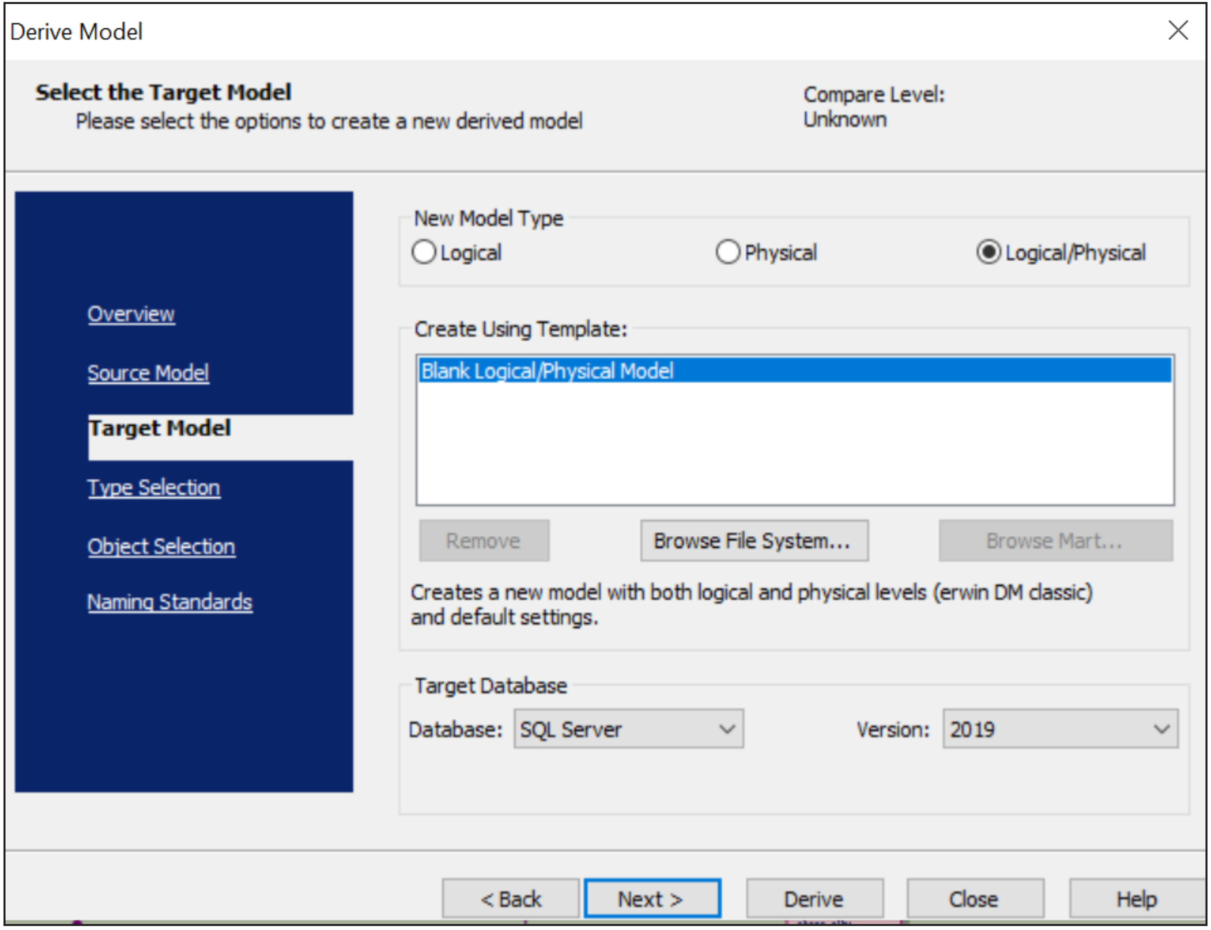


Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	4	□ Indexes:	30	□ Vie
□ Tables:	9	□ Relationships:	20	□ Ma
□ Columns:	75	□ Sub-Categories:	0	□ Da

Style: Pie Style (3D) Pallete: Illustration

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

New Model Type
○ Logical ○ Physical ● Logical/Physical

Create Using Template:
Blank Logical/Physical Model

Creates a new model with both logical and physical levels (erwin DM classic) and default settings.

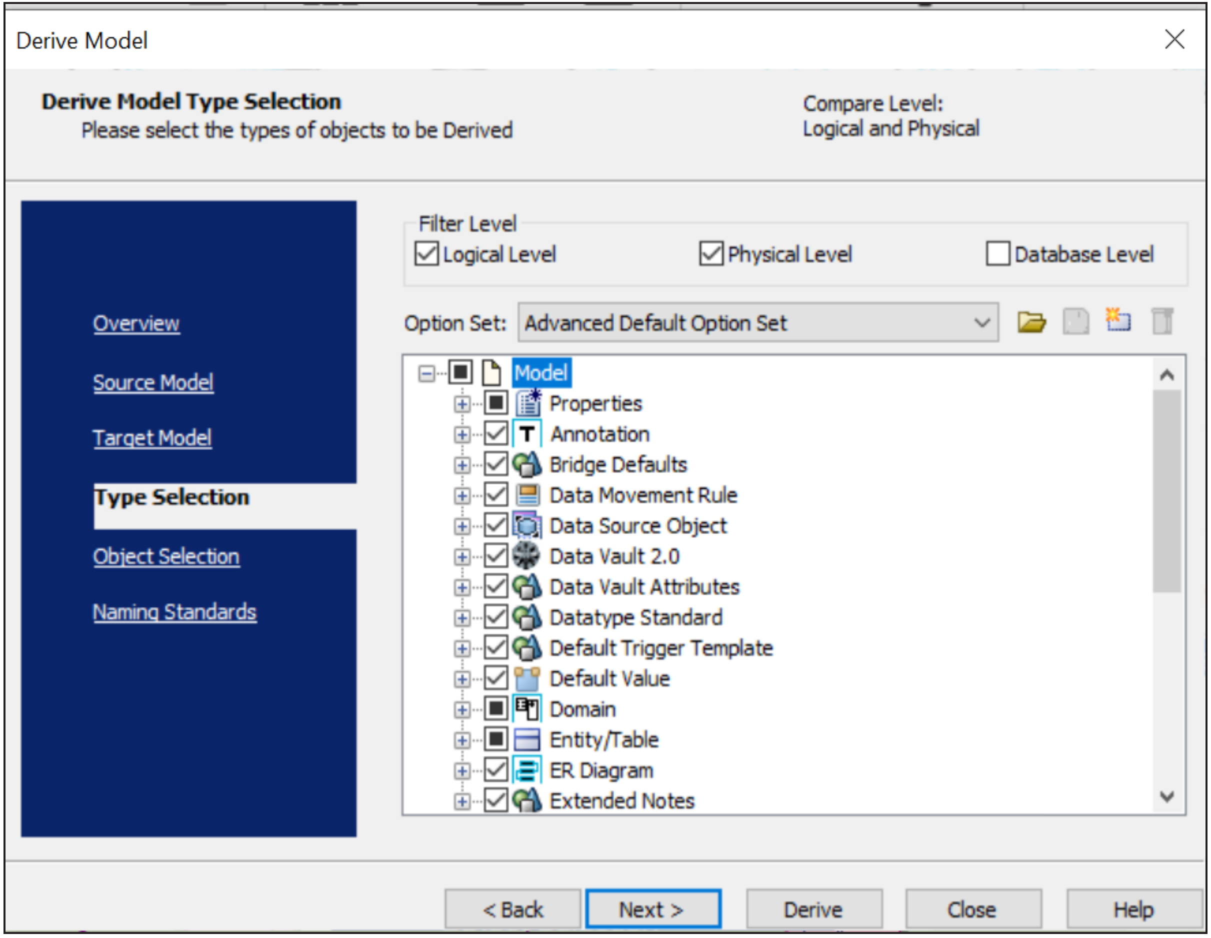
Target Database
Database: SQL Server
Version: 2019

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

4. Click **Next**.

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

The Type Selection section appears.



Derive Model Type Selection
Please select the types of objects to be Derived

Compare Level:
Logical and Physical

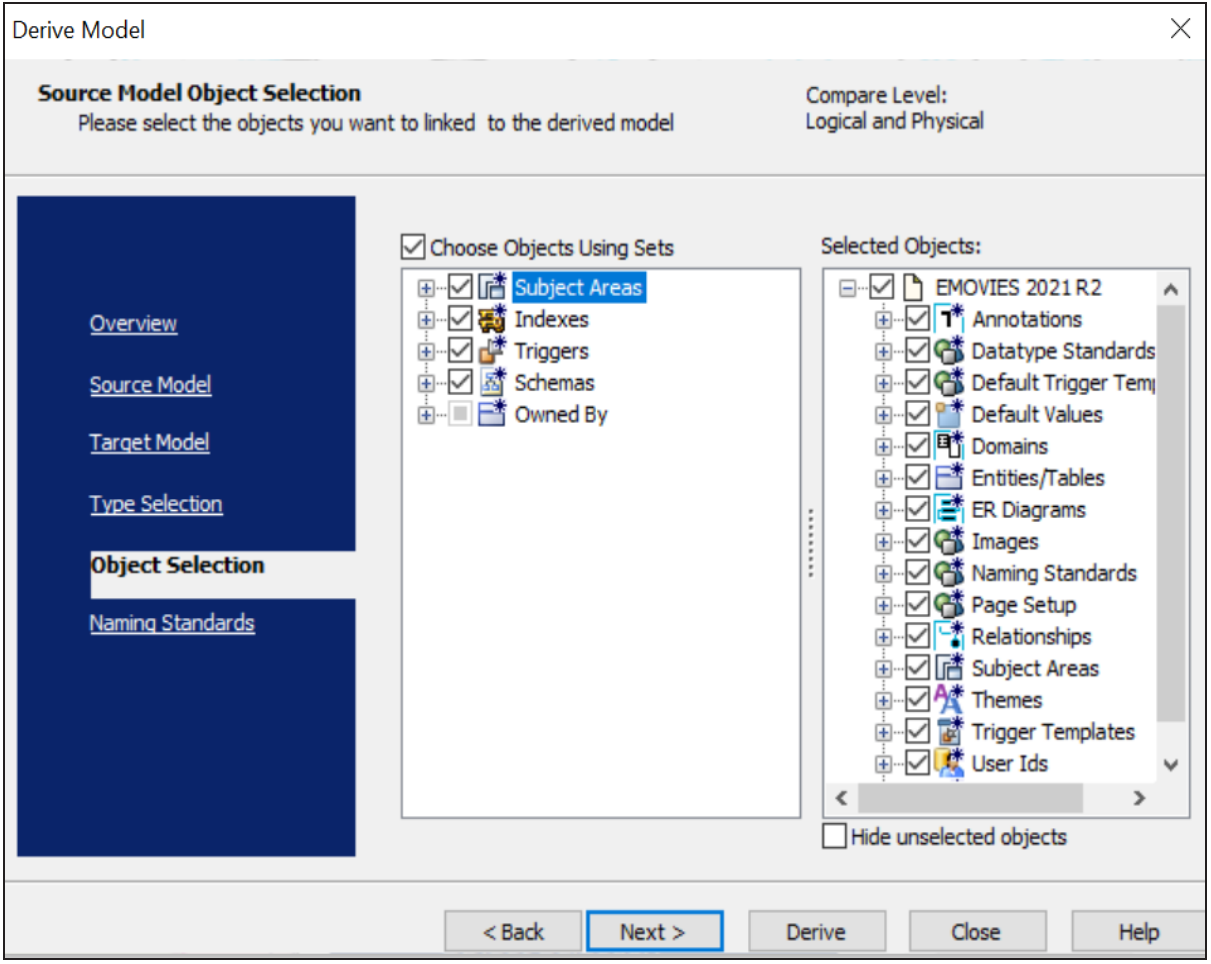
Filter Level
☑ Logical Level ☑ Physical Level □ Database Level

Option Set: Advanced Default Option Set

- ■ Model
  - ■ Properties
  - ☑ Annotation
  - ☑ Bridge Defaults
  - ☑ Data Movement Rule
  - ☑ Data Source Object
  - ☑ Data Vault 2.0
  - ☑ Data Vault Attributes
  - ☑ Datatype Standard
  - ☑ Default Trigger Template
  - ☑ Default Value
  - ■ Domain
  - ■ Entity/Table
  - ☑ ER Diagram
  - ☑ Extended Notes

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


Derive Model

Source Model Object Selection

Please select the objects you want to linked to the derived model

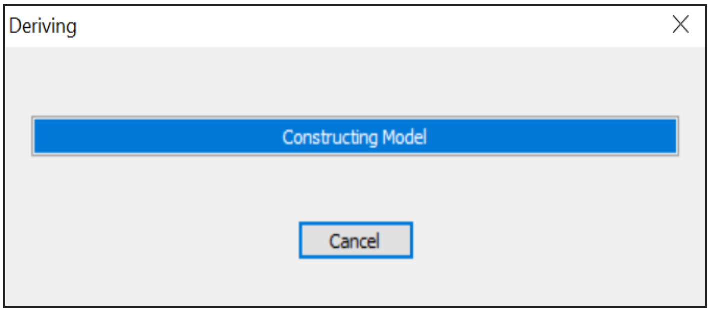
Compare Level:
Logical and Physical

☑ Choose Objects Using Sets

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

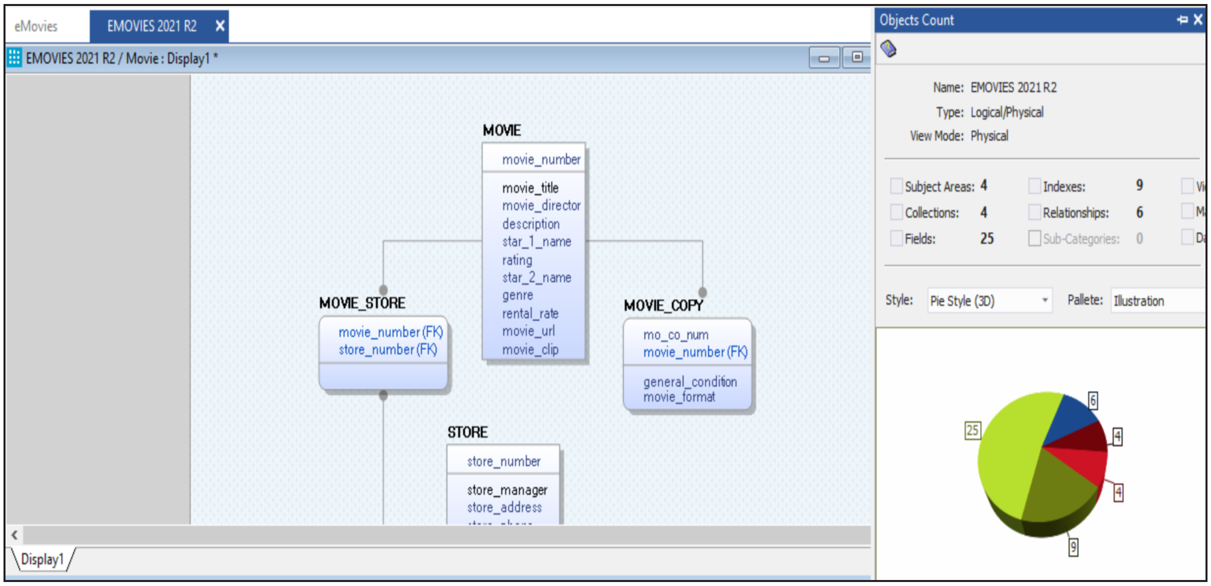
8. Click **Derive**.

   The model derivation process starts.


Deriving

Constructing Model

Once the conversion is complete, the existing model is migrated to an Amazon Keyspaces database.



Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	4	□ Indexes:	9
□ Collections:	4	□ Relationships:	6
□ Fields:	25	□ Sub-Categories:	0

Style: Pie Style (3D) Pallete: Illustration

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

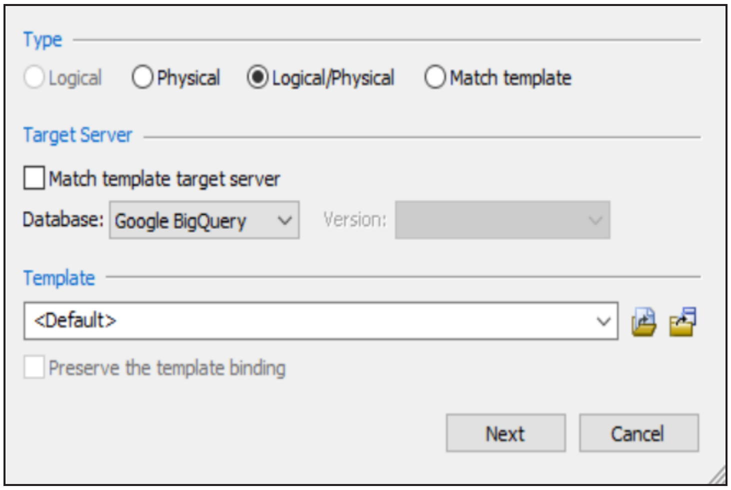
Reverse Engineering Models

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.



Type

○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

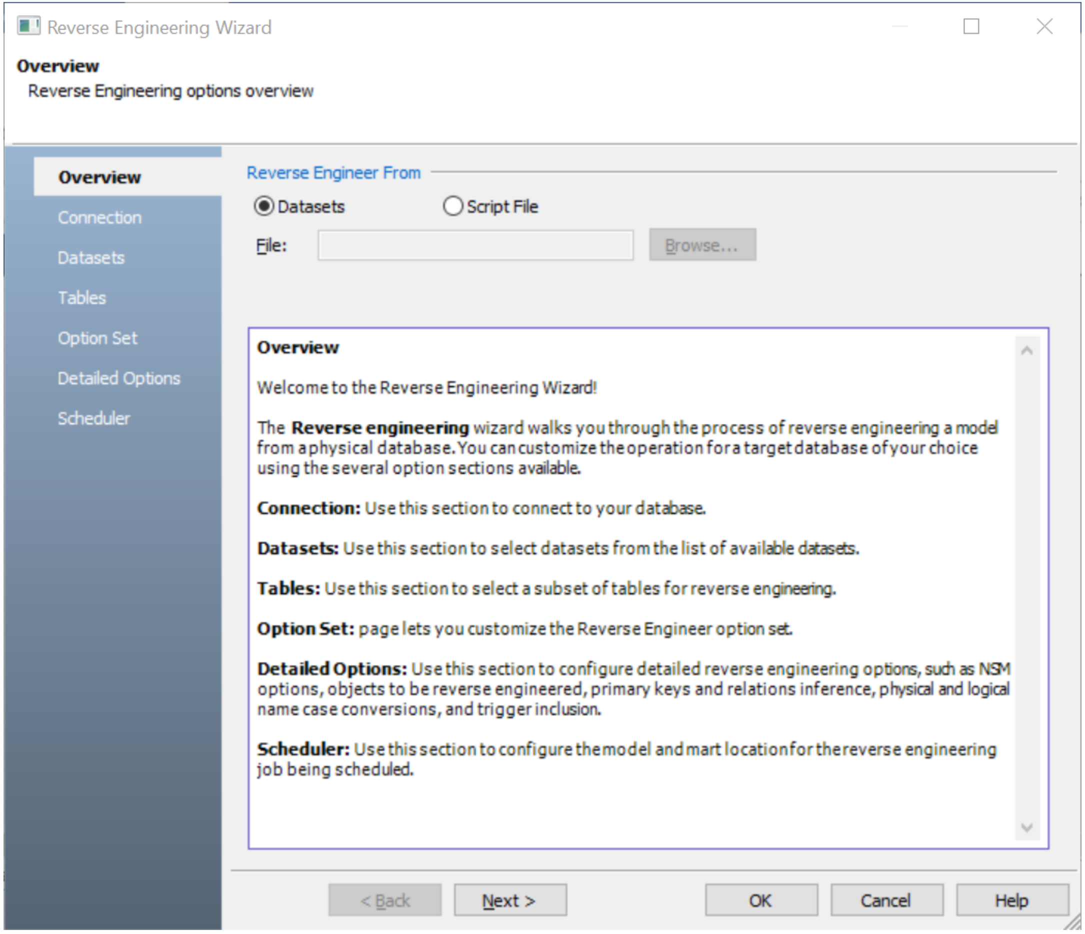
Database: Google BigQuery Version:

Template

<Default>

□ Preserve the template binding

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



Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Datasets: Use this section to select datasets from the list of available datasets.

Tables: Use this section to select a subset of tables for reverse engineering.

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

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

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

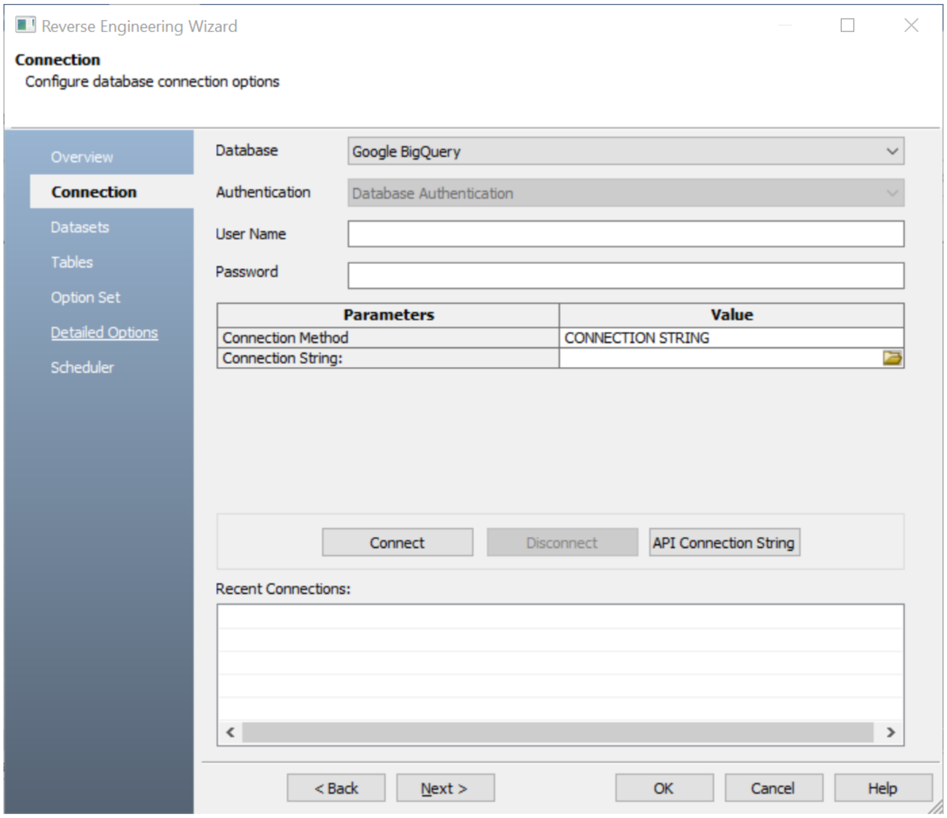
     > If you click **Datasets**, continue to step 5.

   - **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 section appears.



Database	Google BigQuery
Authentication	Database Authentication
User Name	
Password	

Parameters	Value
Connection Method	CONNECTION STRING
Connection String:	

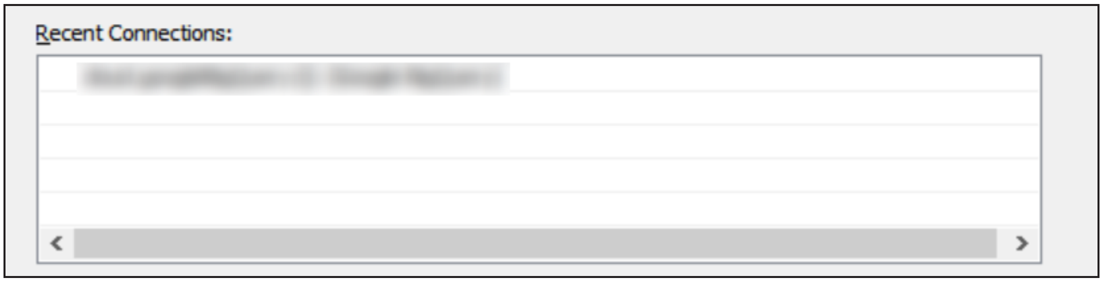
6. Enter your **User Name** and **Password**.

   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

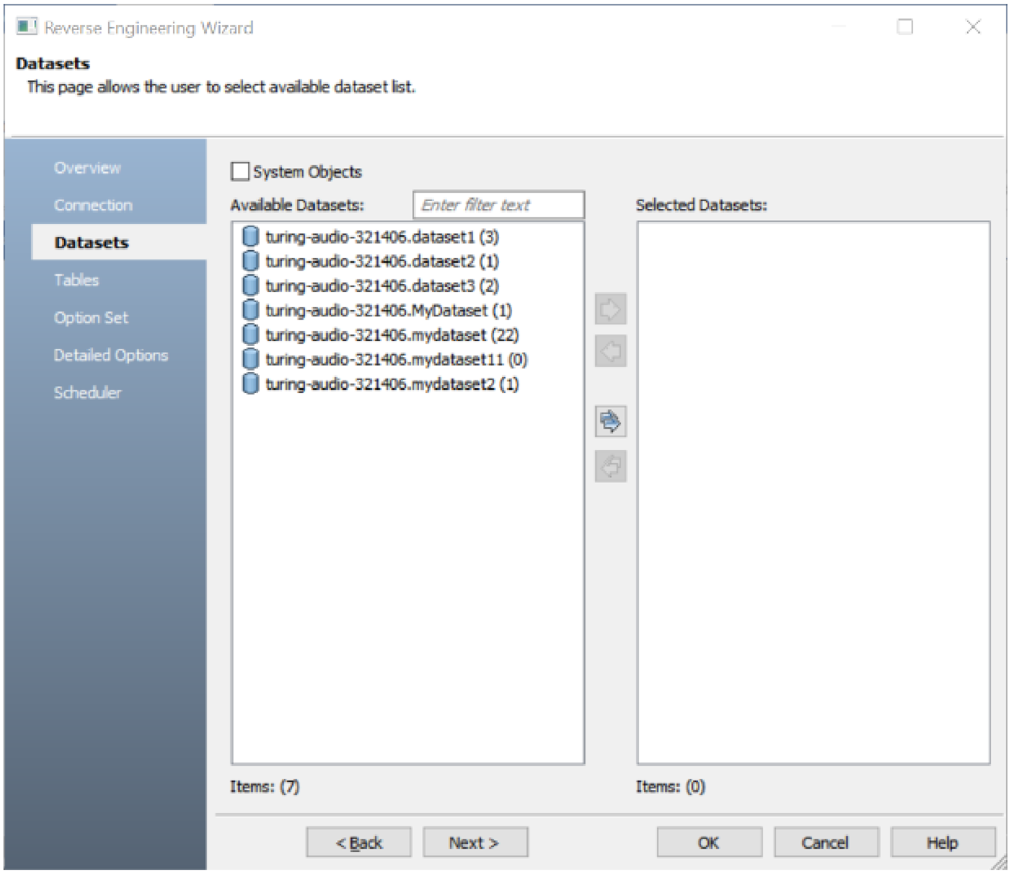
7. Click **Connect**.
   On successful connection, your connection information is displayed under Recent

Connections.



8. Click **Next**.

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



Datasets

This page allows the user to select available dataset list.

□ System Objects

Available Datasets: Enter filter text

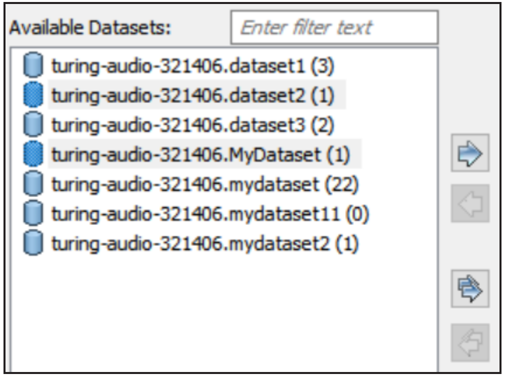
- turing-audio-321406.dataset1 (3)
- turing-audio-321406.dataset2 (1)
- turing-audio-321406.dataset3 (2)
- turing-audio-321406.MyDataset (1)
- turing-audio-321406.mydataset (22)
- turing-audio-321406.mydataset11 (0)
- turing-audio-321406.mydataset2 (1)

Items: (7)

Selected Datasets:

Items: (0)

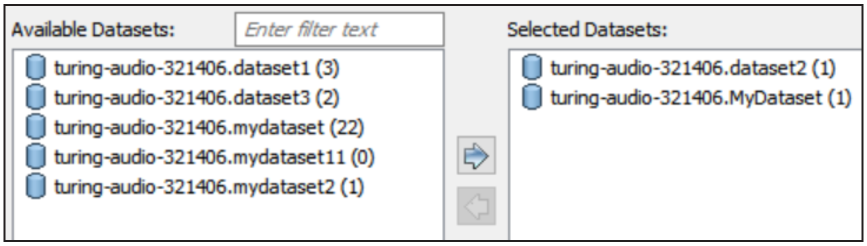
9. Under **Available Datasets**, select the datasets that you want to reverse engineer. Then, click ⇨.



Available Datasets: Enter filter text

turing-audio-321406.dataset1 (3)
turing-audio-321406.dataset2 (1)
turing-audio-321406.dataset3 (2)
turing-audio-321406.MyDataset (1)
turing-audio-321406.mydataset (22)
turing-audio-321406.mydataset11 (0)
turing-audio-321406.mydataset2 (1)

This moves the selected datasets under Selected Datasets.



Available Datasets: Enter filter text

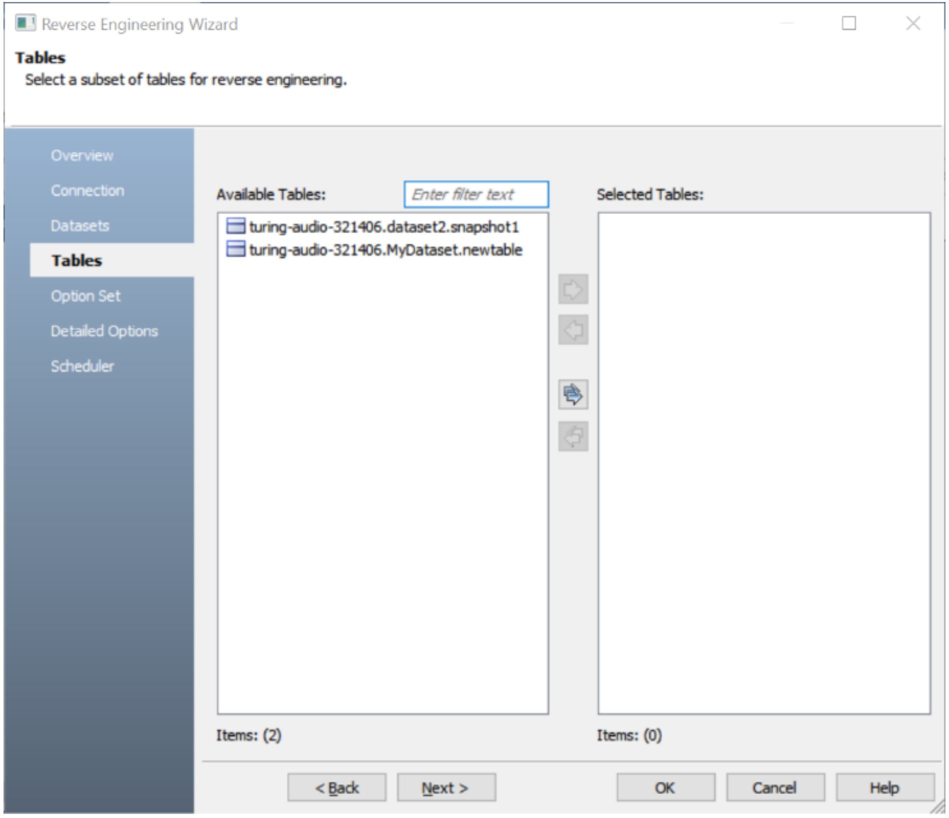
turing-audio-321406.dataset1 (3)
turing-audio-321406.dataset3 (2)
turing-audio-321406.mydataset (22)
turing-audio-321406.mydataset11 (0)
turing-audio-321406.mydataset2 (1)

Selected Datasets:

turing-audio-321406.dataset2 (1)
turing-audio-321406.MyDataset (1)

10. Click **Next**.

    The Tables section appears. It displays a list of available tables in the datasets that you selected in step 8.



Tables

Select a subset of tables for reverse engineering.

Available Tables:

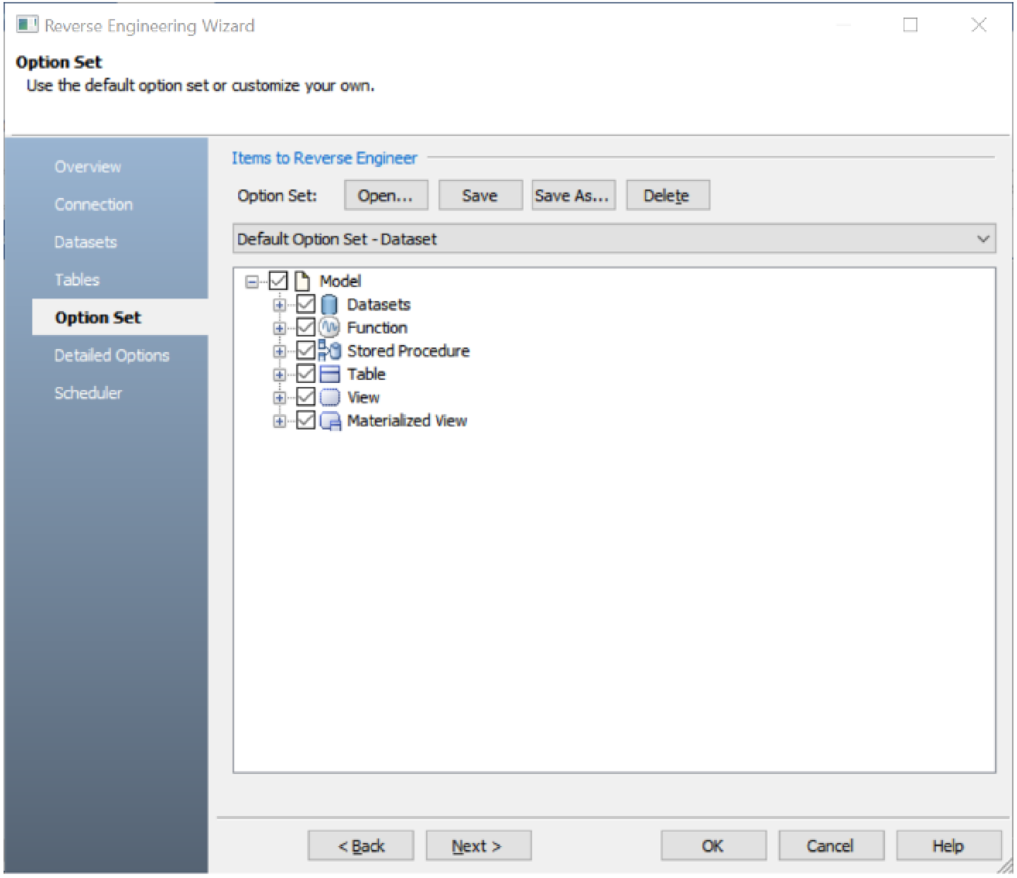
- turing-audio-321406.dataset2.snapshot1
- turing-audio-321406.MyDataset.newtable

Items: (2)

Selected Tables:

Items: (0)

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



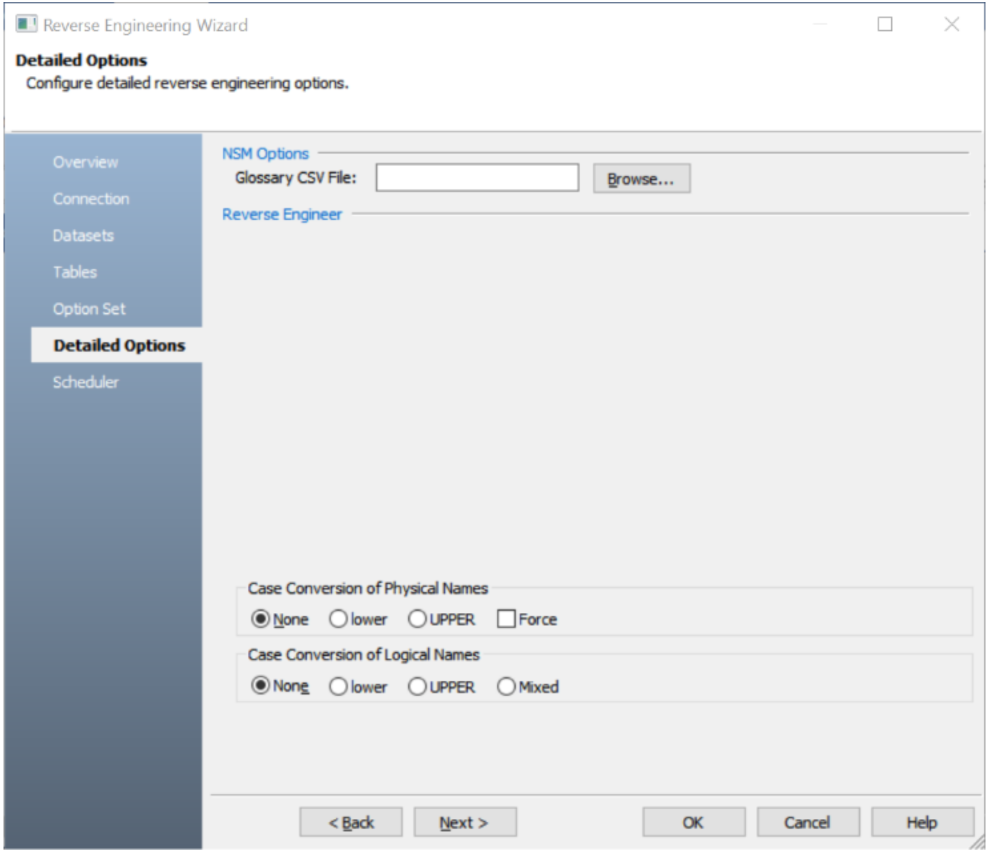
Reverse Engineering Wizard

Option Set

Use the default option set or customize your own.

12. Click **Next**.

    The Detail Options section appears. Set up appropriate options based on your requirement.



Reverse Engineering Wizard

Detailed Options

Configure detailed reverse engineering options.

NSM Options

Glossary CSV File:

Reverse Engineer

Case Conversion of Physical Names

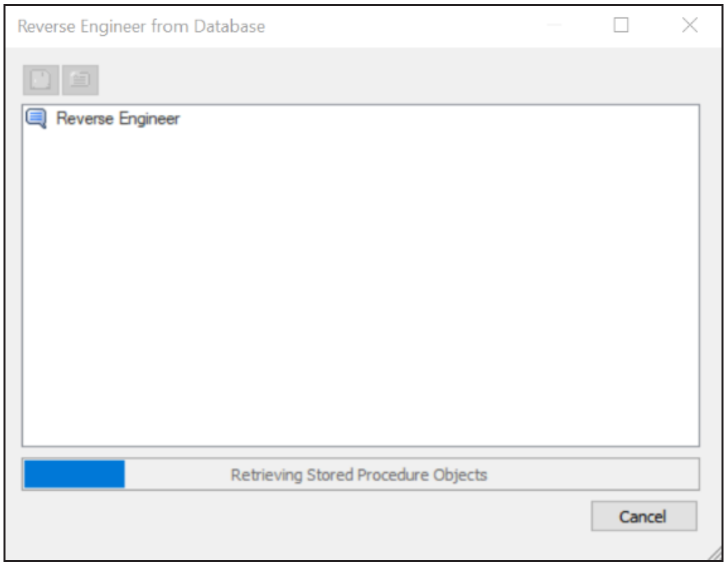
● None ○ lower ○ UPPER □ Force

Case Conversion of Logical Names

● None ○ lower ○ UPPER ○ Mixed

13. Click **OK**.

The reverse engineering process starts.

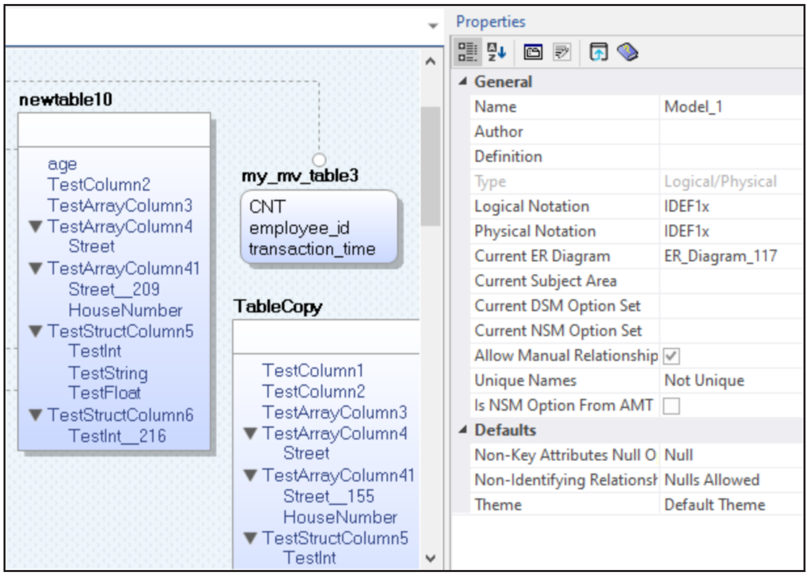


Reverse Engineer from Database

Reverse Engineer

Retrieving Stored Procedure Objects

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



Properties

General	
Name	Model_1
Author	
Definition	
Type	Logical/Physical
Logical Notation	IDEF1x
Physical Notation	IDEF1x
Current ER Diagram	ER_Diagram_117
Current Subject Area	
Current DSM Option Set	
Current NSM Option Set	
Allow Manual Relationship	☑
Unique Names	Not Unique
Is NSM Option From AMT	□
Defaults	
Non-Key Attributes Null O	Null
Non-Identifying Relationsh	Nulls Allowed
Theme	Default Theme

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

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 where the reverse engineered model should be saved and its name	When you schedule a job on a remote server, ensure the model path is same for remote and local server. For example: C:\Scheduler\<Model Name>.erwin
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	Specifies whether the Com-	

Compare	plete 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.

Forward Engineering Models

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

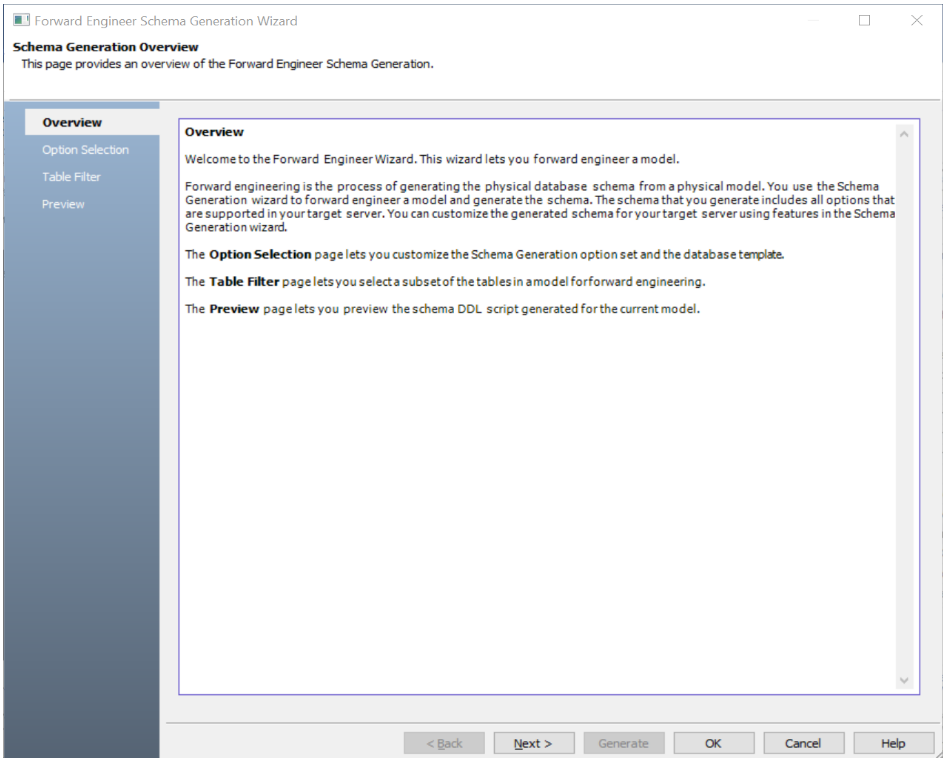
This topic walks you through the steps to forward engineer a Google BigQuery model. For detailed description of forward engineering options, refer to the Forward Engineering Options topic.

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.



Schema Generation Overview

This page provides an overview of the Forward Engineer Schema Generation.

Overview

Welcome to the Forward Engineer Wizard. This wizard lets you forward engineer a model.

Forward engineering is the process of generating the physical database schema from a physical model. You use the Schema Generation wizard to forward engineer a model and generate the schema. The schema that you generate includes all options that are supported in your target server. You can customize the generated schema for your target server using features in the Schema Generation wizard.

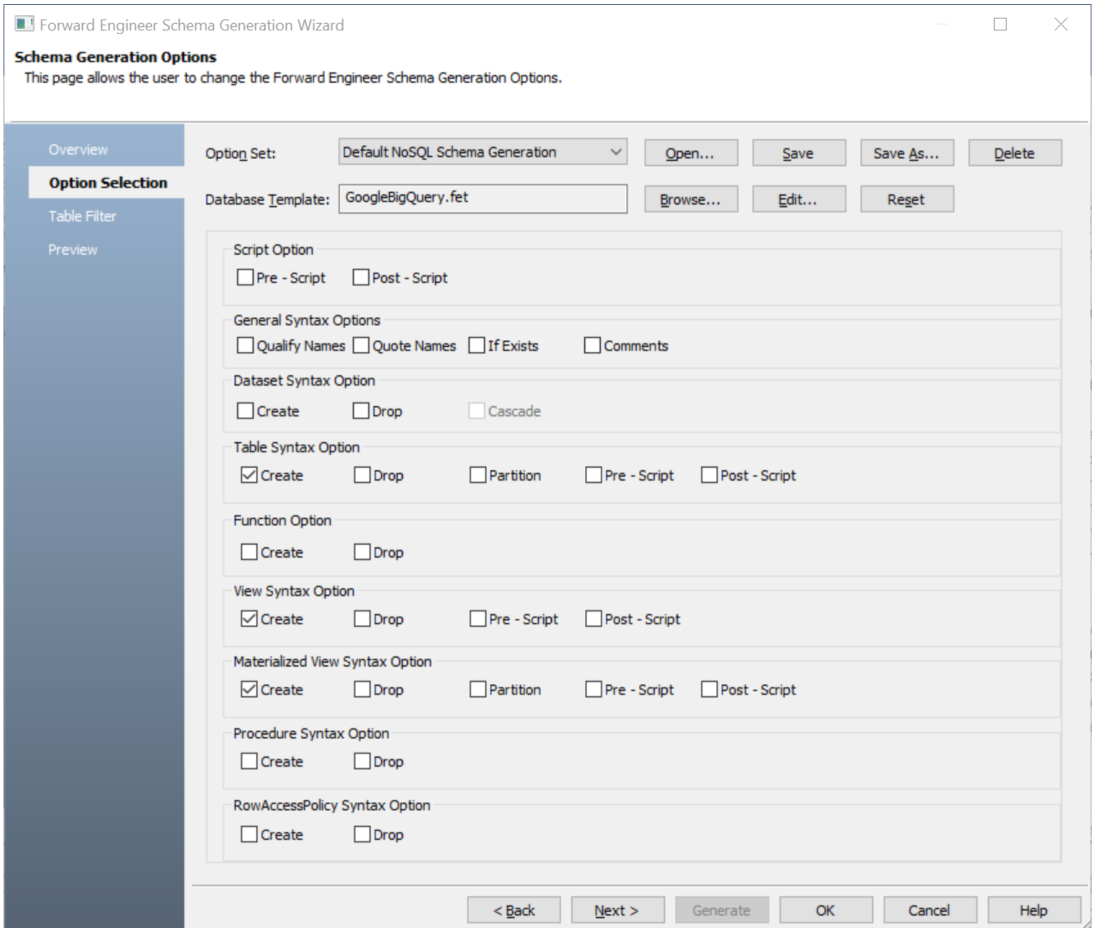
The **Option Selection** page lets you customize the Schema Generation option set and the database template.

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

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

3. Click **Option Selection**.

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



Forward Engineer Schema Generation Wizard

Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default NoSQL Schema Generation

Database Template: GoogleBigQuery.fet

Script Option

□ Pre - Script □ Post - Script

General Syntax Options

□ Qualify Names □ Quote Names □ If Exists □ Comments

Dataset Syntax Option

□ Create □ Drop □ Cascade

Table Syntax Option

☑ Create □ Drop □ Partition □ Pre - Script □ Post - Script

Function Option

□ Create □ Drop

View Syntax Option

☑ Create □ Drop □ Pre - Script □ Post - Script

Materialized View Syntax Option

☑ Create □ Drop □ Partition □ Pre - Script □ Post - Script

Procedure Syntax Option

□ Create □ Drop

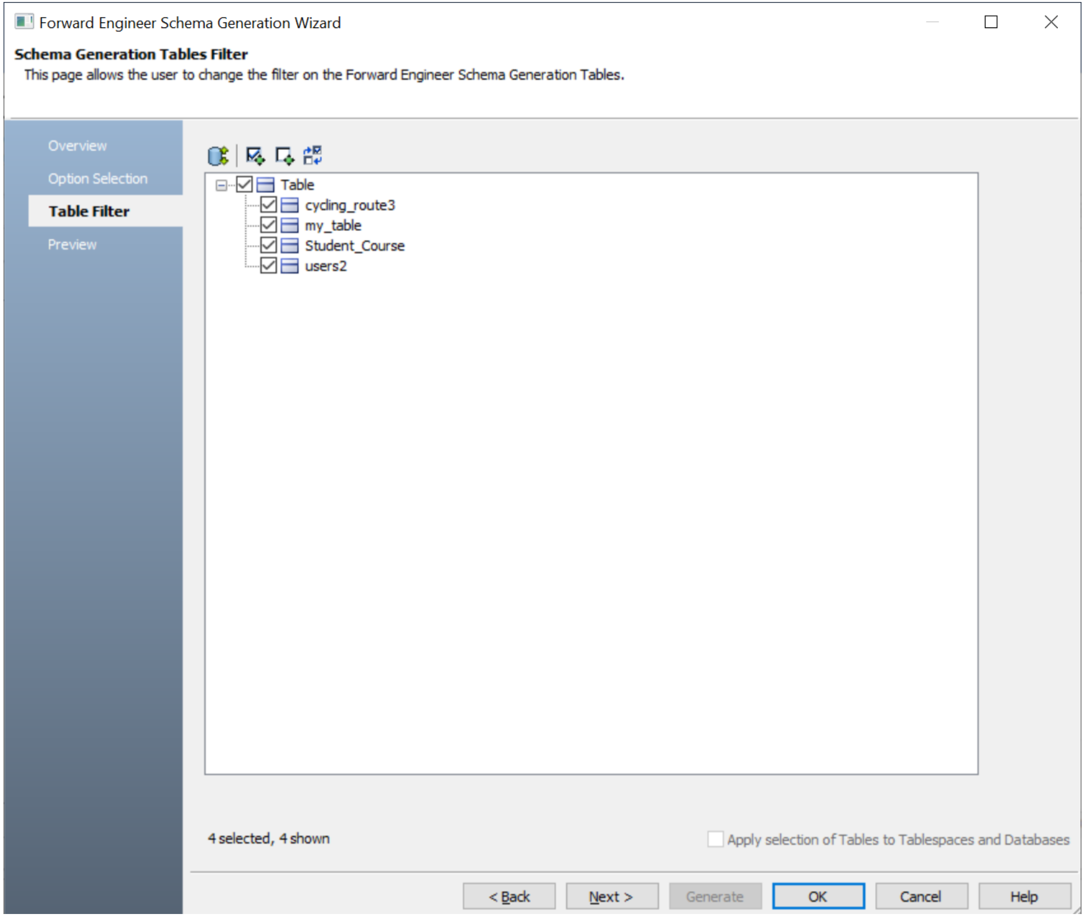
RowAccessPolicy Syntax Option

□ Create □ Drop

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

4. Click **Next**.

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



Forward Engineer Schema Generation Wizard

Schema Generation Tables Filter

This page allows the user to change the filter on the Forward Engineer Schema Generation Tables.

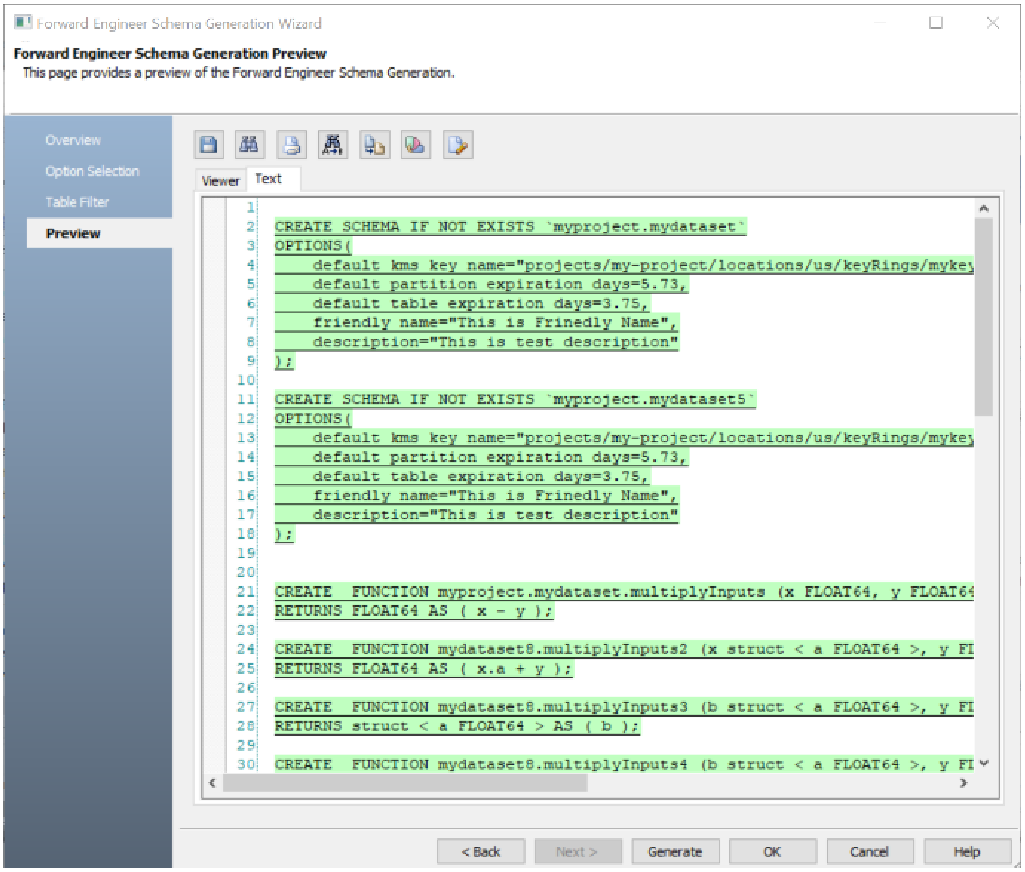
☑ Table
- ☑ cycling_route3
- ☑ my_table
- ☑ Student_Course
- ☑ users2

4 selected, 4 shown

□ Apply selection of Tables to Tablespaces and Databases

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

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



Forward Engineer Schema Generation Preview

This page provides a preview of the Forward Engineer Schema Generation.

```

CREATE SCHEMA IF NOT EXISTS `myproject.mydataset`
OPTIONS(
    default kms key name="projects/my-project/locations/us/keyRings/mykey
    default partition expiration days=5.73,
    default table expiration days=3.75,
    friendly name="This is Frinedly Name",
    description="This is test description"
);

CREATE SCHEMA IF NOT EXISTS `myproject.mydataset5`
OPTIONS(
    default kms key name="projects/my-project/locations/us/keyRings/mykey
    default partition expiration days=5.73,
    default table expiration days=3.75,
    friendly name="This is Frinedly Name",
    description="This is test description"
);


CREATE  FUNCTION myproject.mydataset.multiplyInputs (x FLOAT64, y FLOAT64
RETURNS FLOAT64 AS ( x - y );

CREATE  FUNCTION mydataset8.multiplyInputs2 (x struct < a FLOAT64 >, y FI
RETURNS FLOAT64 AS ( x.a + y );

CREATE  FUNCTION mydataset8.multiplyInputs3 (b struct < a FLOAT64 >, y FI
RETURNS struct < a FLOAT64 > AS ( b );

CREATE  FUNCTION mydataset8.multiplyInputs4 (b struct < a FLOAT64 >, y FI
```

Use the following options:

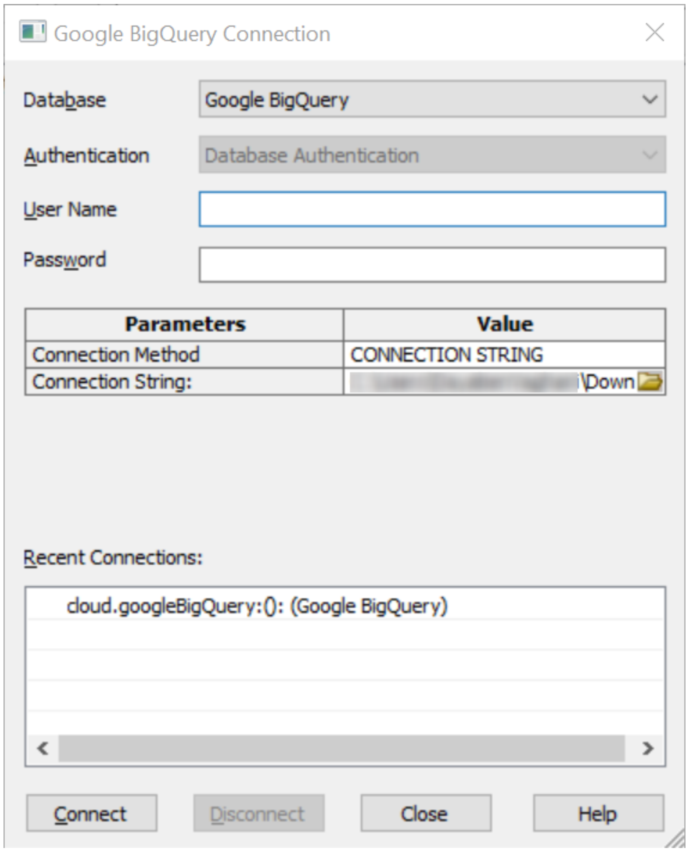
- **Copy** (📋): Use this option to copy the script.
- **Save** (💾): Use this option to save the generated script in the ERS, SQL, or DDL format.
- **Search** (🔍): Use this option to search through the generated schema.
- **Print** (🖨): Use this option to print the generated schema.
- **Replace** (🖨):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



Database: Google BigQuery

Authentication: Database Authentication

User Name:

Password:

Parameters	Value
Connection Method	CONNECTION STRING
Connection String:	[redacted]\Down

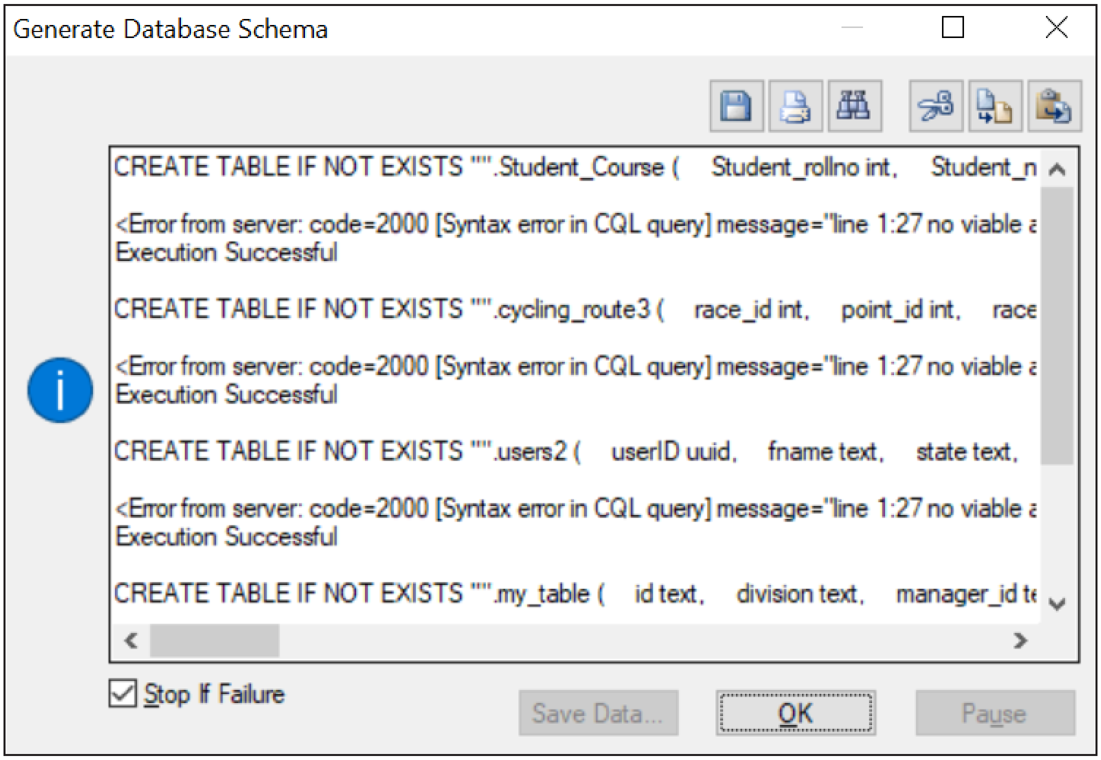
cloud.googleBigQuery:(): (Google BigQuery)

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.



```
CREATE TABLE IF NOT EXISTS "".Student_Course (    Student_rollno int,    Student_n

<Error from server: code=2000 [Syntax error in CQL query] message="line 1:27 no viable a
Execution Successful

CREATE TABLE IF NOT EXISTS "".cycling_route3 (    race_id int,    point_id int,    race

<Error from server: code=2000 [Syntax error in CQL query] message="line 1:27 no viable a
Execution Successful

CREATE TABLE IF NOT EXISTS "".users2 (    userID uuid,    fname text,    state text,

<Error from server: code=2000 [Syntax error in CQL query] message="line 1:27 no viable a
Execution Successful

CREATE TABLE IF NOT EXISTS "".my_table (    id text,    division text,    manager_id te
```

☑ Stop If Failure

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
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
Materialized View Syntax Option	Specifies the materialized view syntax options for schema generation	**Create**: Indicates whether the Create syntax for materialized views is executed **Drop**: Indicates whether the Drop syntax for materialized views is 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 gen-

		erated 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.

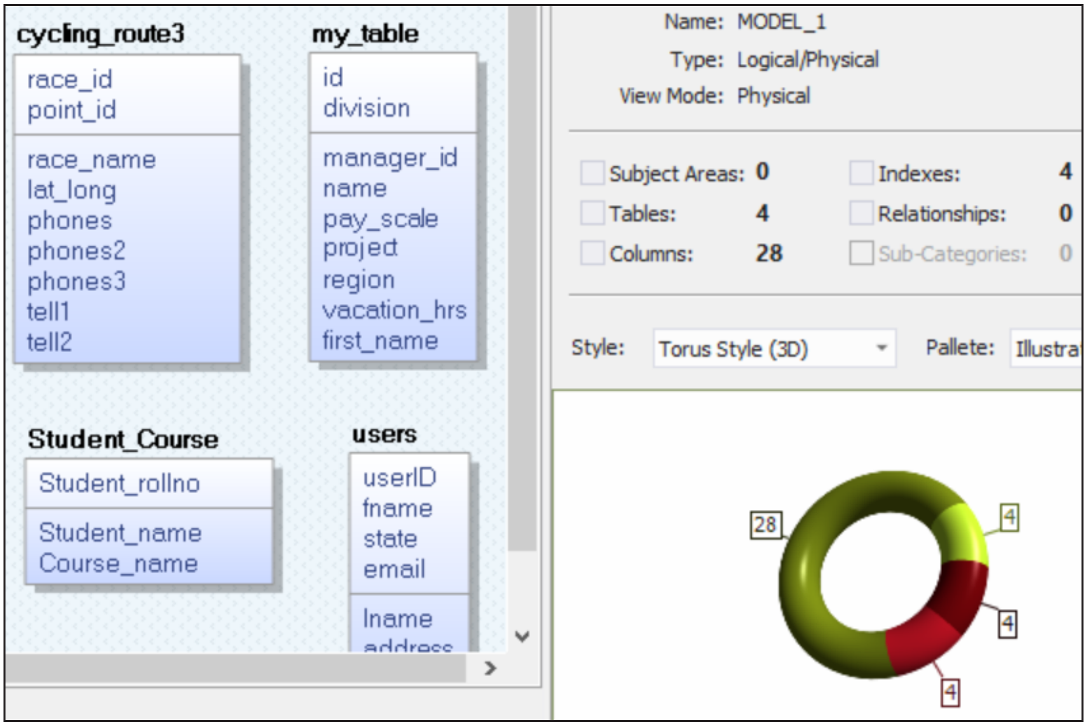
This topic walks you through the steps to compare a Google BigQuery model with database.

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.



cycling_route3
race_id
point_id
race_name
lat_long
phones
phones2
phones3
tell1
tell2

my_table
id
division
manager_id
name
pay_scale
project
region
vacation_hrs
first_name

Student_Course
Student_rollno
Student_name
Course_name

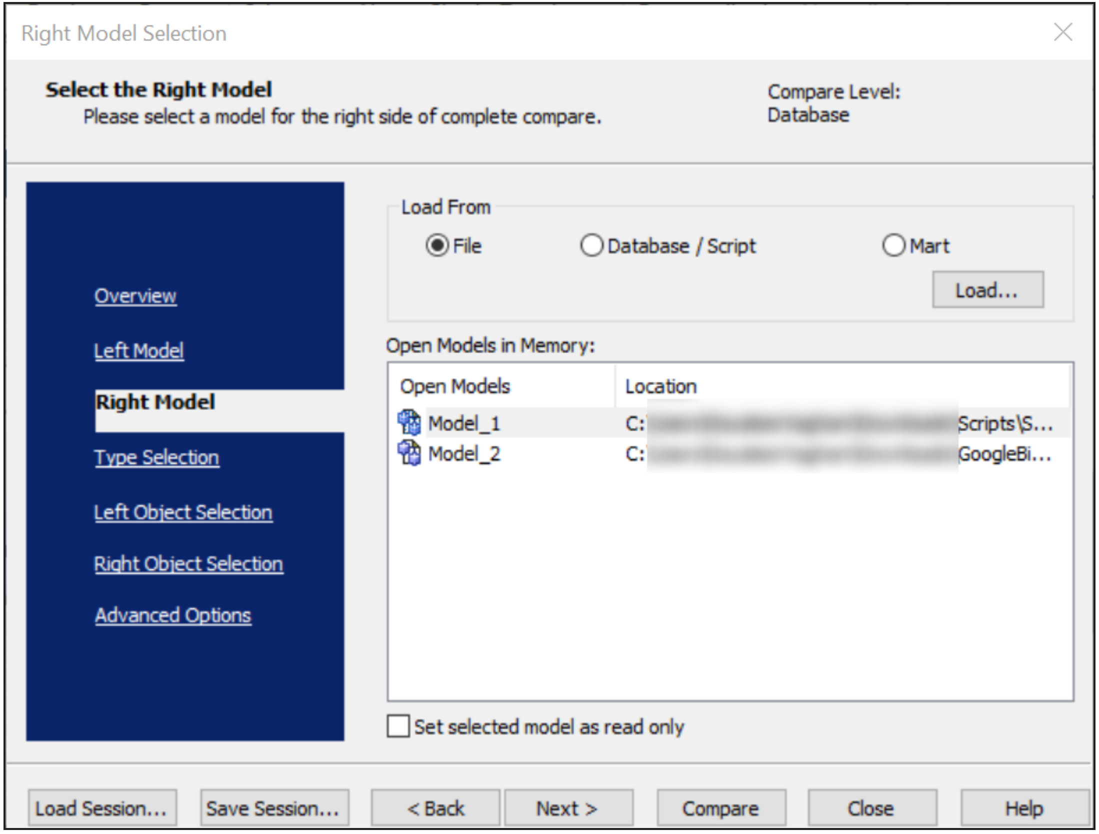
users
userID
fname
state
email
lname
address

Name: MODEL_1
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	0	□ Indexes:	4
□ Tables:	4	□ Relationships:	0
□ Columns:	28	□ Sub-Categories:	0

Style: Torus Style (3D)   Pallete: Illustra

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.



Right Model Selection

Select the Right Model
Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

● File ○ Database / Script ○ Mart

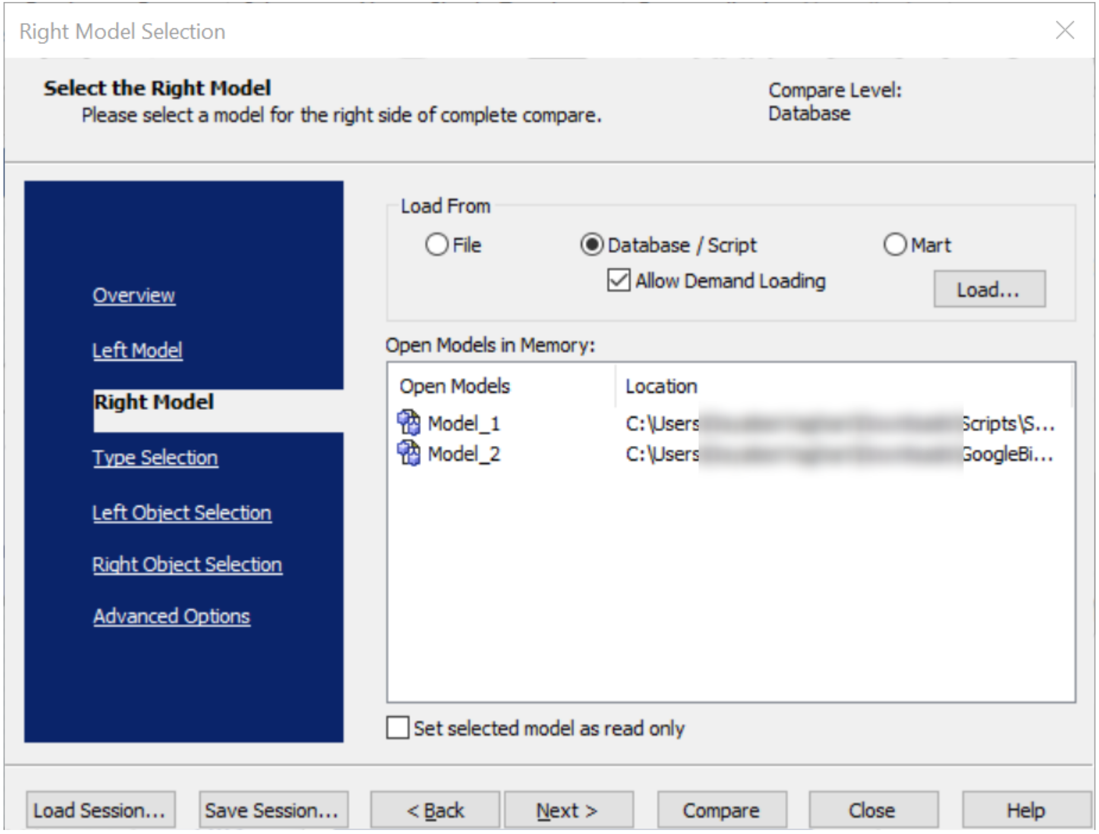
Open Models in Memory:

Open Models	Location
Model_1	C:[redacted]Scripts\S...
Model_2	C:[redacted]GoogleBi...

☐ Set selected model as read only

3. Click **Database/Script**.

   By default, the Allow Demand Loading option is selected.



Right Model Selection

Select the Right Model
Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

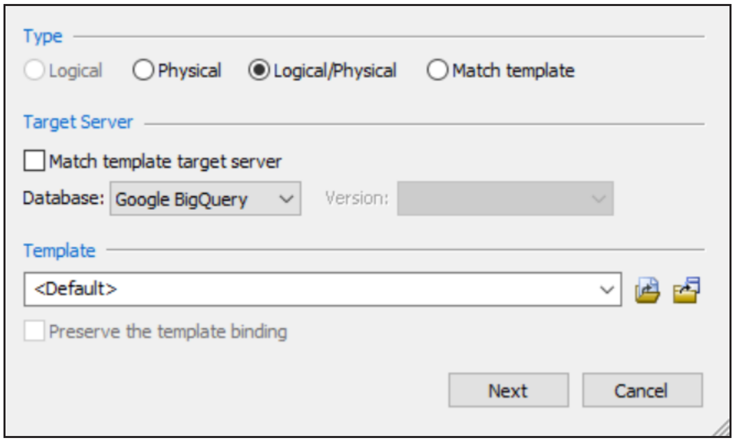
Open Models in Memory:

Open Models	Location
Model_1	C:\Users[redacted]Scripts\S...
Model_2	C:\Users[redacted]GoogleBi...

□ Set selected model as read only

4. Click **Load**.

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



Type

○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

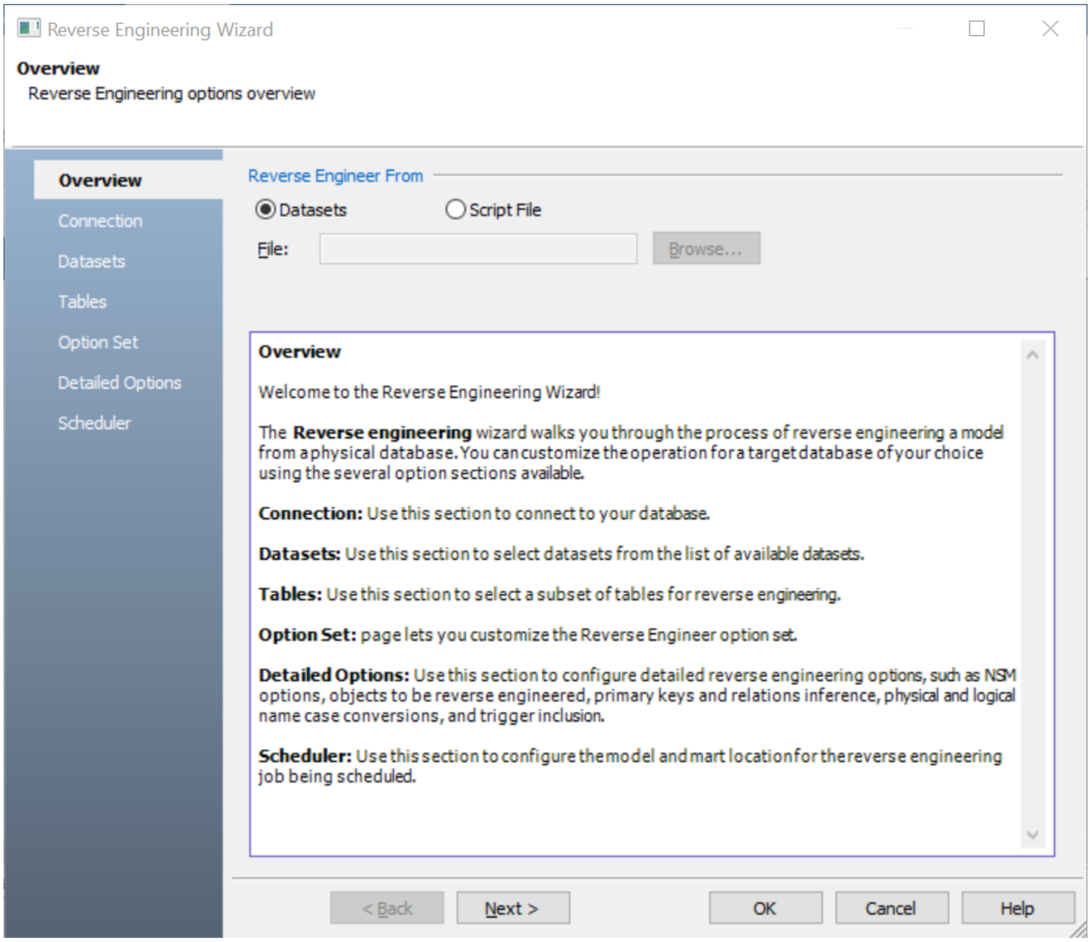
Database: Google BigQuery Version:

Template

<Default>

□ Preserve the template binding

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



Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Datasets: Use this section to select datasets from the list of available datasets.

Tables: Use this section to select a subset of tables for reverse engineering.

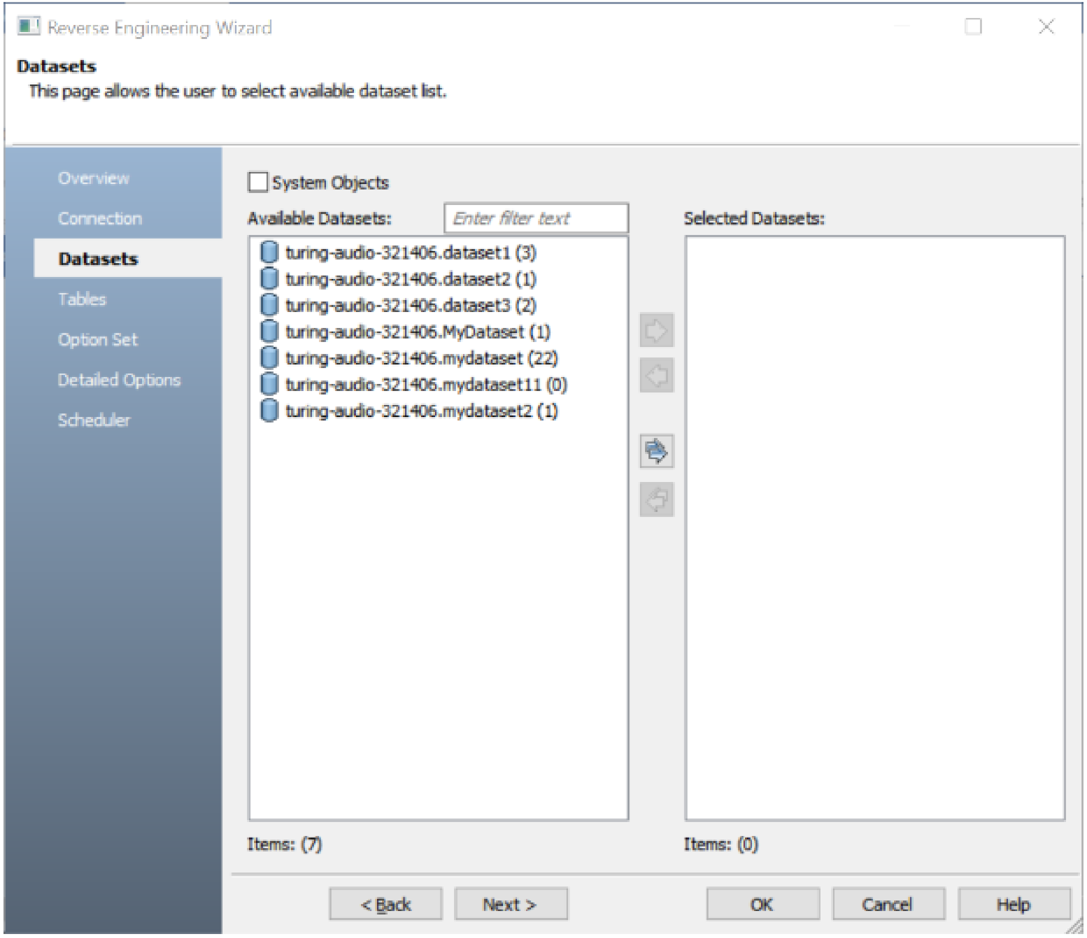
Option Set: page lets you customize the Reverse Engineer option set.

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

6. Click **Datasets**. Then, click **Next**.
   The Connection section appears. Use this section to connect to the database from which you want to reverse engineer the model.

7. After connection is established, click **Next**.
   The Datasets section appears. It displays a list of available datasets.



Reverse Engineering Wizard

Datasets

This page allows the user to select available dataset list.

□ System Objects

Available Datasets: Enter filter text

- turing-audio-321406.dataset1 (3)
- turing-audio-321406.dataset2 (1)
- turing-audio-321406.dataset3 (2)
- turing-audio-321406.MyDataset (1)
- turing-audio-321406.mydataset (22)
- turing-audio-321406.mydataset11 (0)
- turing-audio-321406.mydataset2 (1)

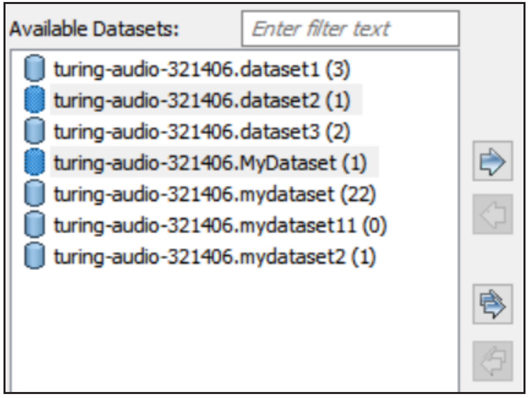
Items: (7)

Selected Datasets:

Items: (0)

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

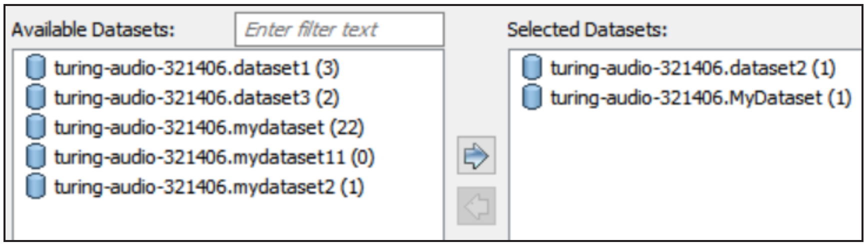
   This moves the selected datasets under Selected Datasets .



Available Datasets: Enter filter text

- turing-audio-321406.dataset1 (3)
- turing-audio-321406.dataset2 (1)
- turing-audio-321406.dataset3 (2)
- turing-audio-321406.MyDataset (1)
- turing-audio-321406.mydataset (22)
- turing-audio-321406.mydataset11 (0)
- turing-audio-321406.mydataset2 (1)

9. Click **Next** and in the Tables section, click ⇛.
   This selects all the available tables.



Available Datasets: Enter filter text

- turing-audio-321406.dataset1 (3)
- turing-audio-321406.dataset3 (2)
- turing-audio-321406.mydataset (22)
- turing-audio-321406.mydataset11 (0)
- turing-audio-321406.mydataset2 (1)

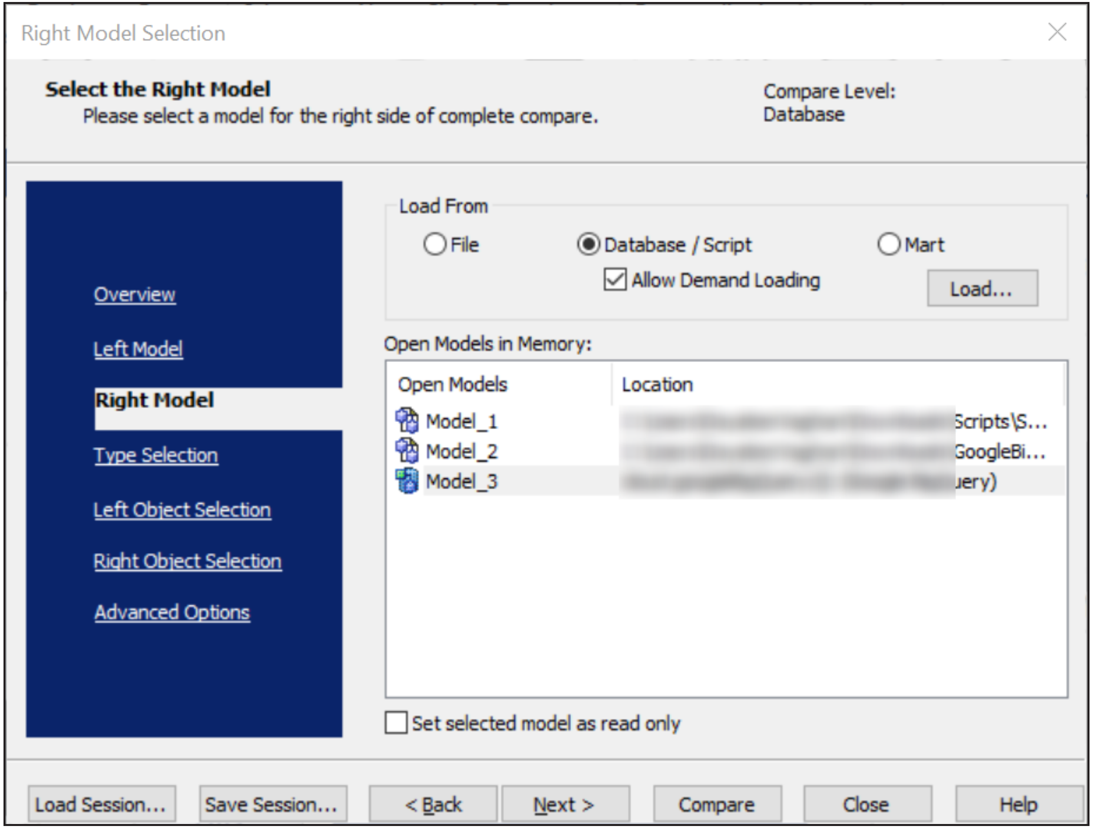
Selected Datasets:

- turing-audio-321406.dataset2 (1)
- turing-audio-321406.MyDataset (1)

10. Click **Next** and in the Option Set section, keep the default configuration.

11. Click **Next** and in the Detail Options section, keep the default configuration.

12. Click **OK**.
    The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse engineered.



Right Model Selection

Select the Right Model
Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

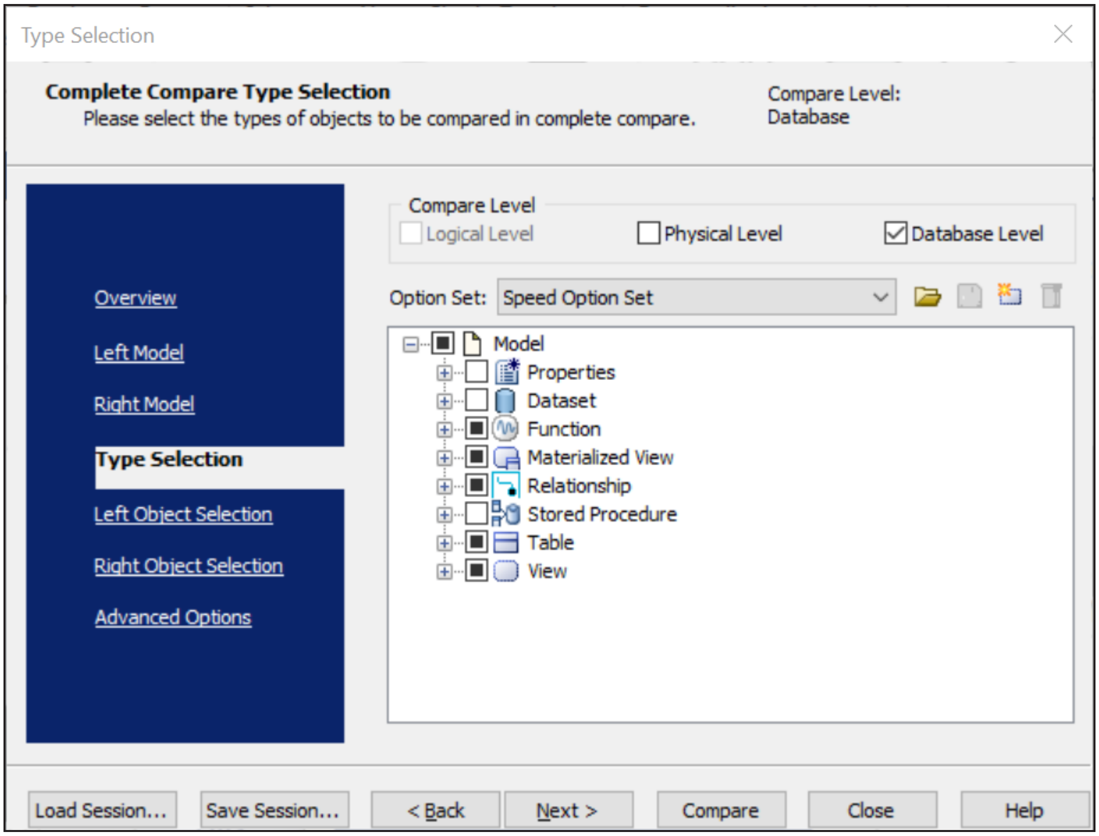
Open Models in Memory:

Open Models	Location
Model_1	[redacted]Scripts\S...
Model_2	[redacted]GoogleBi...
Model_3	[redacted]uery)

□ Set selected model as read only

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

For example, the following image shows the default options.



Type Selection

Complete Compare Type Selection

Please select the types of objects to be compared in complete compare.

Compare Level:
Database

Compare Level

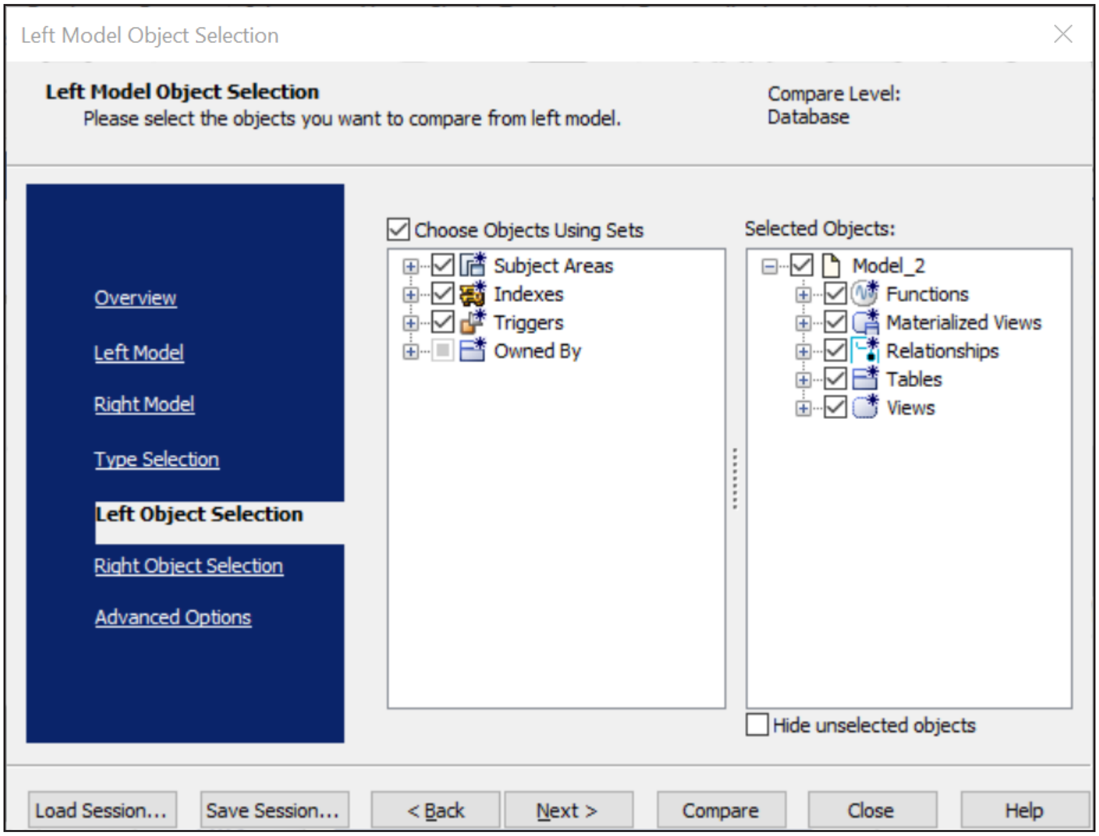
□ Logical Level ☐ Physical Level ☑ Database Level

Option Set: Speed Option Set

- ■ Model
  - □ Properties
  - □ Dataset
  - ■ Function
  - ■ Materialized View
  - ■ Relationship
  - □ Stored Procedure
  - ■ Table
  - ■ View

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

For example, the following image shows the default options.



Left Model Object Selection

Left Model Object Selection
Please select the objects you want to compare from left model.

Compare Level:
Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- ▪ Owned By

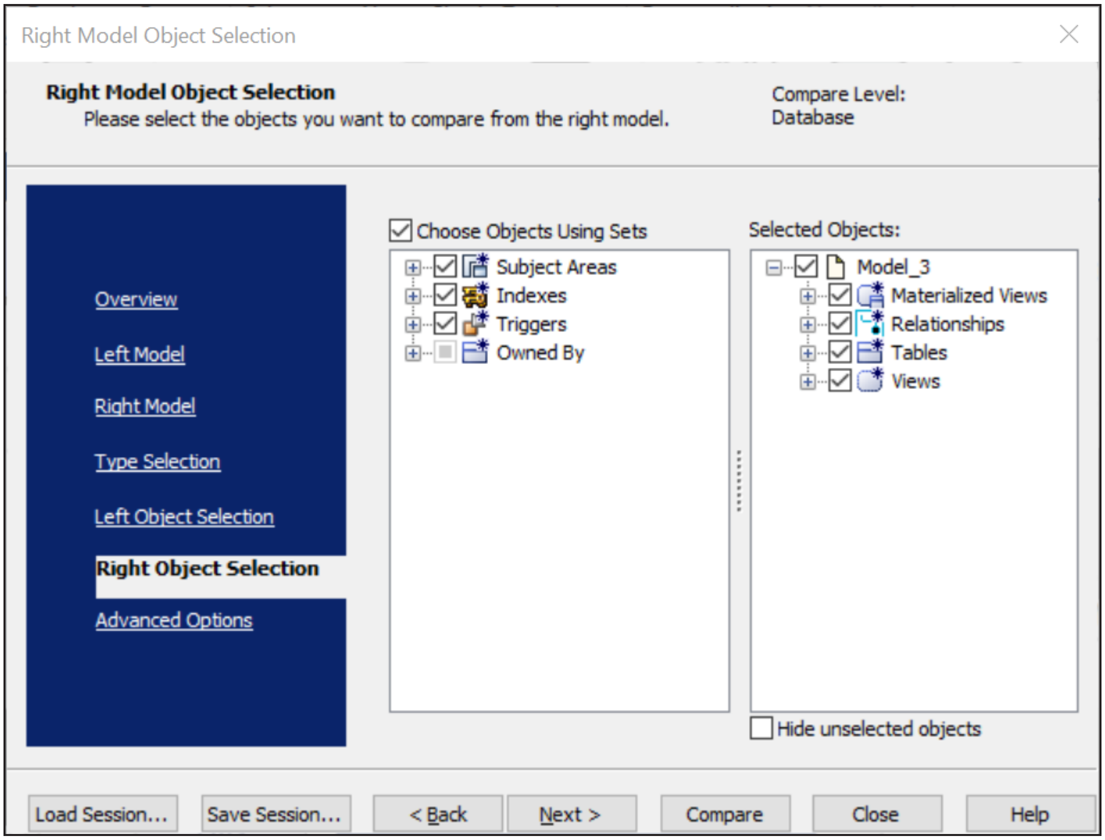
Selected Objects:

- ☑ Model_2
  - ☑ Functions
  - ☑ Materialized Views
  - ☑ Relationships
  - ☑ Tables
  - ☑ Views

☐ Hide unselected objects

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

For example, the following image shows the default options.



Right Model Object Selection

Right Model Object Selection
Please select the objects you want to compare from the right model.

Compare Level:
Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

Selected Objects:

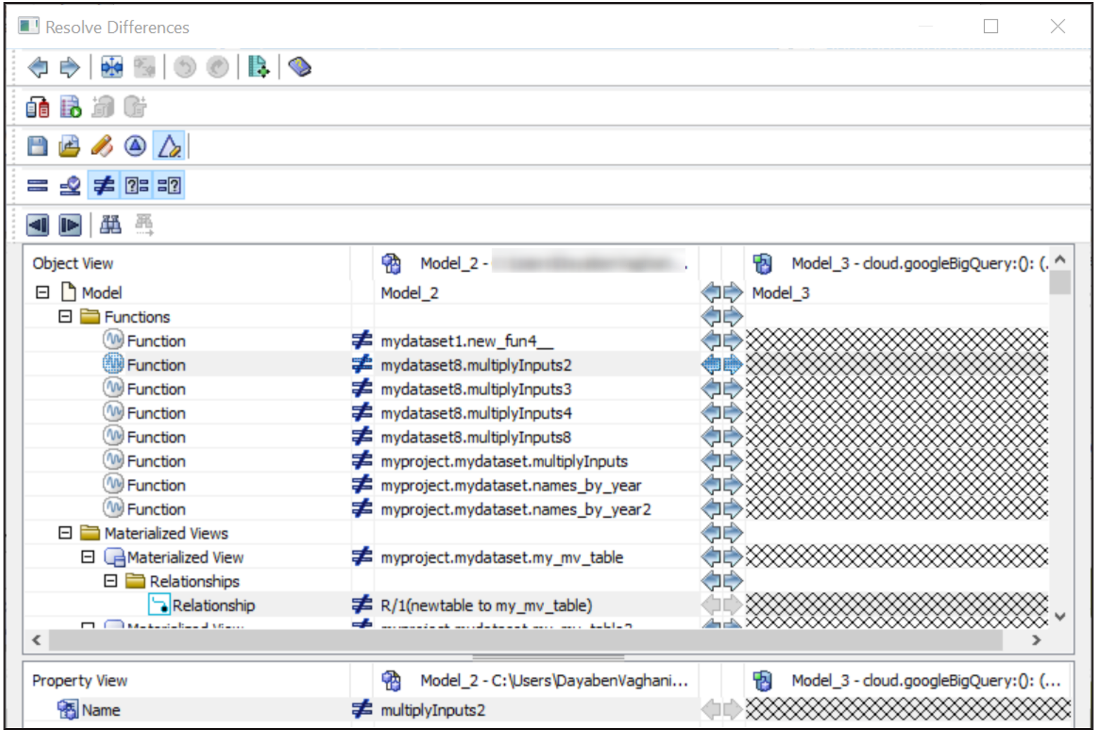
- ☑ Model_3
  - ☑ Materialized Views
  - ☑ Relationships
  - ☑ Tables
  - ☑ Views

□ Hide unselected objects

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.



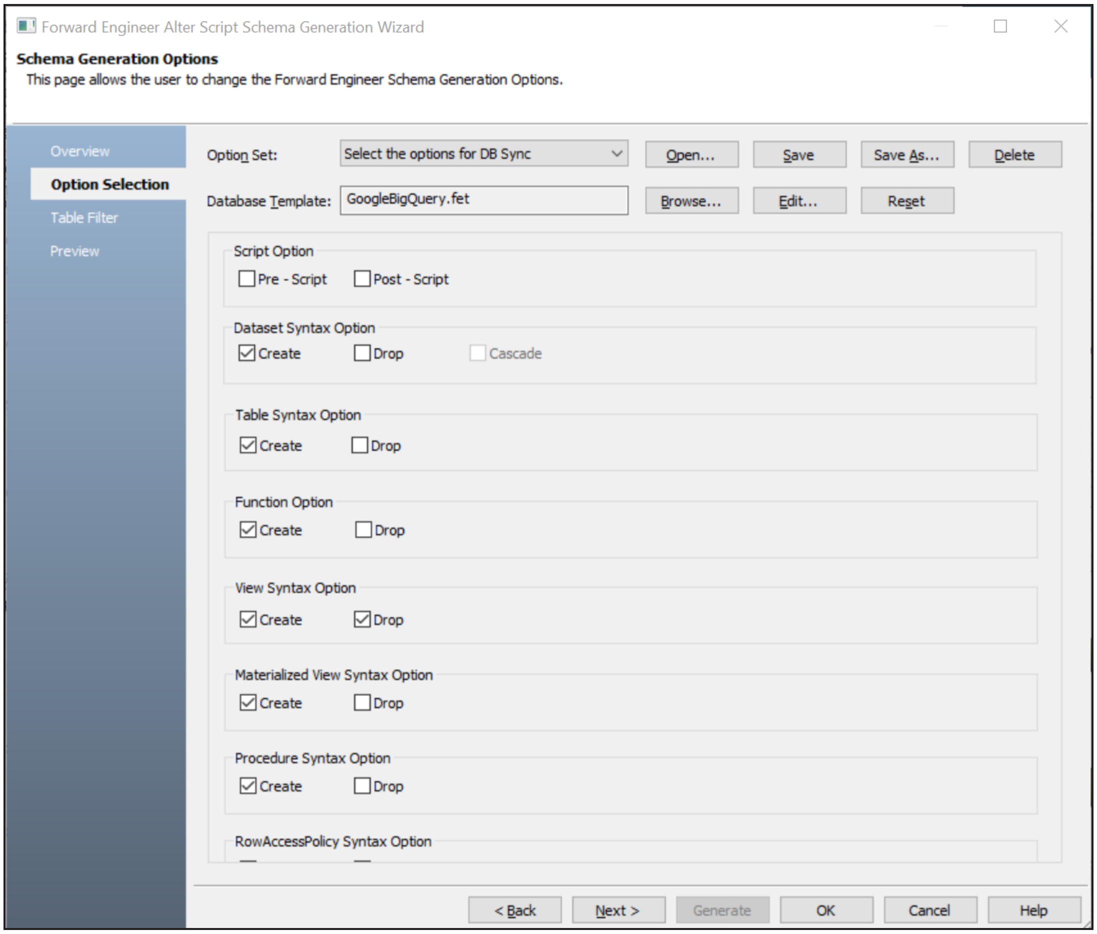
Object View	Model_2 - [redacted]	Model_3 - cloud.googleBigQuery:(): (...
Model	Model_2	Model_3
Functions		
Function	≠ mydataset1.new_fun4__	
Function	≠ mydataset8.multiplyInputs2	
Function	≠ mydataset8.multiplyInputs3	
Function	≠ mydataset8.multiplyInputs4	
Function	≠ mydataset8.multiplyInputs8	
Function	≠ myproject.mydataset.multiplyInputs	
Function	≠ myproject.mydataset.names_by_year	
Function	≠ myproject.mydataset.names_by_year2	
Materialized Views		
Materialized View	≠ myproject.mydataset.my_mv_table	
Relationships		
Relationship	≠ R/1(newtable to my_mv_table)	

Property View	Model_2 - C:\Users\DayabenVaghani...	Model_3 - cloud.googleBigQuery:(): (...
Name	≠ multiplyInputs2	

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.



Forward Engineer Alter Script Schema Generation Wizard

Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Select the options for DB Sync

Database Template: GoogleBigQuery.fet

Script Option
□ Pre - Script □ Post - Script

Dataset Syntax Option
☑ Create □ Drop □ Cascade

Table Syntax Option
☑ Create □ Drop

Function Option
☑ Create □ Drop

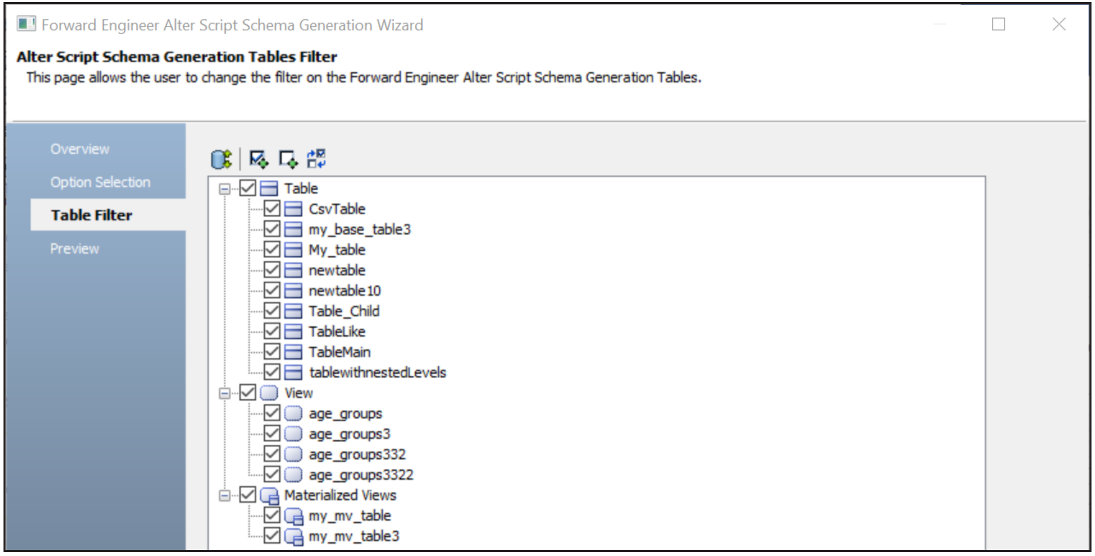
View Syntax Option
☑ Create ☑ Drop

Materialized View Syntax Option
☑ Create □ Drop

Procedure Syntax Option
☑ Create □ Drop

RowAccessPolicy Syntax Option

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



Alter Script Schema Generation Tables Filter

This page allows the user to change the filter on the Forward Engineer Alter Script Schema Generation Tables.

- ☑ Table
  - ☑ CsvTable
  - ☑ my_base_table3
  - ☑ My_table
  - ☑ newtable
  - ☑ newtable10
  - ☑ Table_Child
  - ☑ TableLike
  - ☑ TableMain
  - ☑ tablewithnestedLevels
- ☑ View
  - ☑ age_groups
  - ☑ age_groups3
  - ☑ age_groups332
  - ☑ age_groups3322
- ☑ Materialized Views
  - ☑ my_mv_table
  - ☑ my_mv_table3

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**.

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

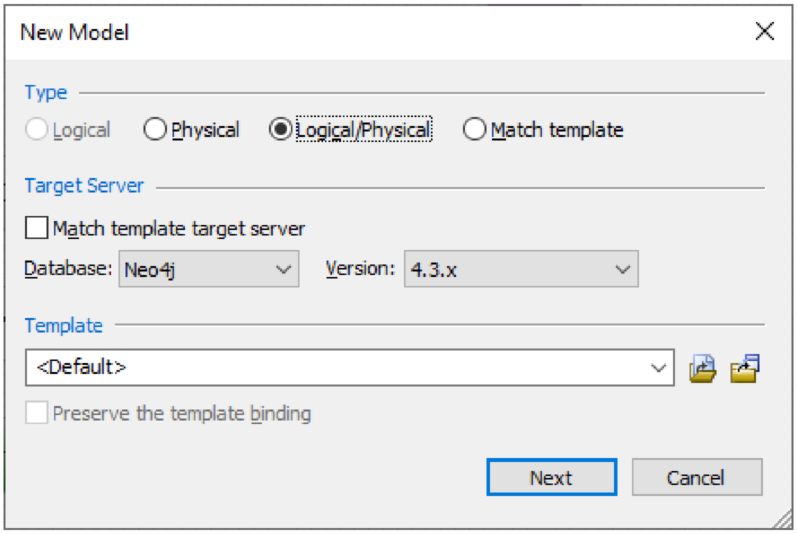
Reverse Engineering Models

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.



New Model

Type

○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

Database: Neo4j

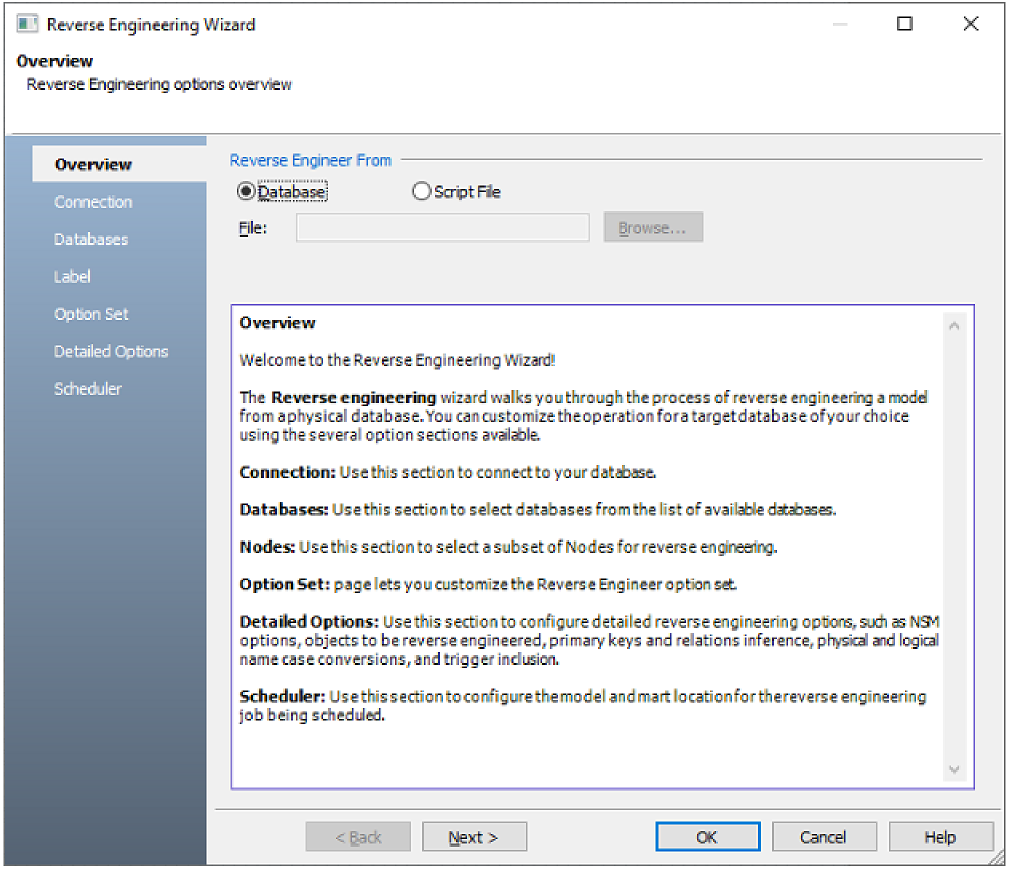
Version: 4.3.x

Template

<Default>

□ Preserve the template binding

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



Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Databases: Use this section to select databases from the list of available databases.

Nodes: Use this section to select a subset of Nodes for reverse engineering.

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

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

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

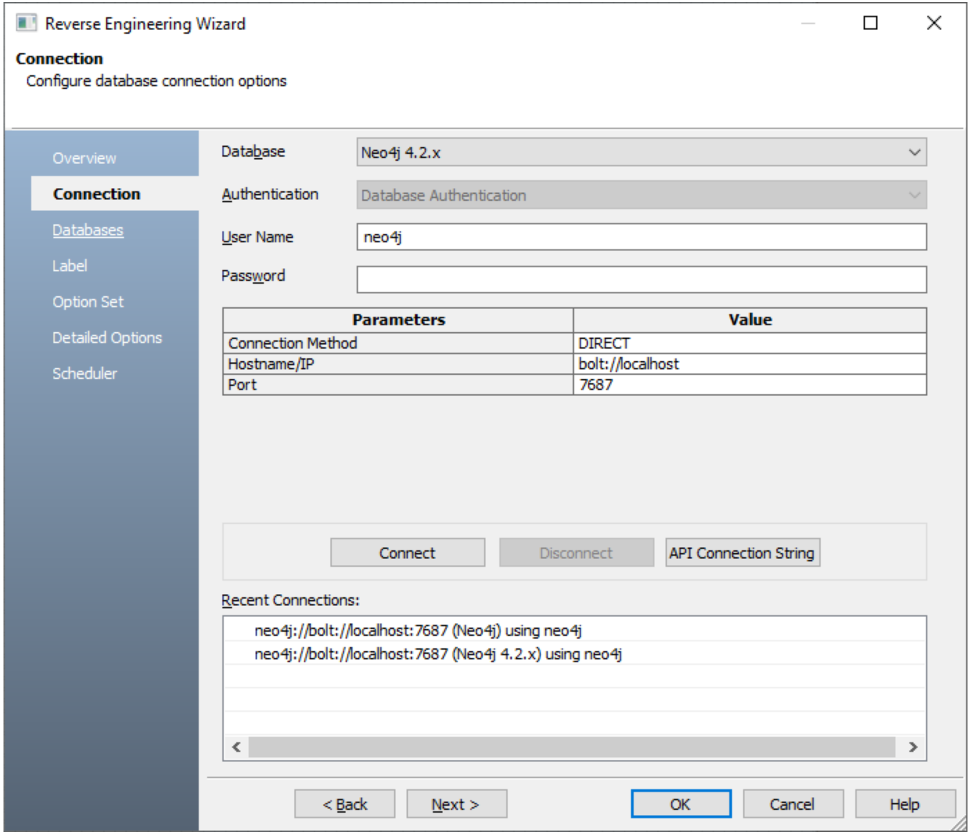
     If you click **Database**, continue to step 5.

   - **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 section appears.



Connection

Configure database connection options

Database: Neo4j 4.2.x

Authentication: Database Authentication

User Name: neo4j

Password:

Parameters	Value
Connection Method	DIRECT
Hostname/IP	bolt://localhost
Port	7687

Recent Connections:

neo4j://bolt://localhost:7687 (Neo4j) using neo4j

neo4j://bolt://localhost:7687 (Neo4j 4.2.x) using neo4j

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

   The following table explains the connection parameters:

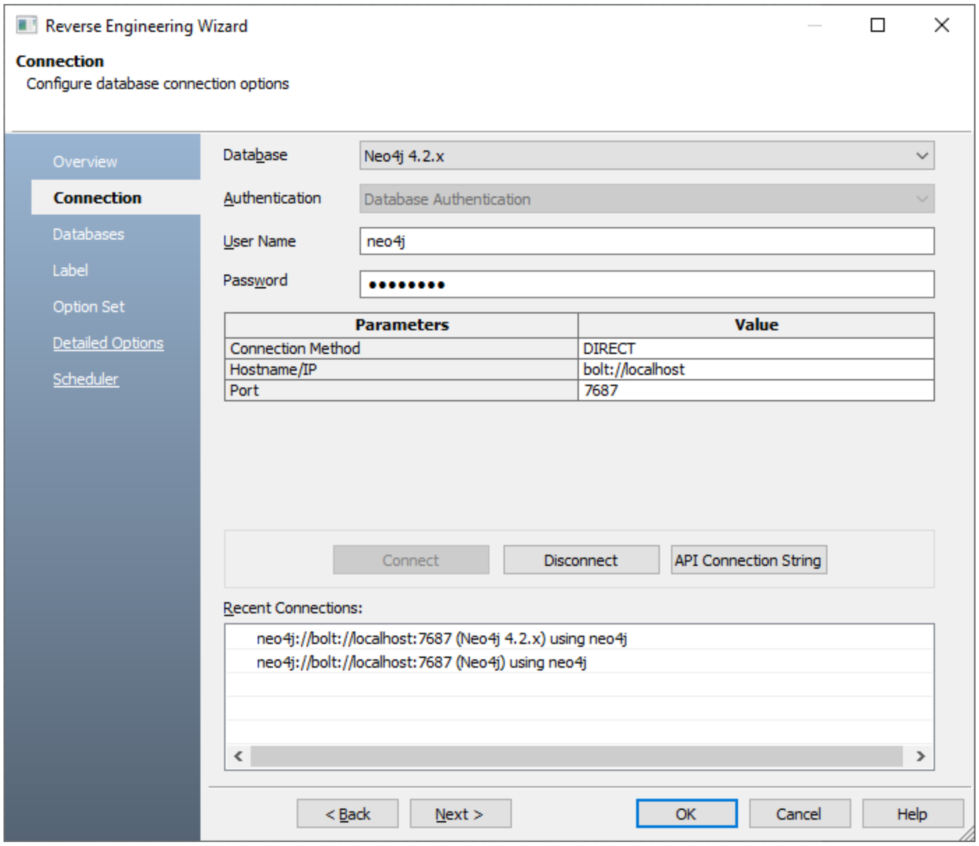
Parameter	Description	Additional Information
Connection Method	Specifies the type of connection you want to use. Using **Direct** connects to your database directly.	
Hostname/IP	Specifies the hostname or IP address of the server where your database is hosted in one of the following formats: • *Neo4j://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



Connection

Configure database connection options

Database: Neo4j 4.2.x

Authentication: Database Authentication

User Name: neo4j

Password: ••••••••

Parameters	Value
Connection Method	DIRECT
Hostname/IP	bolt://localhost
Port	7687

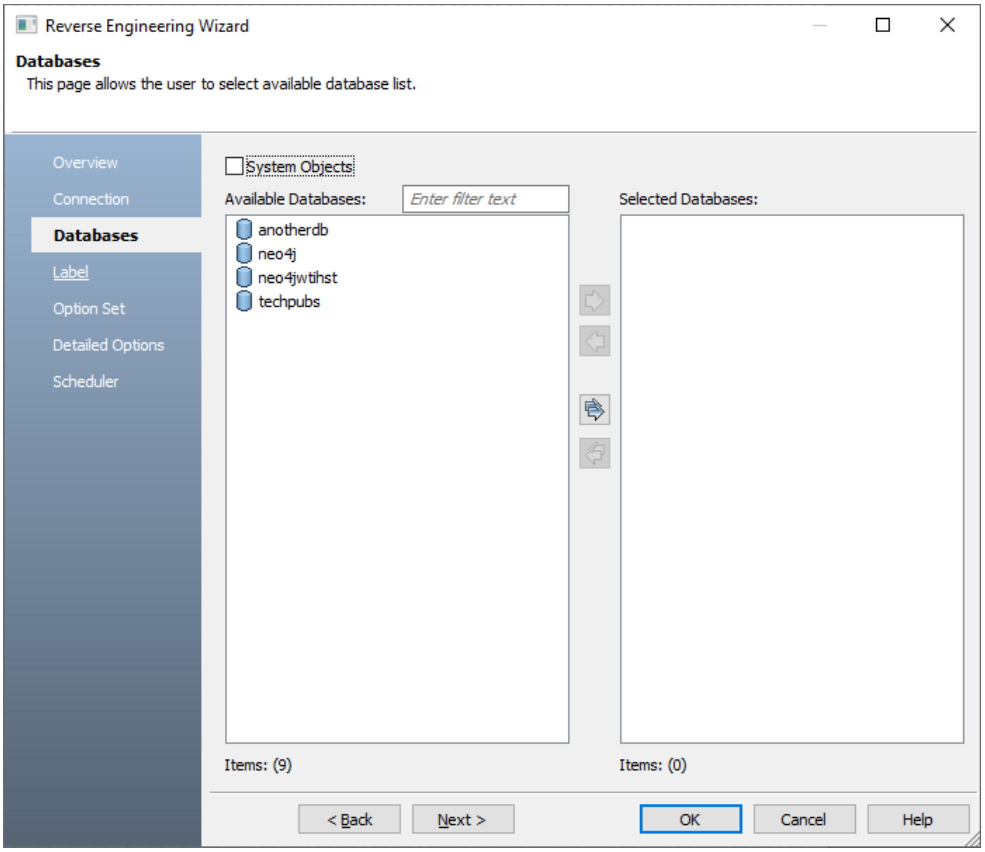
Recent Connections:

neo4j://bolt://localhost:7687 (Neo4j 4.2.x) using neo4j

neo4j://bolt://localhost:7687 (Neo4j) using neo4j

8. Click **Next**.

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



Databases

This page allows the user to select available database list.

☐ System Objects

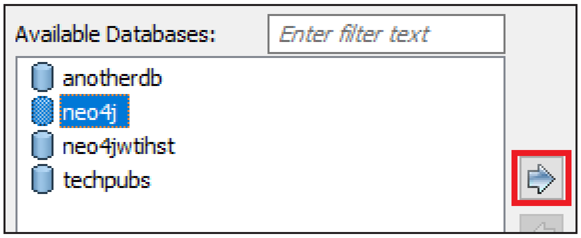
Available Databases: anotherdb, neo4j, neo4jwtihst, techpubs

Items: (9)

Selected Databases:

Items: (0)

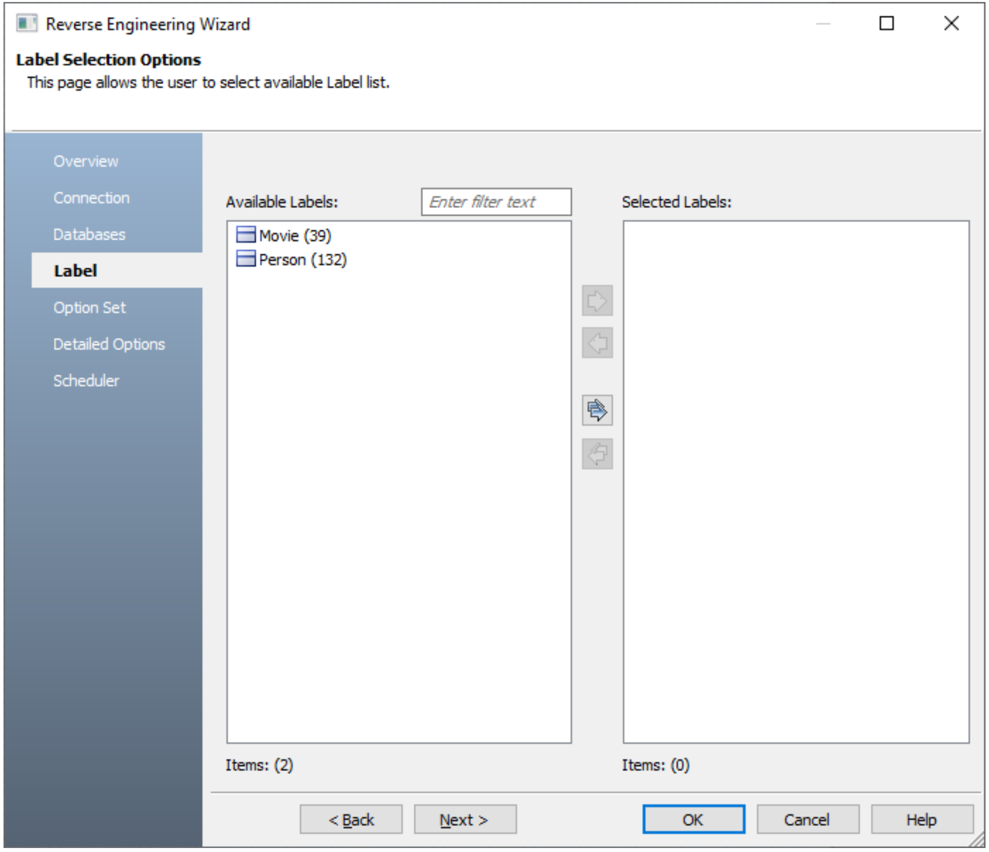
9. Under **Available Databases**, select the databases that you want to reverse engineer. Then, click ⇨.



Available Databases: anotherdb, neo4j, neo4jwtihst, techpubs

10. Click **Next**.

    The Label section appears. It displays a list of available labels in the databases that you selected in step 9.

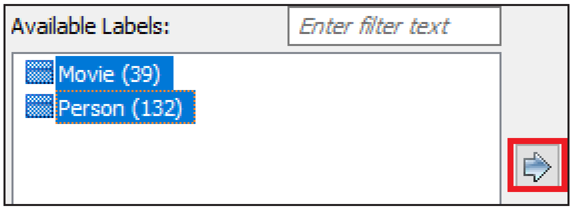


Reverse Engineering Wizard

Label Selection Options

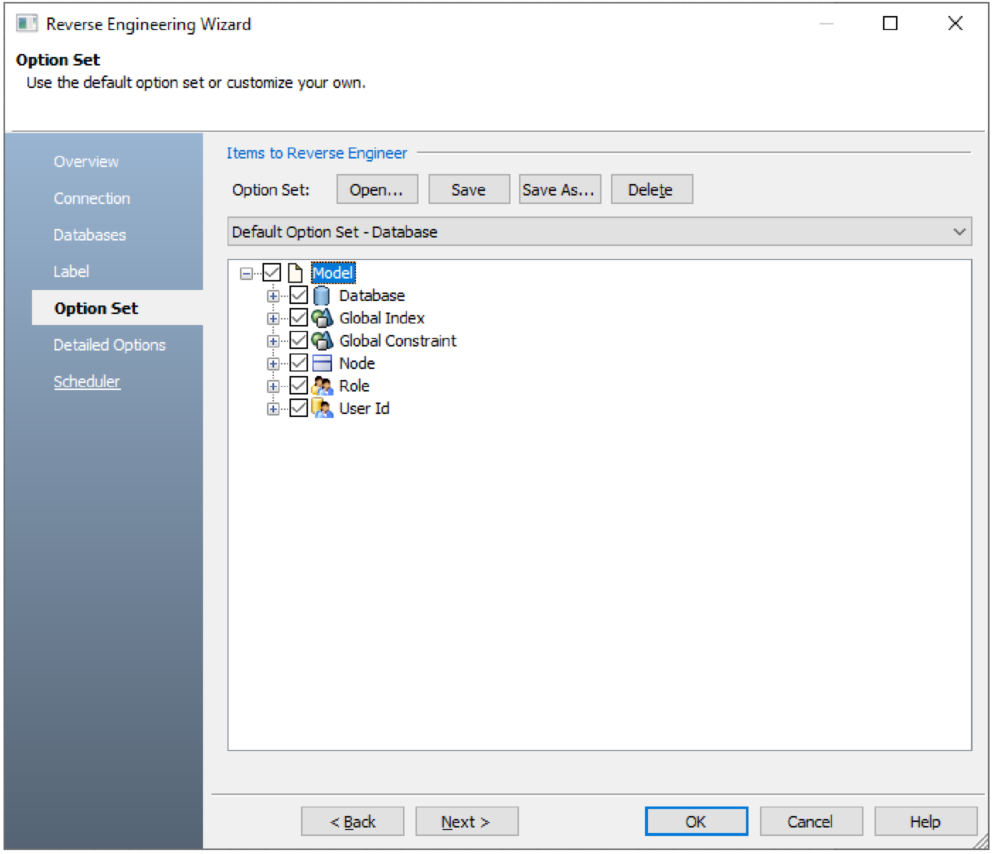
This page allows the user to select available Label list.

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



12. Click **Next**.

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



Reverse Engineering Wizard

Option Set

Use the default option set or customize your own.

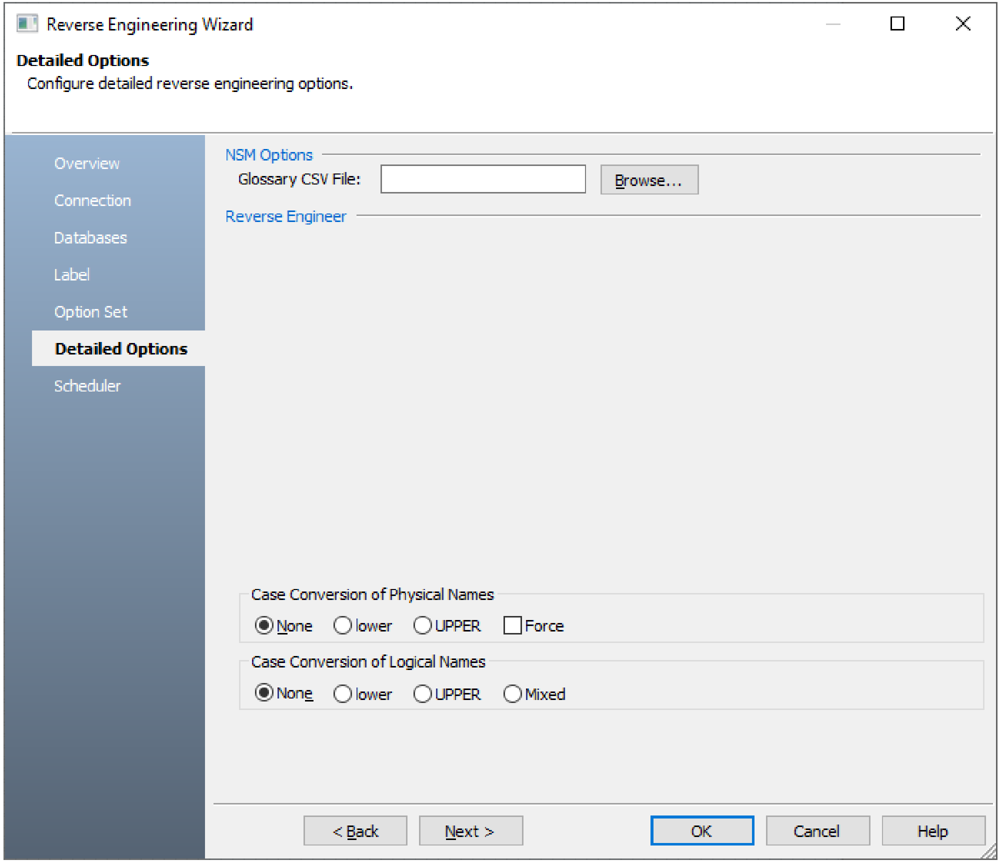
Items to Reverse Engineer

Option Set: Default Option Set - Database

- ☑ Model
  - ☑ Database
  - ☑ Global Index
  - ☑ Global Constraint
  - ☑ Node
  - ☑ Role
  - ☑ User Id

13. Click **Next**.

    The Detailed Options section appears. Set up appropriate options based on your requirement.



Detailed Options
Configure detailed reverse engineering options.

NSM Options

Glossary CSV File:

Reverse Engineer

Case Conversion of Physical Names

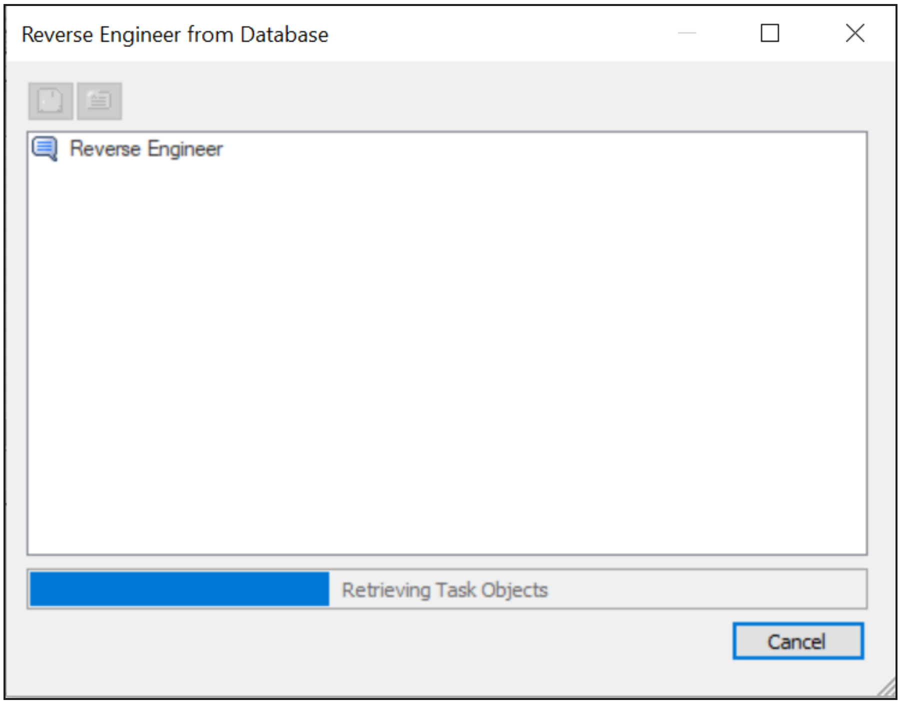
● None ○ lower ○ UPPER □ Force

Case Conversion of Logical Names

● None ○ lower ○ UPPER ○ Mixed

14. Click **OK**.

The reverse engineering process starts.

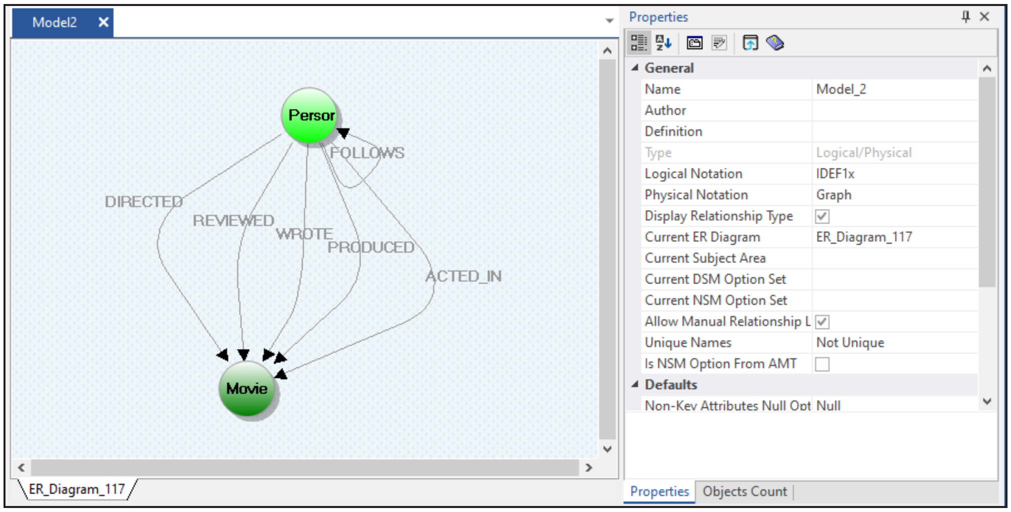


Reverse Engineer from Database

Reverse Engineer

Retrieving Task Objects

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

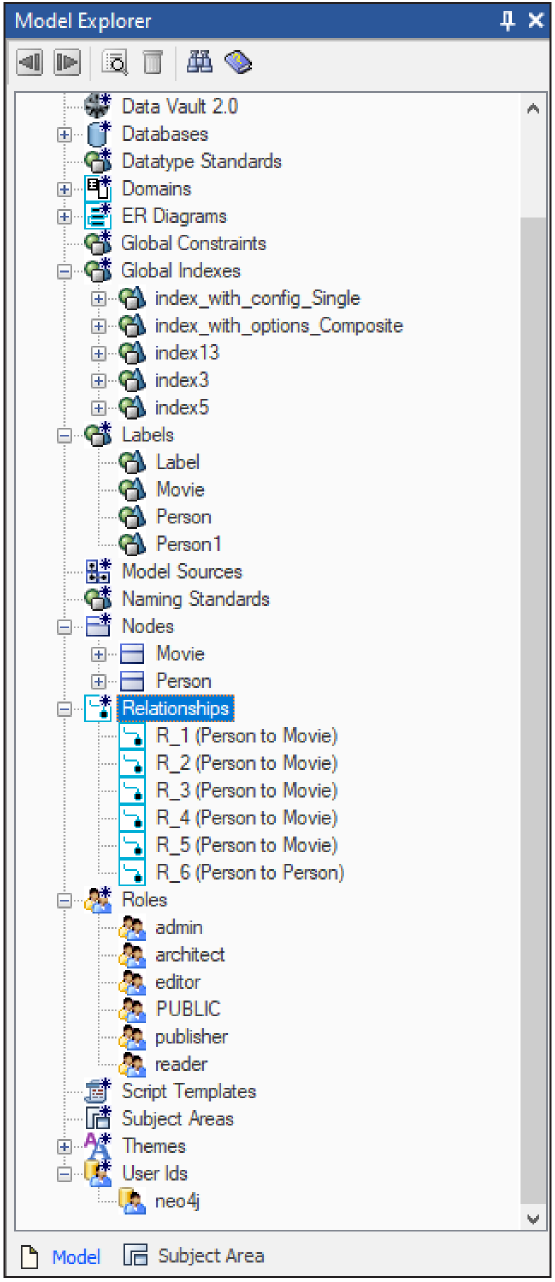


Properties

General	
Name	Model_2
Author	
Definition	
Type	Logical/Physical
Logical Notation	IDEF1x
Physical Notation	Graph
Display Relationship Type	☑
Current ER Diagram	ER_Diagram_117
Current Subject Area	
Current DSM Option Set	
Current NSM Option Set	
Allow Manual Relationship L	☑
Unique Names	Not Unique
Is NSM Option From AMT	□
Defaults	
Non-Key Attributes Null Opt	Null

> 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 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.



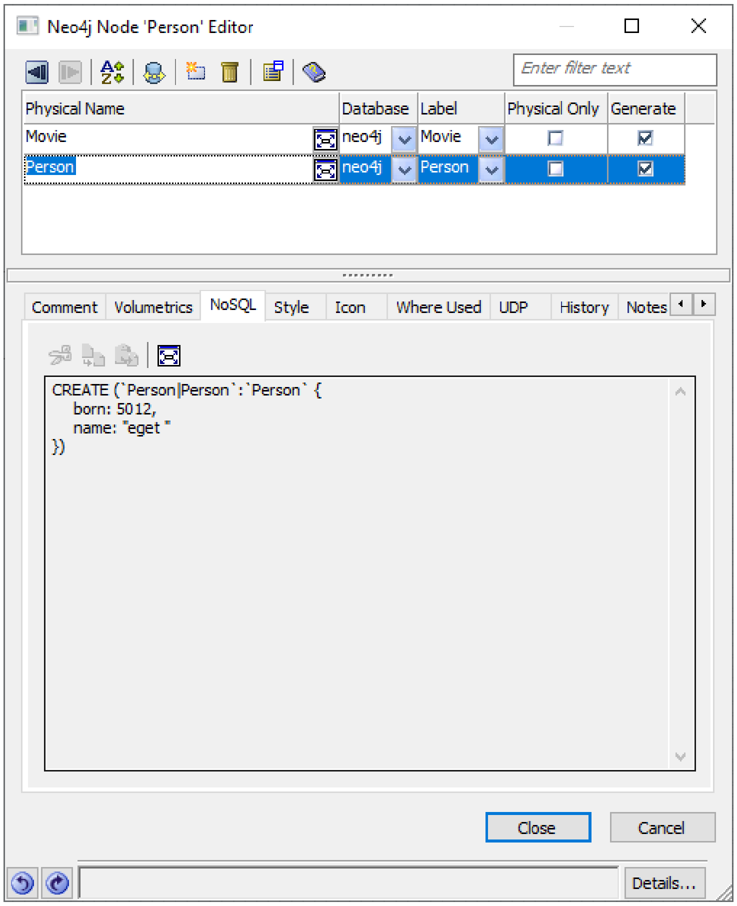
Model Explorer

- Data Vault 2.0
- Databases
- Datatype Standards
- Domains
- ER Diagrams
- Global Constraints
- Global Indexes
  - index_with_config_Single
  - index_with_options_Composite
  - index13
  - index3
  - index5
- Labels
  - Label
  - Movie
  - Person
  - Person1
- Model Sources
- Naming Standards
- Nodes
  - Movie
  - Person
- Relationships
  - R_1 (Person to Movie)
  - R_2 (Person to Movie)
  - R_3 (Person to Movie)
  - R_4 (Person to Movie)
  - R_5 (Person to Movie)
  - R_6 (Person to Person)
- Roles
  - admin
  - architect
  - editor
  - PUBLIC
  - publisher
  - reader
- Script Templates
- Subject Areas
- Themes
- User Ids
  - neo4j

Model | Subject Area

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.



Physical Name	Database	Label	Physical Only	Generate
Movie	neo4j	Movie	☐	☑
Person	neo4j	Person	☐	☑

```
CREATE (`Person|Person`:`Person` {
    born: 5012,
    name: "eget "
})
```

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 where the reverse engineered model	When you schedule a job on a remote server, ensure the model path is same for remote and

	should be saved and its name	local server. For example: C:\Scheduler\<Model Name>.erwin
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.

Forward Engineering Models

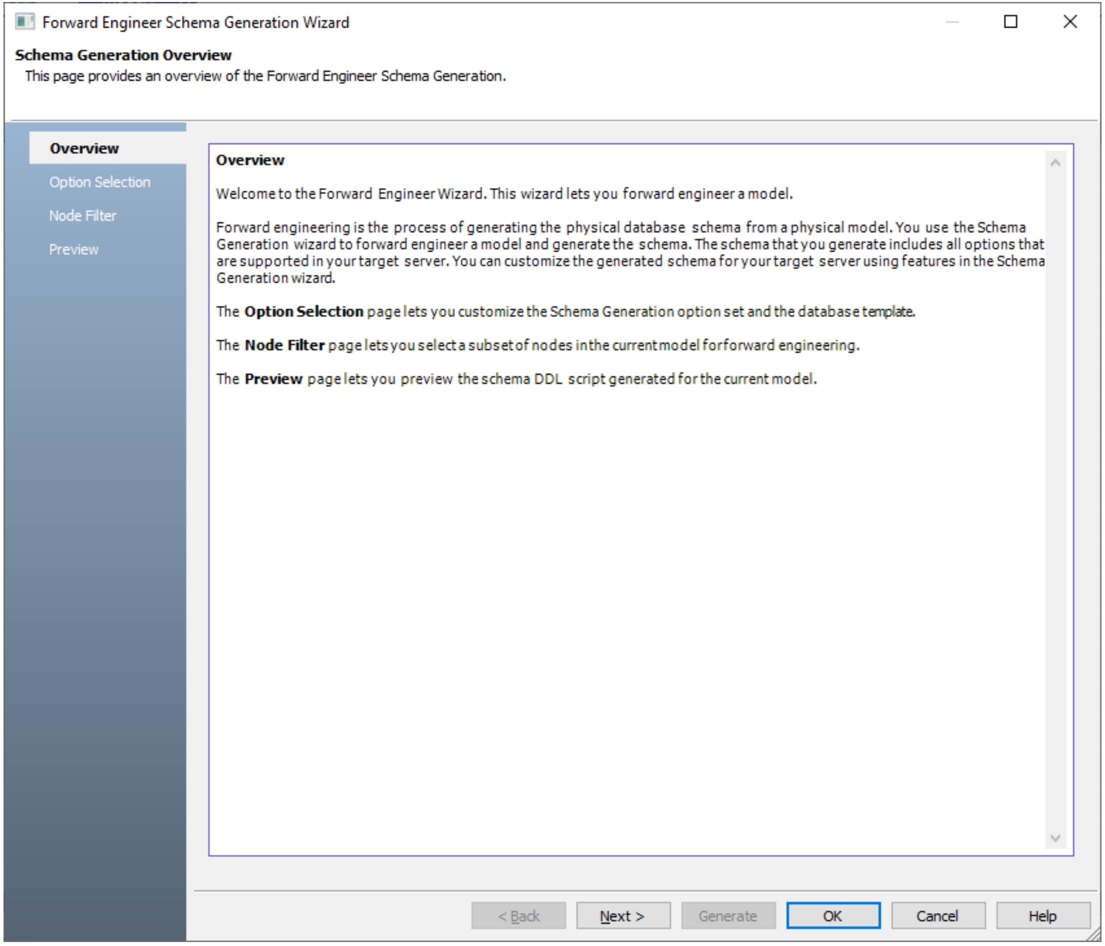
You can generate a physical database schema from a physical model using the Forward Engineering process. This topic walks you through the steps to forward engineer a Neo4j model. For detailed description of forward engineering options, refer to the Forward Engineering Options topic.

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.



Schema Generation Overview

This page provides an overview of the Forward Engineer Schema Generation.

Overview

Welcome to the Forward Engineer Wizard. This wizard lets you forward engineer a model.

Forward engineering is the process of generating the physical database schema from a physical model. You use the Schema Generation wizard to forward engineer a model and generate the schema. The schema that you generate includes all options that are supported in your target server. You can customize the generated schema for your target server using features in the Schema Generation wizard.

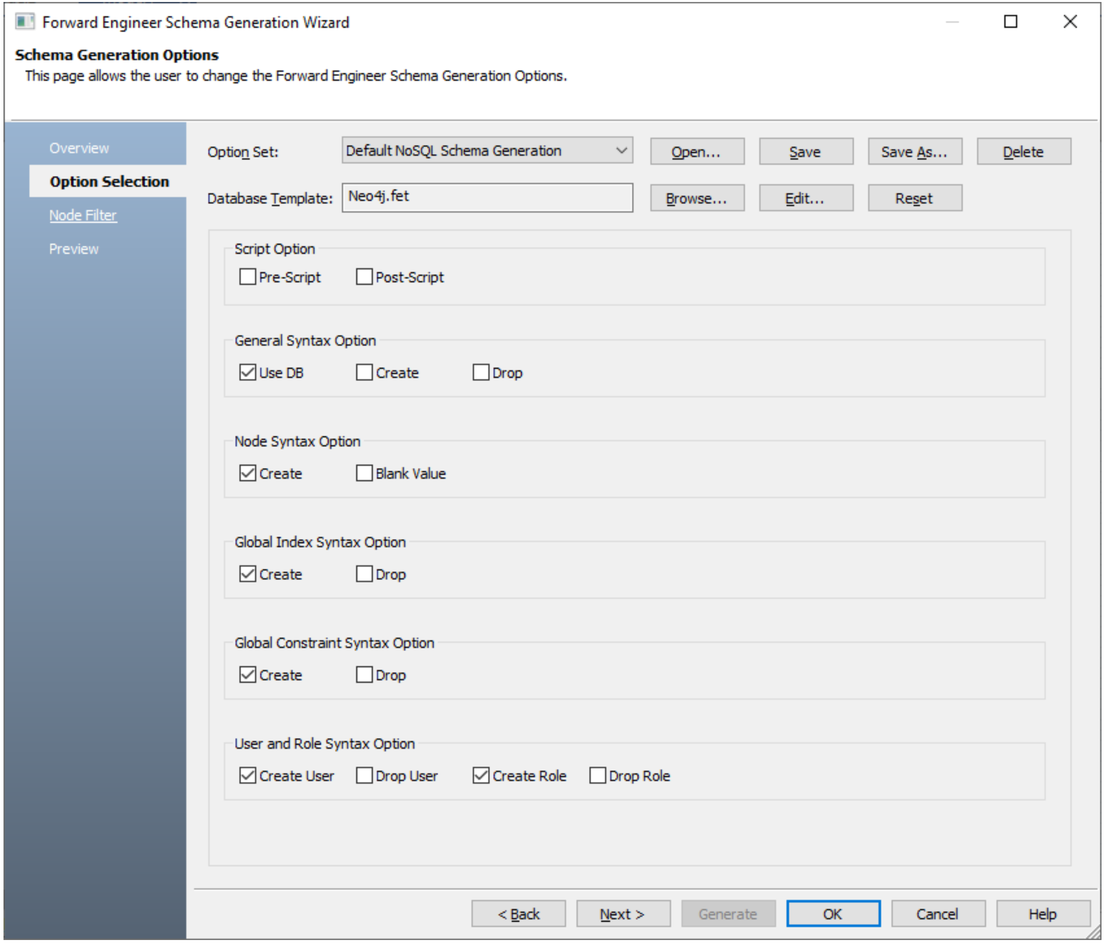
The **Option Selection** page lets you customize the Schema Generation option set and the database template.

The **Node Filter** page lets you select a subset of nodes in the current model for forward engineering.

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

3. Click **Option Selection**.

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



Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default NoSQL Schema Generation

Database Template: Neo4j.fet

Script Option

□ Pre-Script □ Post-Script

General Syntax Option

☑ Use DB □ Create □ Drop

Node Syntax Option

☑ Create □ Blank Value

Global Index Syntax Option

☑ Create □ Drop

Global Constraint Syntax Option

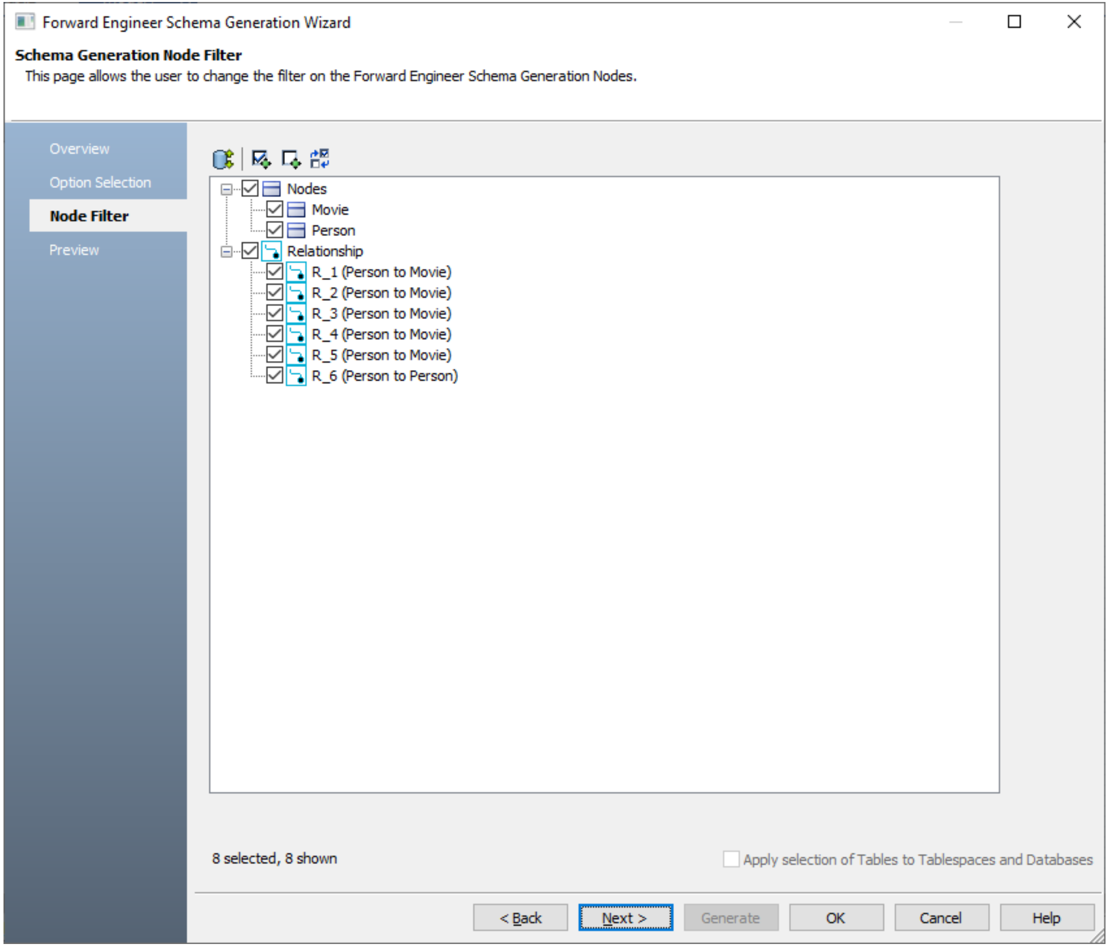
☑ Create □ Drop

User and Role Syntax Option

☑ Create User □ Drop User ☑ Create Role □ Drop Role

4. Click **Next**.

   The Node Filter section appears. It displays a list of nodes available in your model.



Forward Engineer Schema Generation Wizard

Schema Generation Node Filter

This page allows the user to change the filter on the Forward Engineer Schema Generation Nodes.

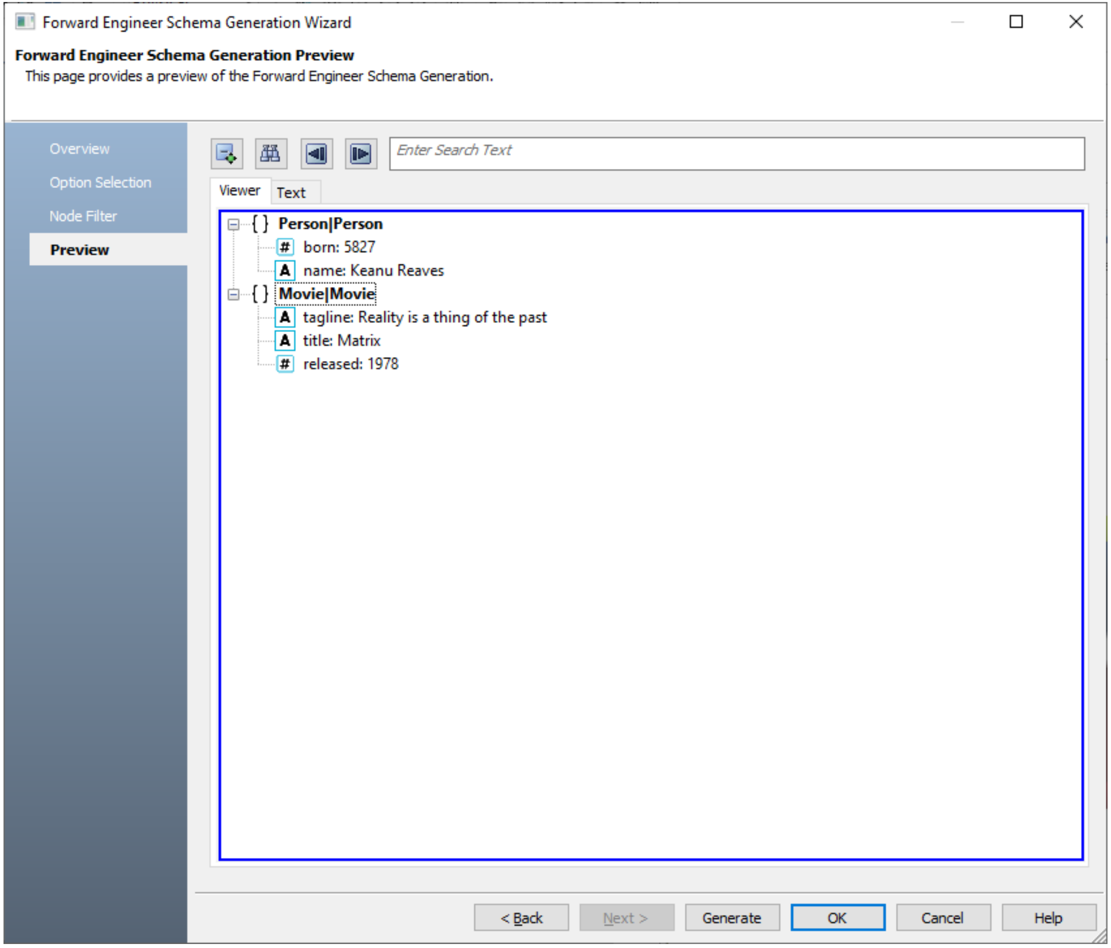
- ☑ Nodes
  - ☑ Movie
  - ☑ Person
- ☑ Relationship
  - ☑ R_1 (Person to Movie)
  - ☑ R_2 (Person to Movie)
  - ☑ R_3 (Person to Movie)
  - ☑ R_4 (Person to Movie)
  - ☑ R_5 (Person to Movie)
  - ☑ R_6 (Person to Person)

8 selected, 8 shown

□ Apply selection of Tables to Tablespaces and Databases

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.

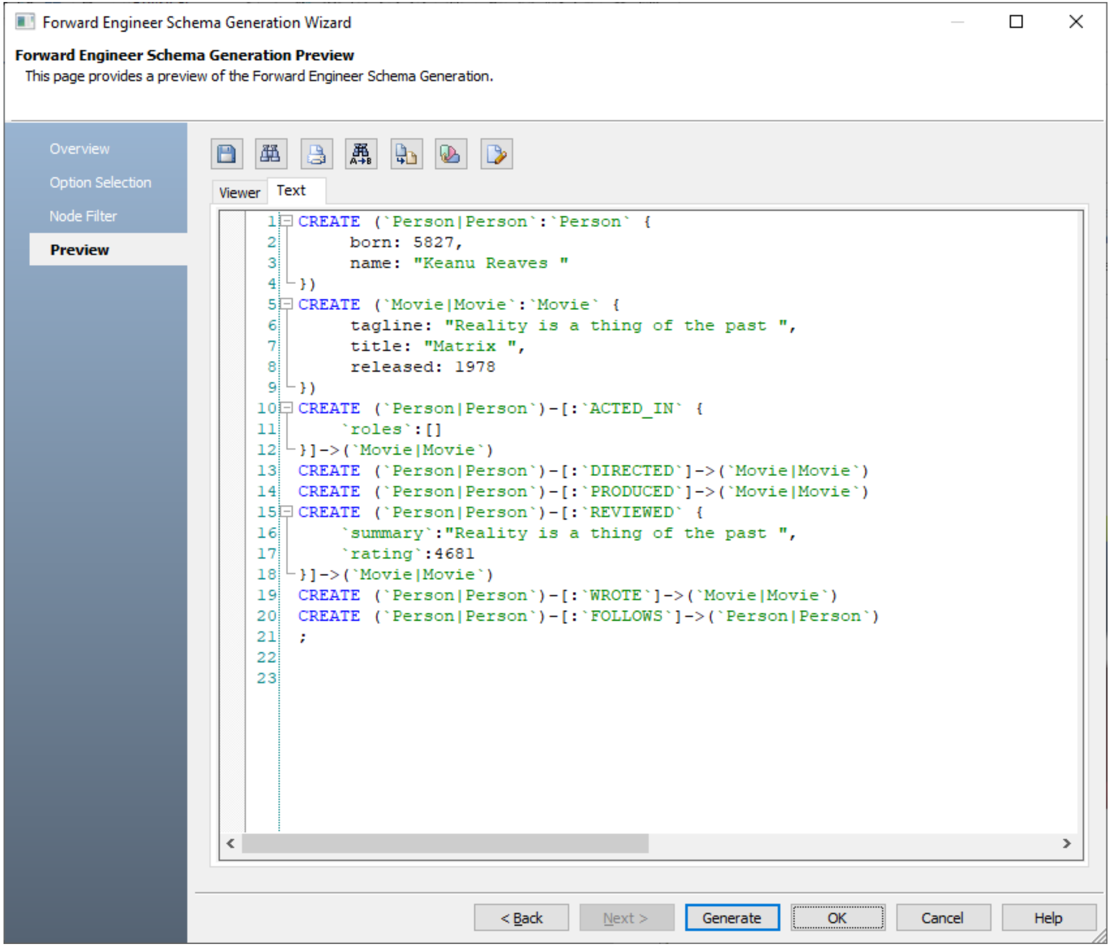


Forward Engineer Schema Generation Preview

This page provides a preview of the Forward Engineer Schema Generation.

```
{ } Person|Person
    # born: 5827
    A name: Keanu Reaves
{ } Movie|Movie
    A tagline: Reality is a thing of the past
    A title: Matrix
    # released: 1978
```

The script is available on the Text tab.



Forward Engineer Schema Generation Wizard

Forward Engineer Schema Generation Preview

This page provides a preview of the Forward Engineer Schema Generation.

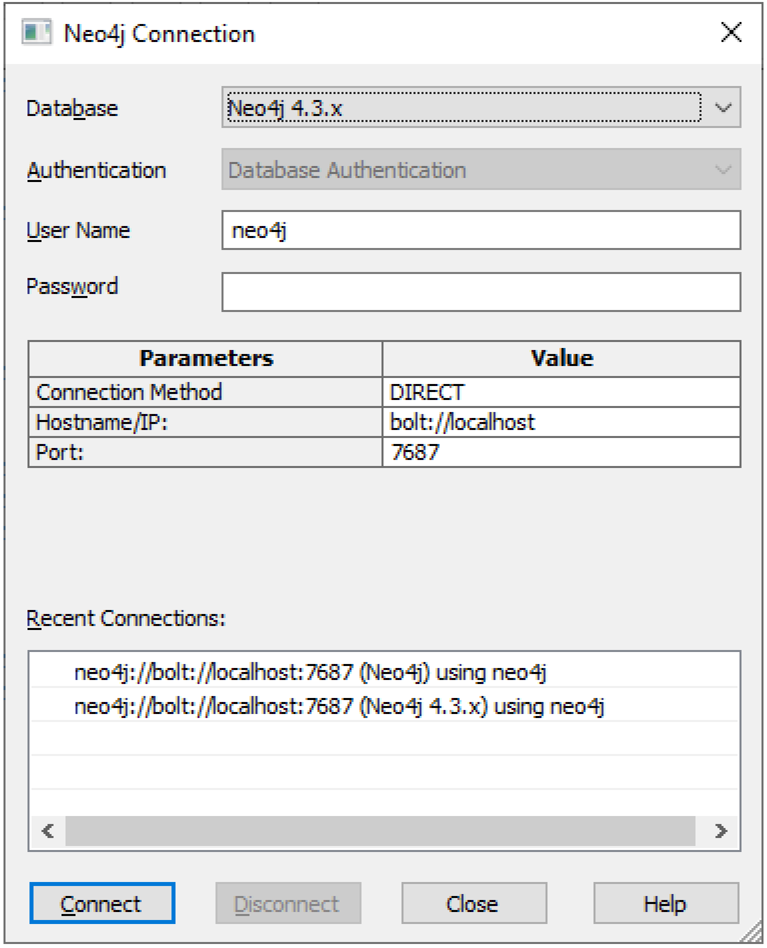
```
CREATE (`Person|Person`:`Person` {
      born: 5827,
      name: "Keanu Reaves "
})
CREATE (`Movie|Movie`:`Movie` {
      tagline: "Reality is a thing of the past ",
      title: "Matrix ",
      released: 1978
})
CREATE (`Person|Person`)-[:`ACTED_IN` {
     `roles`:[]
}]->(`Movie|Movie`)
CREATE (`Person|Person`)-[:`DIRECTED`]->(`Movie|Movie`)
CREATE (`Person|Person`)-[:`PRODUCED`]->(`Movie|Movie`)
CREATE (`Person|Person`)-[:`REVIEWED` {
     `summary`:"Reality is a thing of the past ",
     `rating`:4681
}]->(`Movie|Movie`)
CREATE (`Person|Person`)-[:`WROTE`]->(`Movie|Movie`)
CREATE (`Person|Person`)-[:`FOLLOWS`]->(`Person|Person`)
;
```

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.



Database: Neo4j 4.3.x

Authentication: Database Authentication

User Name: neo4j

Password:

Parameters	Value
Connection Method	DIRECT
Hostname/IP:	bolt://localhost
Port:	7687

Recent Connections:

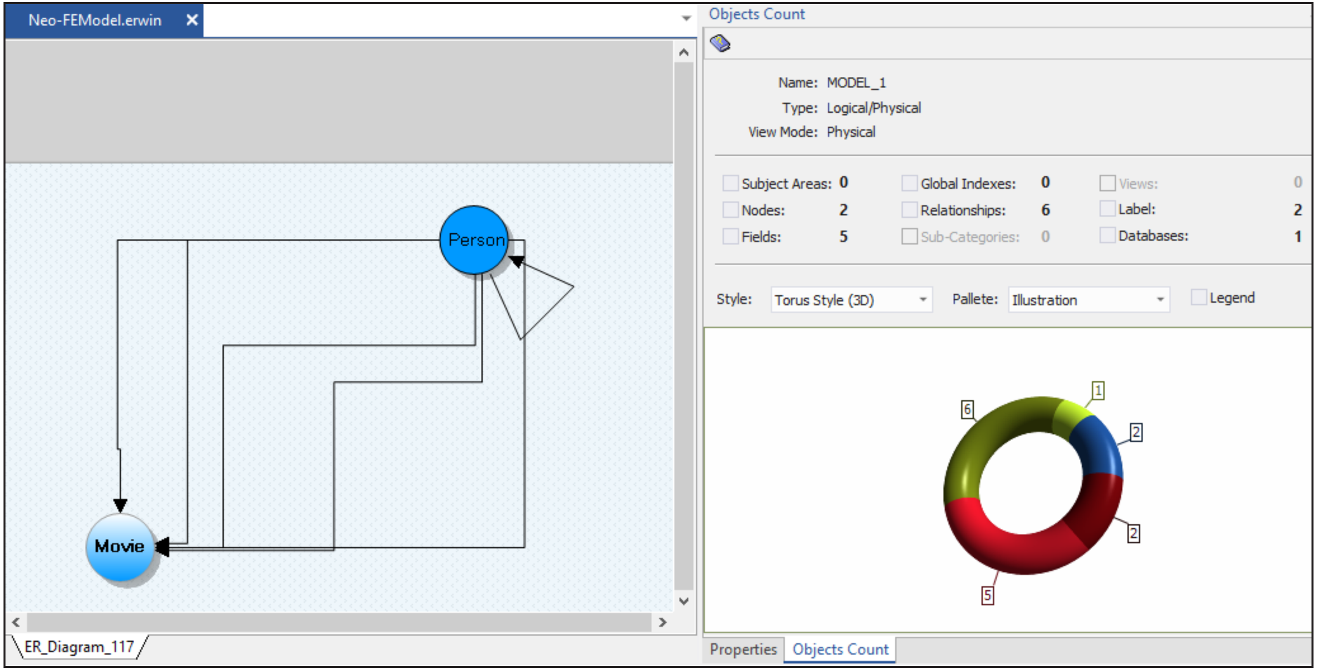
neo4j://bolt://localhost:7687 (Neo4j) using neo4j

neo4j://bolt://localhost:7687 (Neo4j 4.3.x) using neo4j

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.

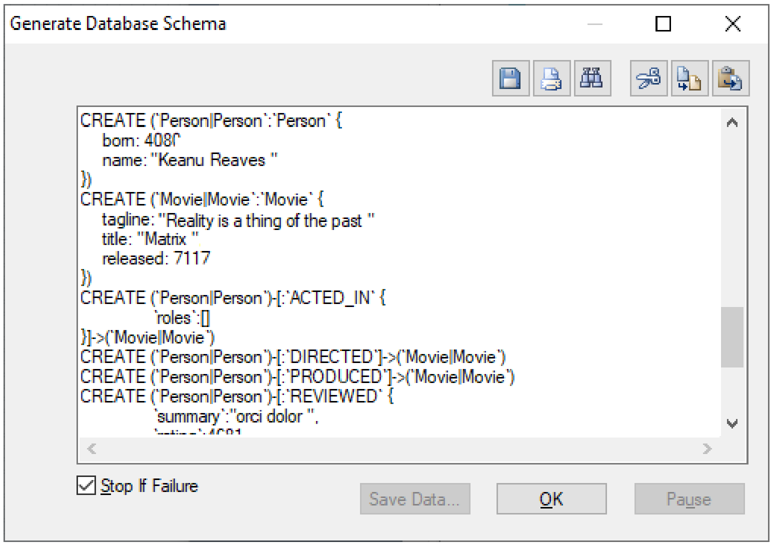


Name: MODEL_1
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 0 □ Global Indexes: 0 □ Views: 0
□ Nodes: 2 □ Relationships: 6 □ Label: 2
□ Fields: 5 □ Sub-Categories: 0 □ Databases: 1

Style: Torus Style (3D) Pallete: Illustration □ Legend

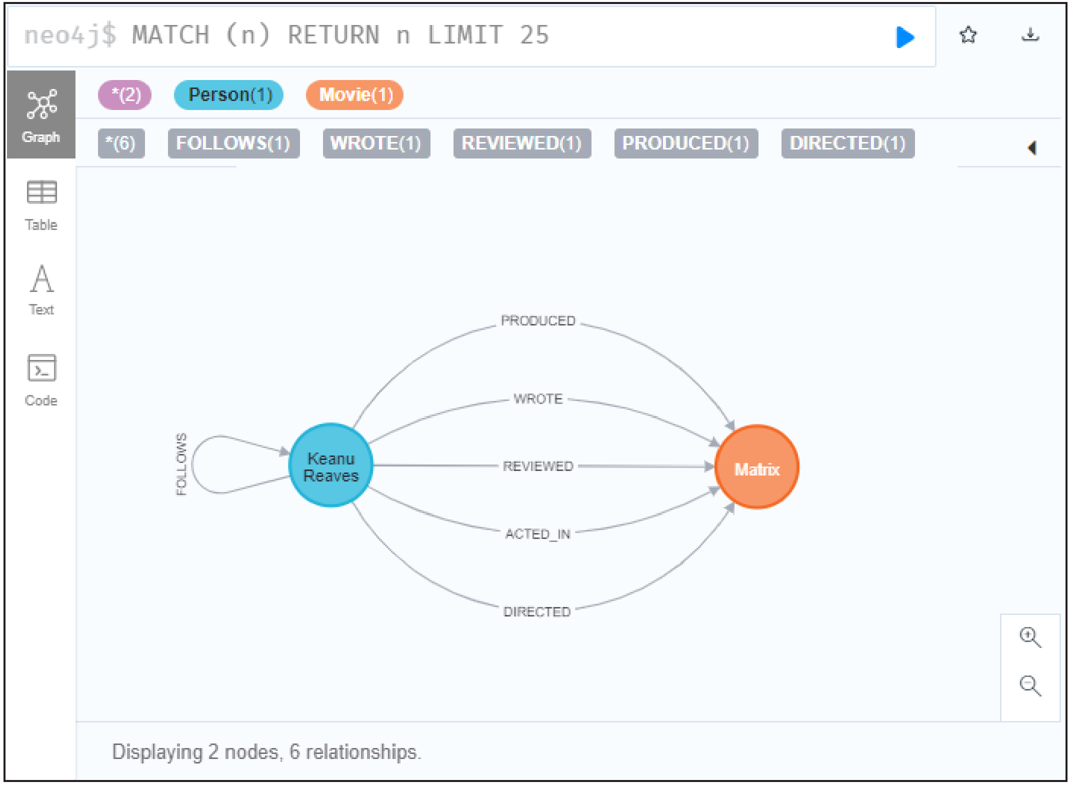
On forward engineering, the following script was generated:



```
CREATE (`Person|Person`:`Person` {
    bom: 408ſ
    name: "Keanu Reaves "
})
CREATE (`Movie|Movie`:`Movie` {
    tagline: "Reality is a thing of the past "
    title: "Matrix "
    released: 7117
})
CREATE (`Person|Person`)-[:`ACTED_IN` {
            `roles`:[]
}]->(`Movie|Movie`)
CREATE (`Person|Person`)-[:`DIRECTED`]->(`Movie|Movie`)
CREATE (`Person|Person`)-[:`PRODUCED`]->(`Movie|Movie`)
CREATE (`Person|Person`)-[:`REVIEWED` {
            `summary`:"orci dolor ",
```

☑ Stop If Failure

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
```

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. **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.

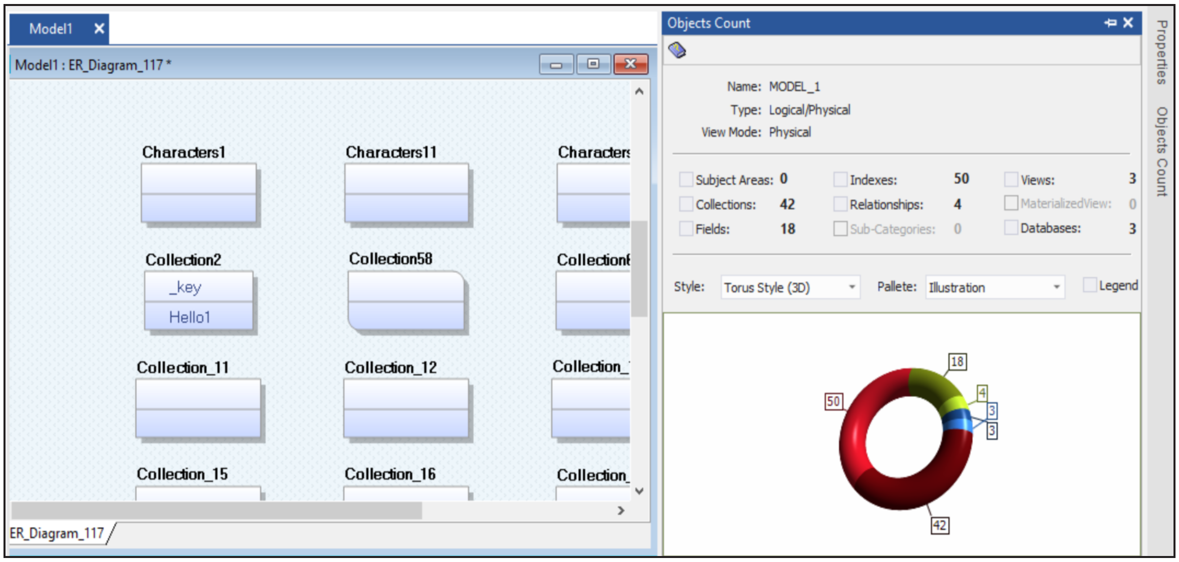
This topic walks you through the steps to compare an Neo4j model with database.

To compare models with database:

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.

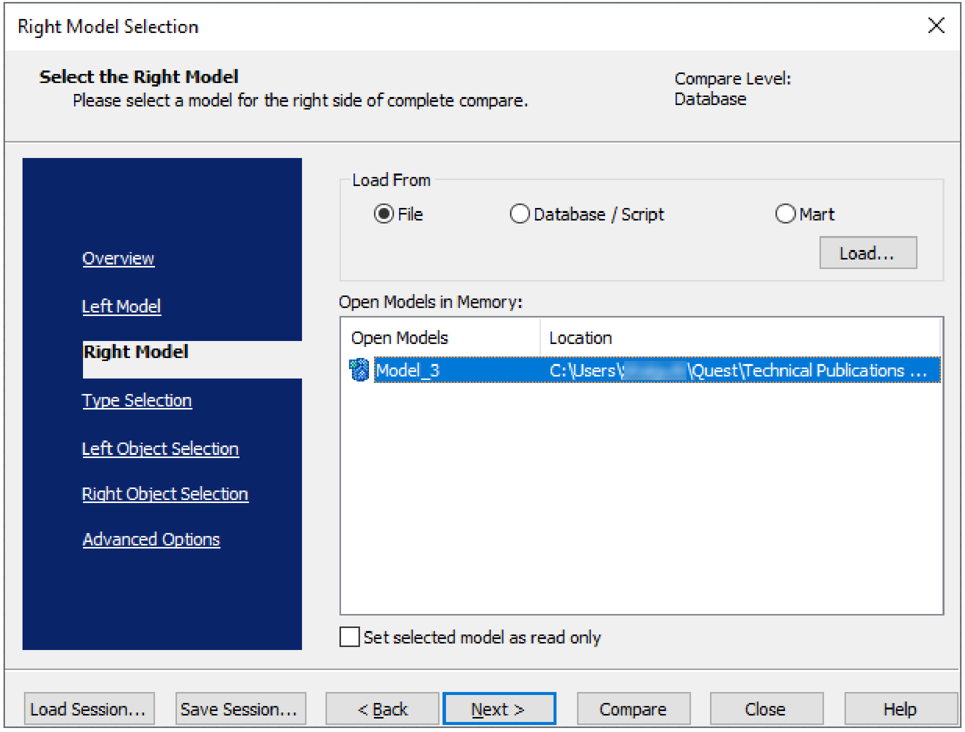


Name: MODEL_1
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	0	□ Indexes:	50	□ Views:	3
□ Collections:	42	□ Relationships:	4	□ MaterializedView:	0
□ Fields:	18	□ Sub-Categories:	0	□ Databases:	3

Style: Torus Style (3D) Pallete: Illustration □ Legend

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.



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level: Database

Load From

● File ○ Database / Script ○ Mart

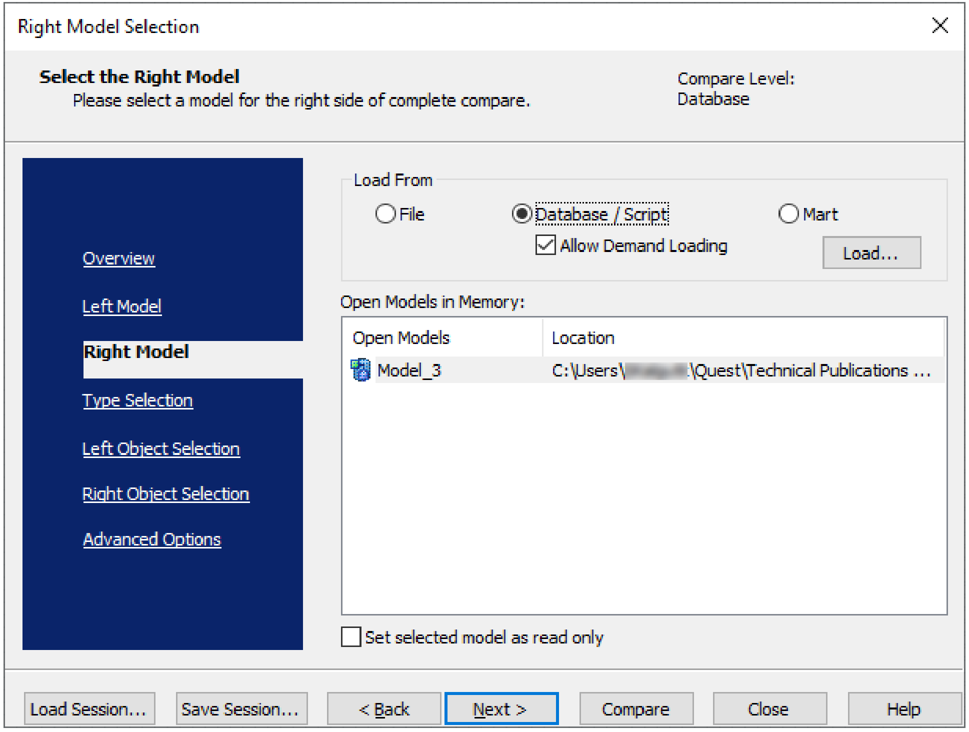
Open Models in Memory:

Open Models	Location
Model_3	C:\Users\[redacted]\Quest\Technical Publications ...

□ Set selected model as read only

3. Click **Database/Script**.

   By default, the Allow Demand Loading option is selected.



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level: Database

Load From: ○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

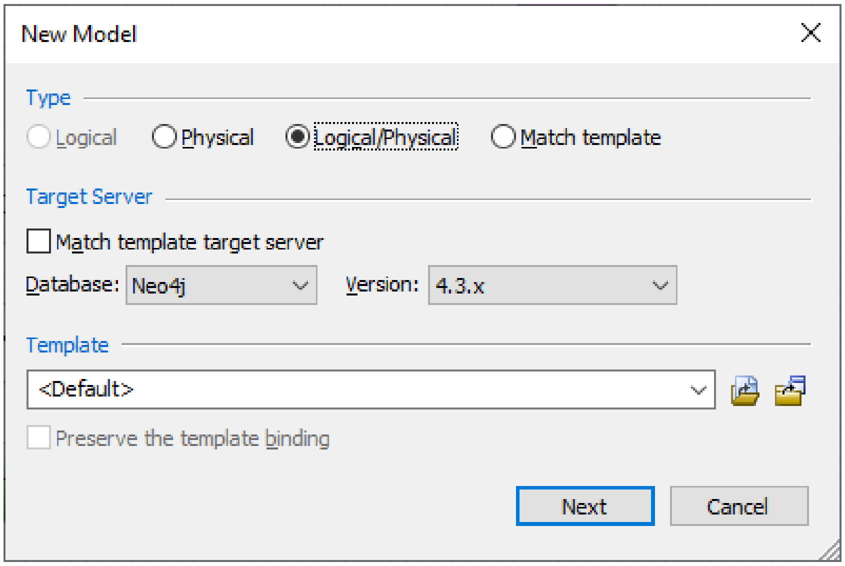
Open Models in Memory:

Open Models	Location
Model_3	C:\Users\[redacted]\Quest\Technical Publications ...

□ Set selected model as read only

4. Click **Load**.

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



New Model

Type: ○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

Database: Neo4j

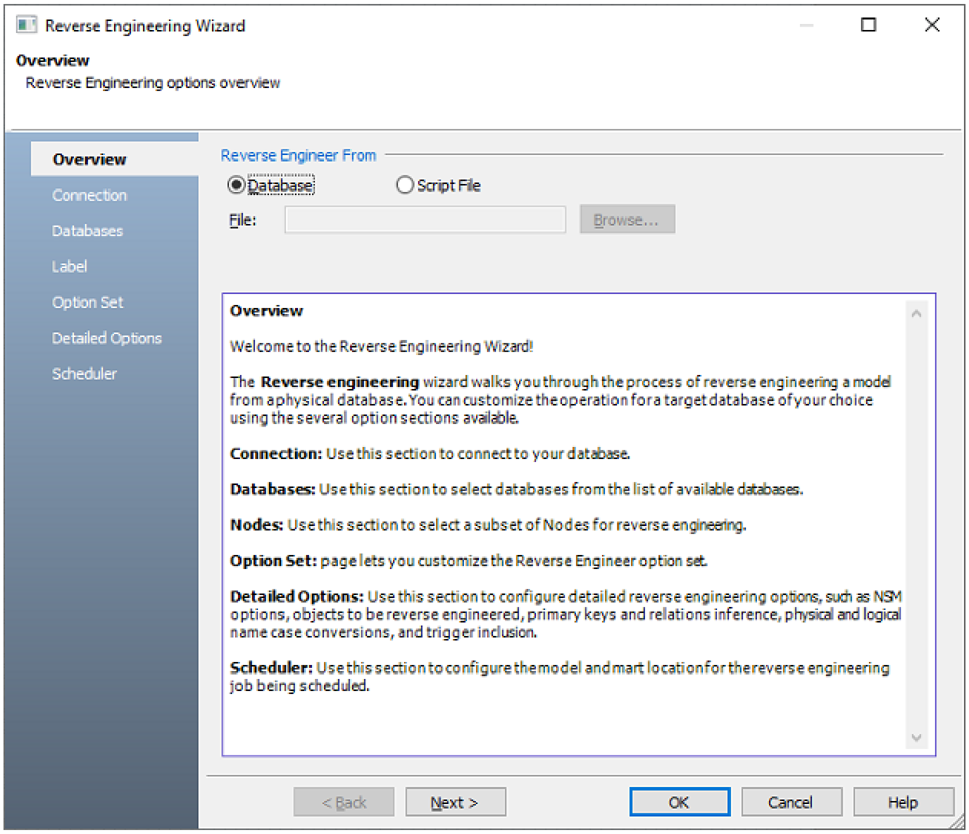
Version: 4.3.x

Template: <Default>

□ Preserve the template binding

5. Ensure that the Database is set to Neo4j. Then, click **Next**.

   The Reverse Engineering Wizard appears.



● Database ○ Script File

File:

Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Databases: Use this section to select databases from the list of available databases.

Nodes: Use this section to select a subset of Nodes for reverse engineering.

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

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

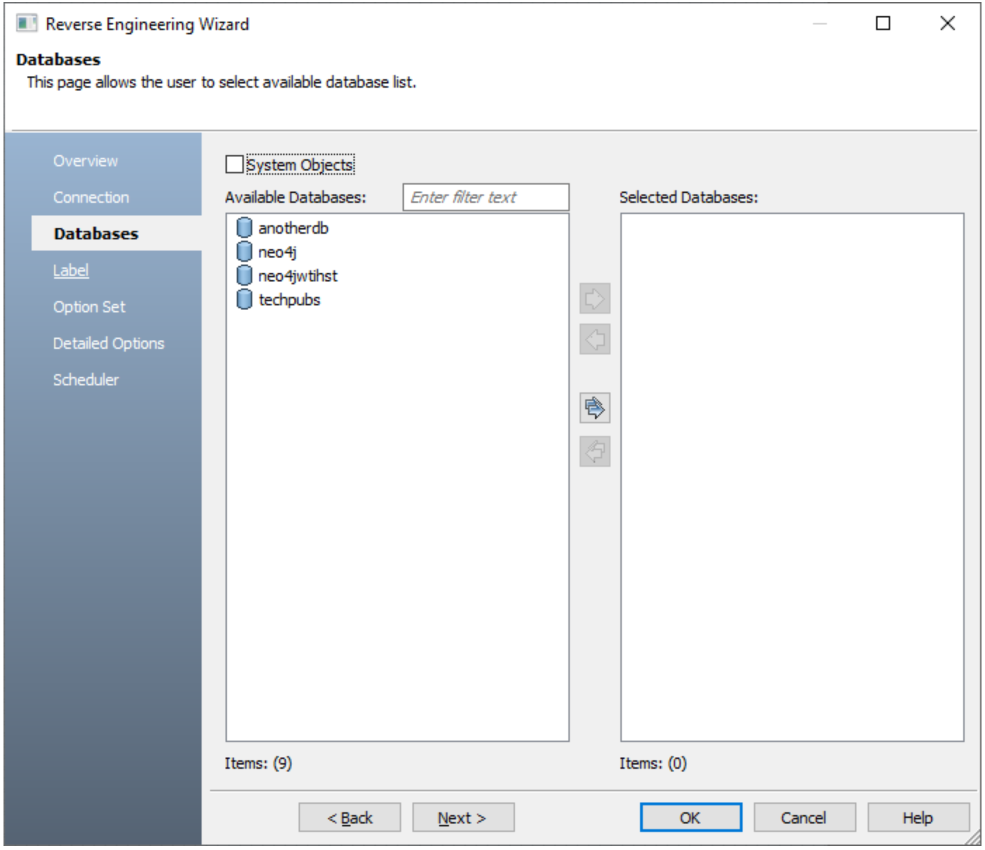
Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

6. Click **Database**. Then, click **Next**.

   The Connection section appears. Use this section to connect to the database from which you want to reverse engineer the model.

7. After connection is established, click **Next**.

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



Reverse Engineering Wizard

Databases

This page allows the user to select available database list.

□ System Objects

Available Databases: Enter filter text

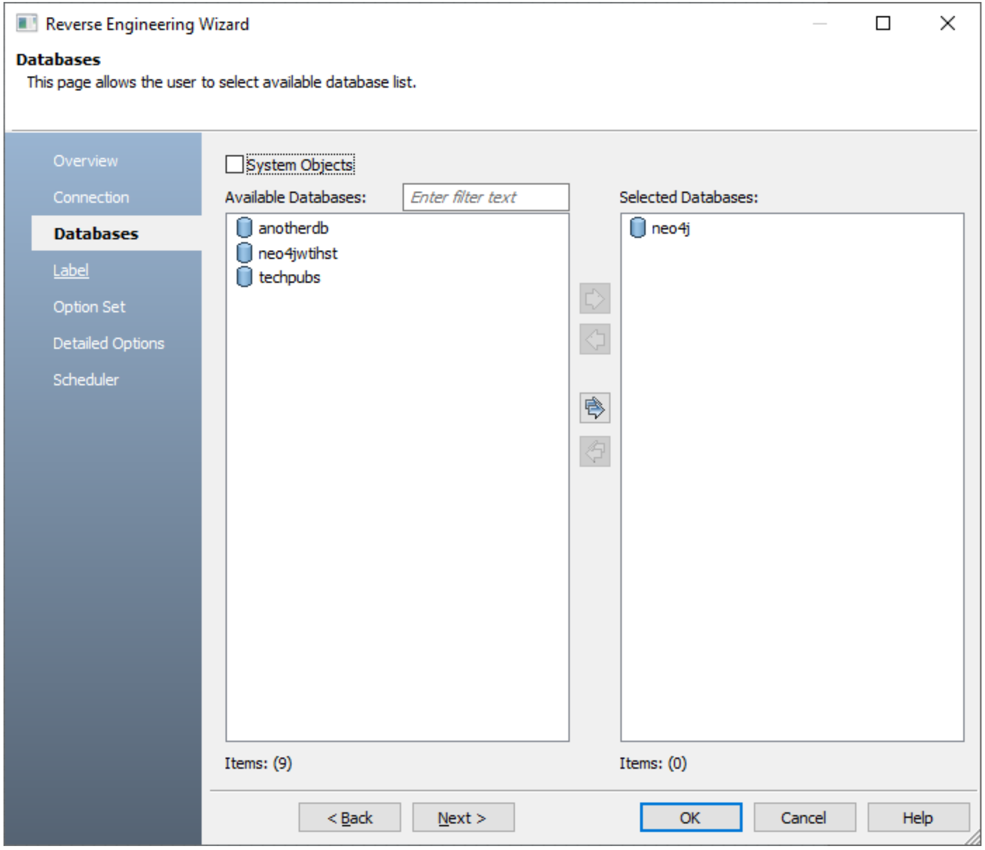
- anotherdb
- neo4j
- neo4jwtihst
- techpubs

Items: (9)

Selected Databases:

Items: (0)

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



Reverse Engineering Wizard

Databases

This page allows the user to select available database list.

□ System Objects

Available Databases: Enter filter text

- anotherdb
- neo4jwtihst
- techpubs

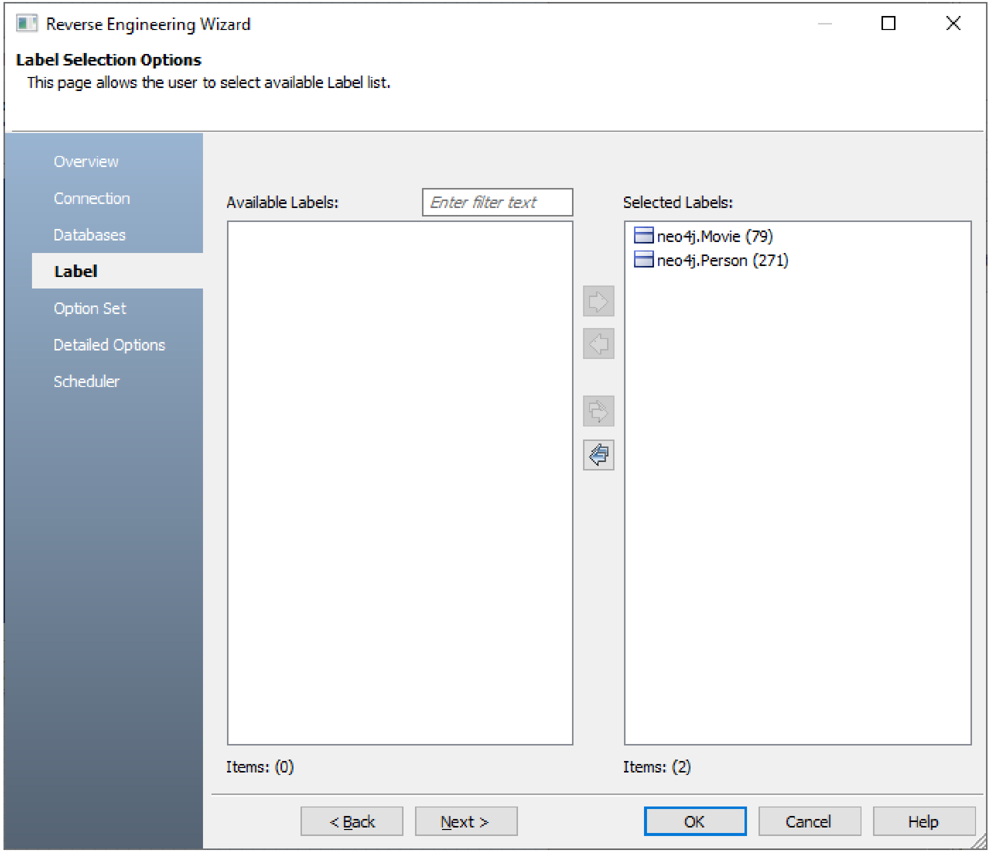
Items: (9)

Selected Databases:

- neo4j

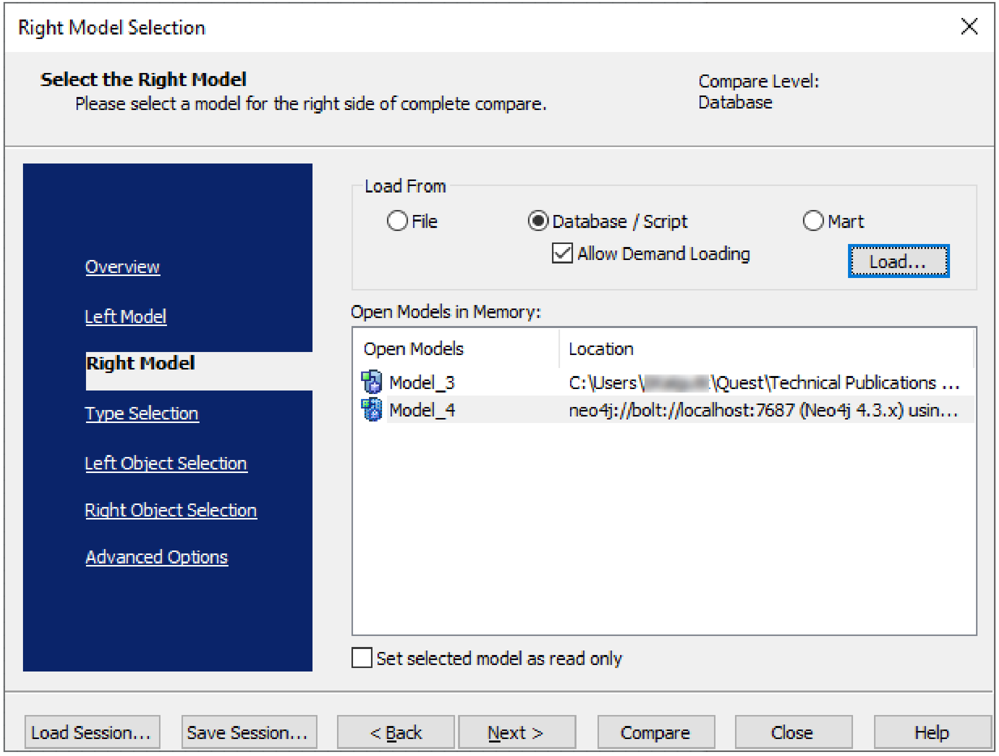
Items: (0)

9. Click **Next** and in the Label section, click ⇒.
   This selects all the available labels.



10. Click **Next** and in the Option Set section, keep the default configuration.
11. Click **Next** and in the Detail Options section, keep the default configuration.
12. Click **OK**.

    The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse engineered.



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

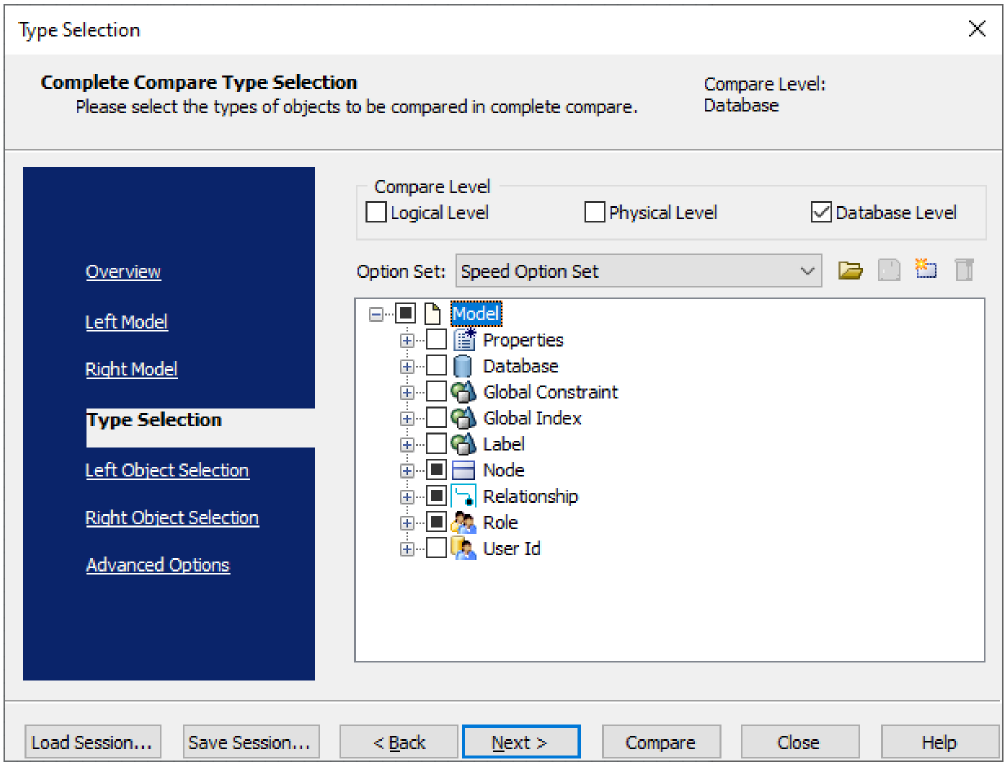
Open Models in Memory:

Open Models	Location
Model_3	C:\Users\[redacted]\Quest\Technical Publications ...
Model_4	neo4j://bolt://localhost:7687 (Neo4j 4.3.x) usin...

□ Set selected model as read only

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

For example, the following image shows the default options.



Type Selection

Complete Compare Type Selection

Please select the types of objects to be compared in complete compare.

Compare Level:
Database

Compare Level

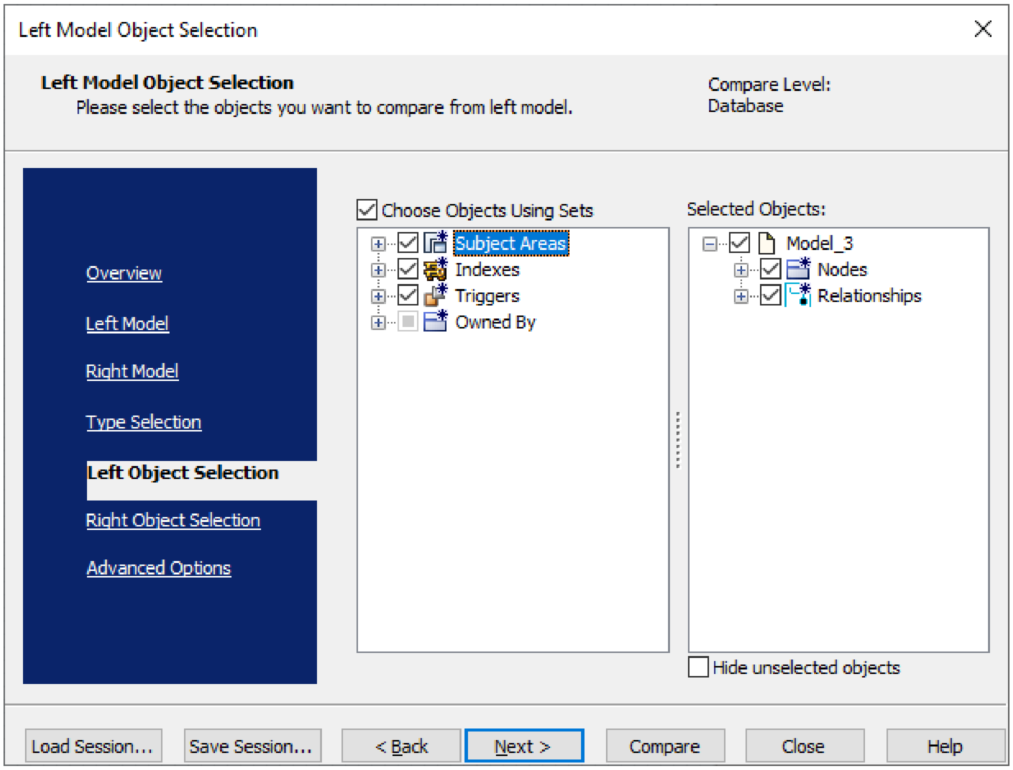
□ Logical Level □ Physical Level ☑ Database Level

Option Set: Speed Option Set

- ■ Model
  - □ Properties
  - □ Database
  - □ Global Constraint
  - □ Global Index
  - □ Label
  - ■ Node
  - ■ Relationship
  - ■ Role
  - □ User Id

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

For example, the following image shows the default options.



Left Model Object Selection

Left Model Object Selection

Please select the objects you want to compare from left model.

Compare Level:
Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

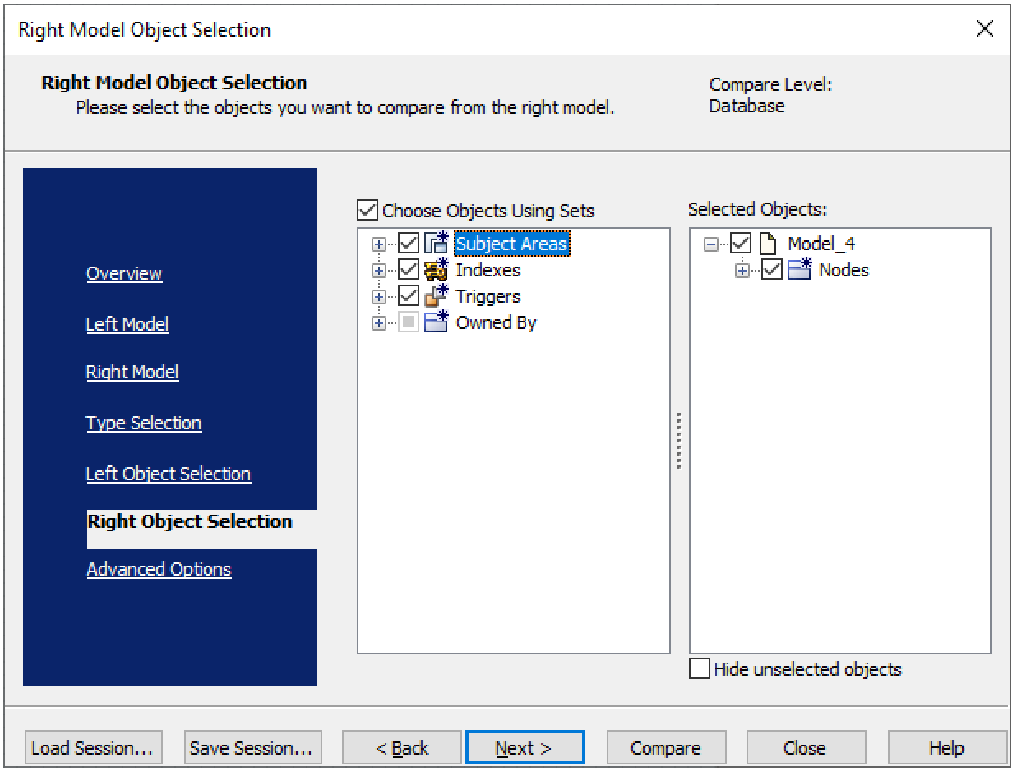
Selected Objects:

- ☑ Model_3
  - ☑ Nodes
  - ☑ Relationships

□ Hide unselected objects

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

For example, the following image shows the default options.



Right Model Object Selection

Right Model Object Selection

Please select the objects you want to compare from the right model.

Compare Level: Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

Selected Objects:

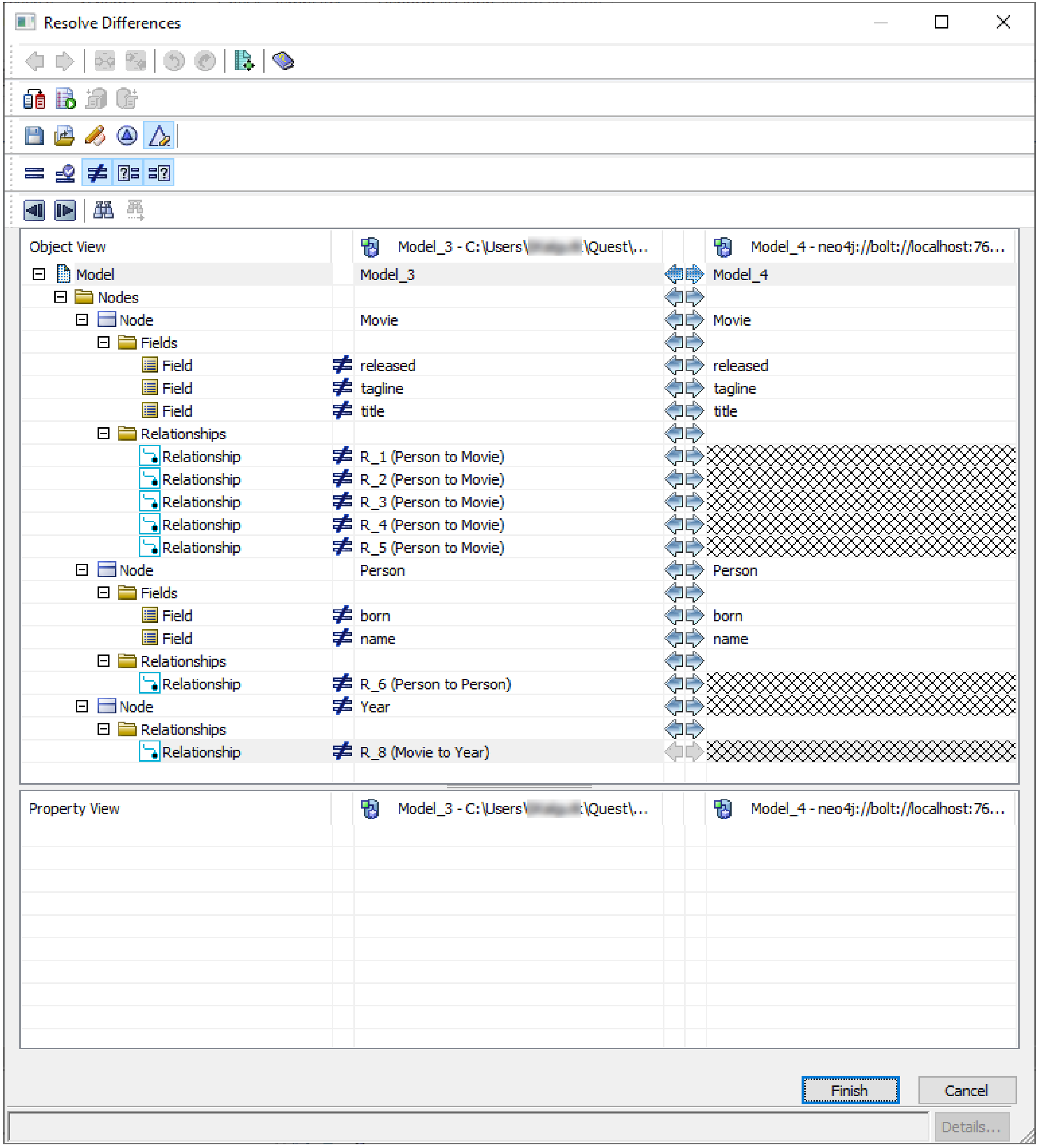
- ☑ Model_4
  - ☑ Nodes

□ Hide unselected objects

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.



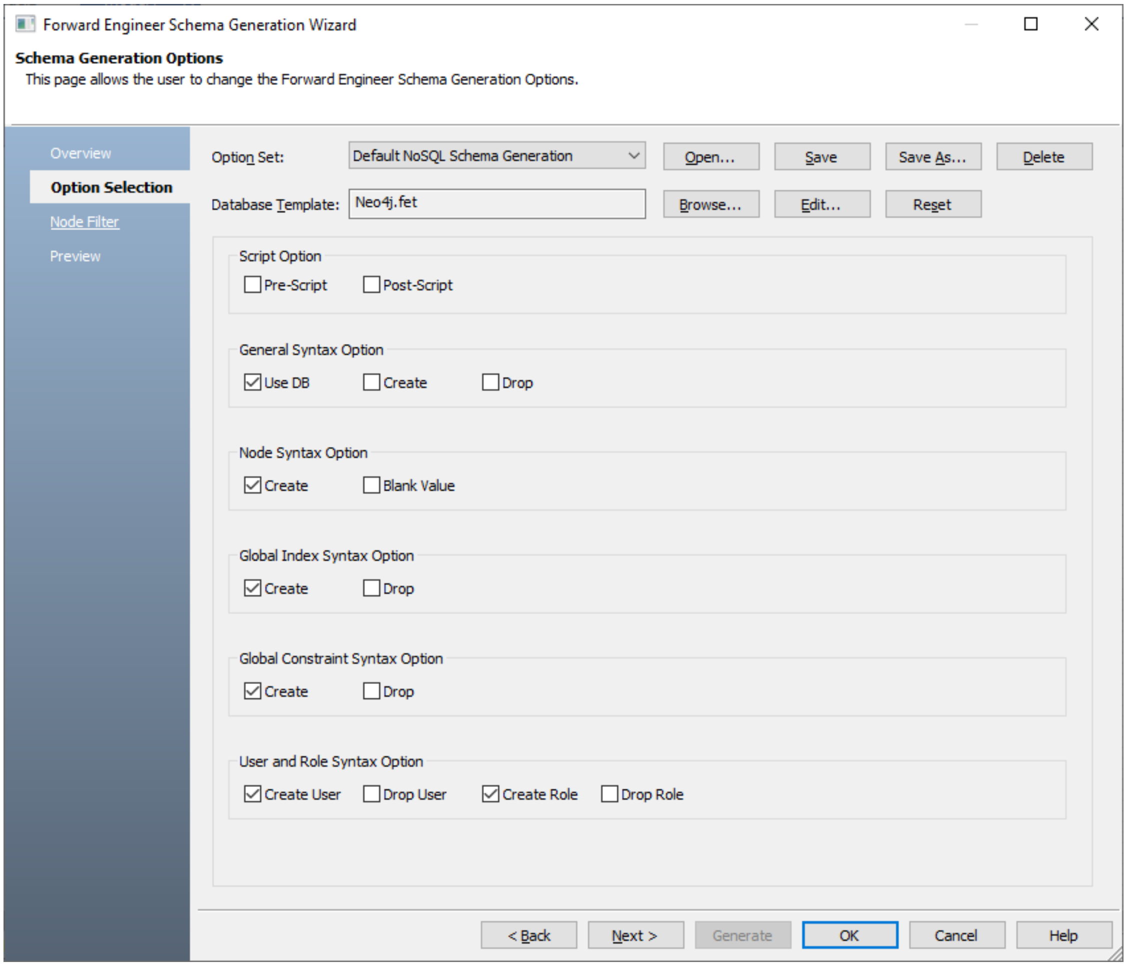
Object View		Model_3 - C:\Users\[redacted]\Quest\...	Model_4 - neo4j://bolt://localhost:76...
Model		Model_3	Model_4
Nodes			
Node		Movie	Movie
Fields			
Field	≠	released	released
Field	≠	tagline	tagline
Field	≠	title	title
Relationships			
Relationship	≠	R_1 (Person to Movie)	
Relationship	≠	R_2 (Person to Movie)	
Relationship	≠	R_3 (Person to Movie)	
Relationship	≠	R_4 (Person to Movie)	
Relationship	≠	R_5 (Person to Movie)	
Node		Person	Person
Fields			
Field	≠	born	born
Field	≠	name	name
Relationships			
Relationship	≠	R_6 (Person to Person)	
Node	≠	Year	
Relationships			
Relationship	≠	R_8 (Movie to Year)	

Property View	Model_3 - C:\Users\[redacted]\Quest\...	Model_4 - neo4j://bolt://localhost:76...

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.



Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default NoSQL Schema Generation

Database Template: Neo4j.fet

Script Option

□ Pre-Script □ Post-Script

General Syntax Option

☑ Use DB □ Create □ Drop

Node Syntax Option

☑ Create □ Blank Value

Global Index Syntax Option

☑ Create □ Drop

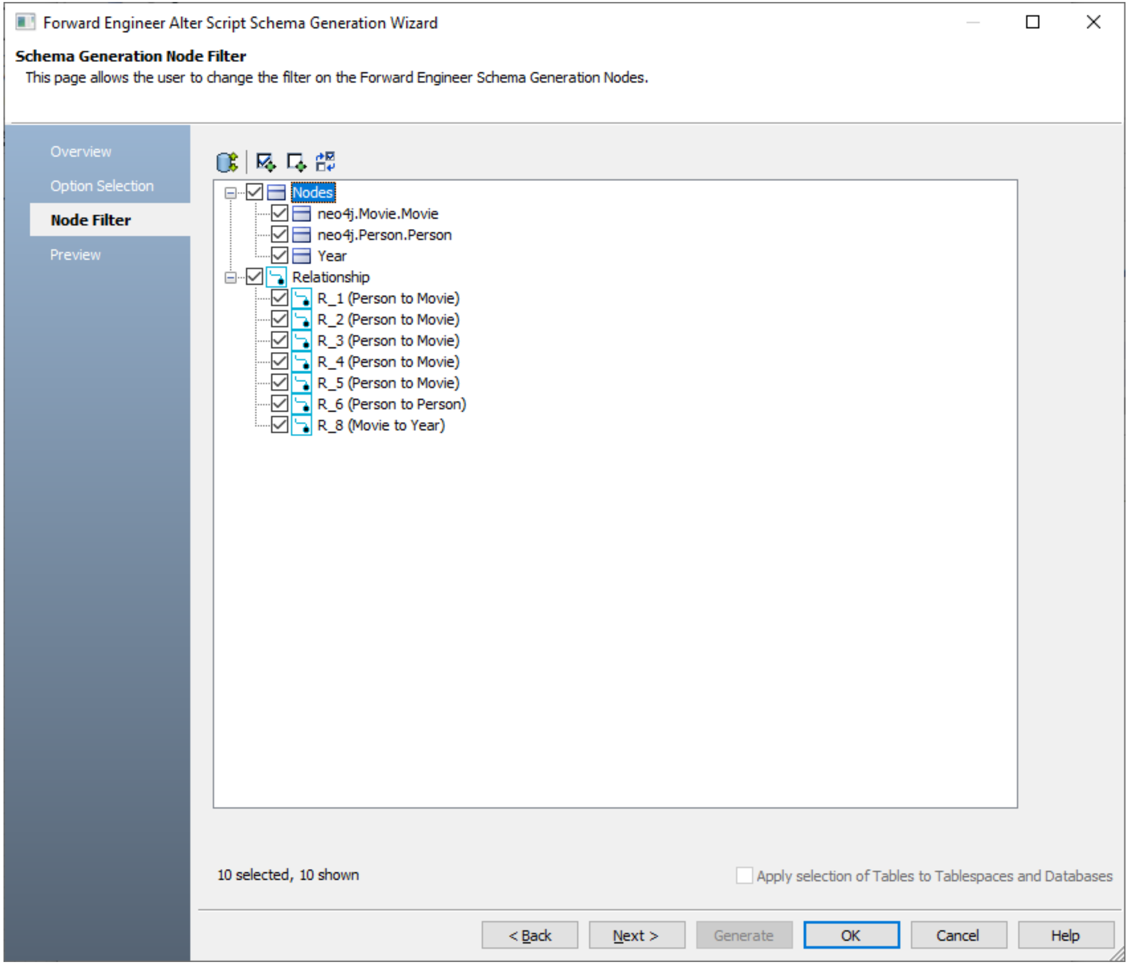
Global Constraint Syntax Option

☑ Create □ Drop

User and Role Syntax Option

☑ Create User □ Drop User ☑ Create Role □ Drop Role

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



Forward Engineer Alter Script Schema Generation Wizard

Schema Generation Node Filter

This page allows the user to change the filter on the Forward Engineer Schema Generation Nodes.

- ☑ Nodes
  - ☑ neo4j.Movie.Movie
  - ☑ neo4j.Person.Person
  - ☑ Year
- ☑ Relationship
  - ☑ R_1 (Person to Movie)
  - ☑ R_2 (Person to Movie)
  - ☑ R_3 (Person to Movie)
  - ☑ R_4 (Person to Movie)
  - ☑ R_5 (Person to Movie)
  - ☑ R_6 (Person to Person)
  - ☑ R_8 (Movie to Year)

10 selected, 10 shown

□ Apply selection of Tables to Tablespaces and Databases

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.

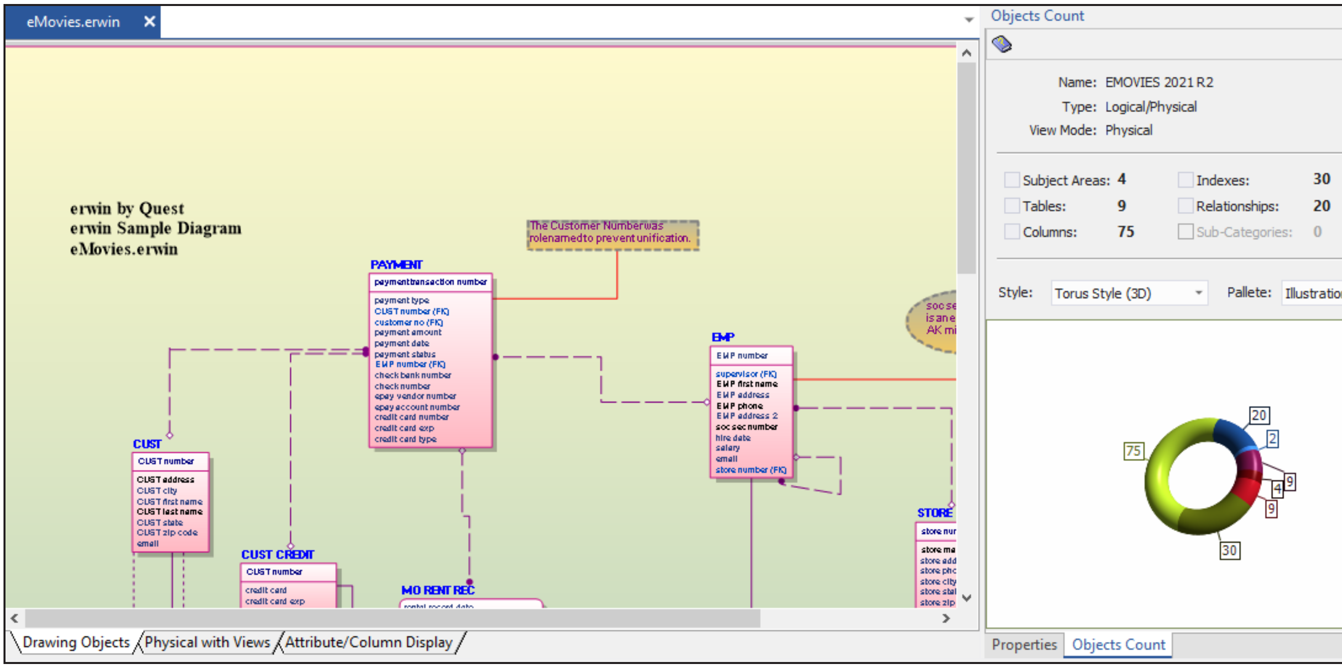
Migration by Changing the Target Database

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.



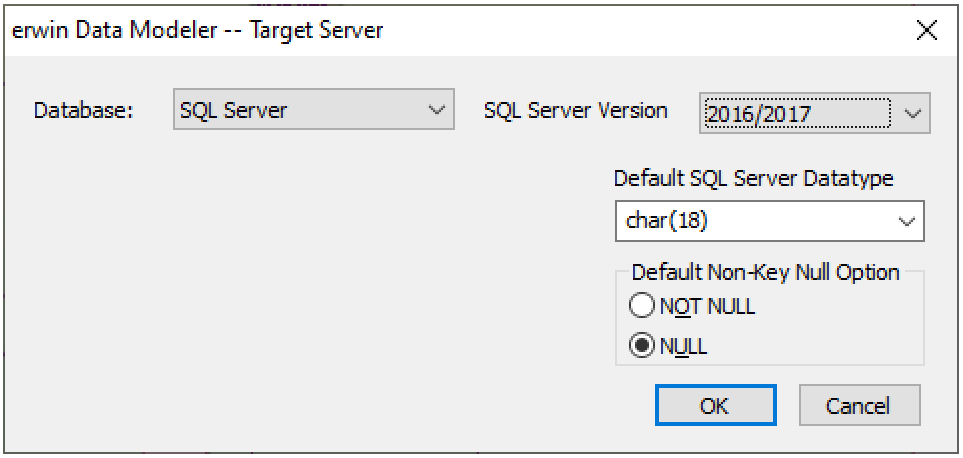
Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

☐ Subject Areas: 4 ☐ Indexes: 30
☐ Tables: 9 ☐ Relationships: 20
☐ Columns: 75 ☐ Sub-Categories: 0

Style: Torus Style (3D) Pallete: Illustratio

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.



erwin Data Modeler -- Target Server

Database: SQL Server

SQL Server Version: 2016/2017

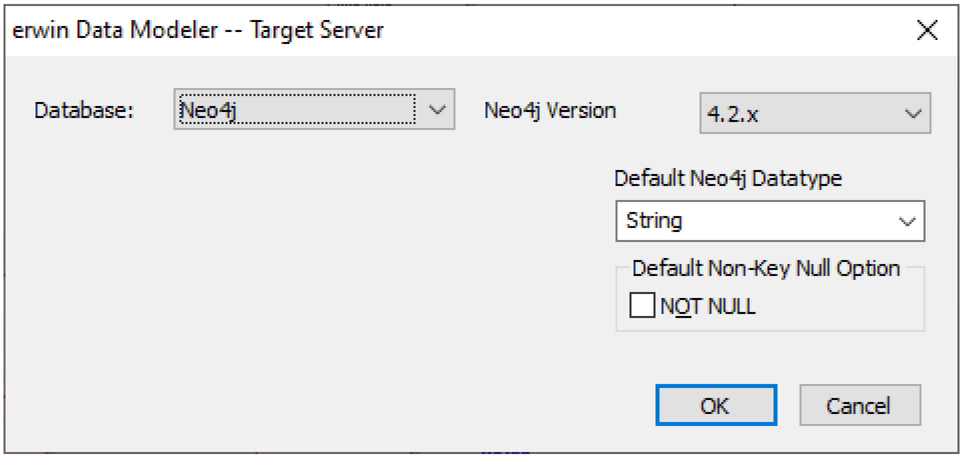
Default SQL Server Datatype: char(18)

Default Non-Key Null Option

○ NOT NULL

● NULL

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



erwin Data Modeler -- Target Server

Database: Neo4j

Neo4j Version: 4.2.x

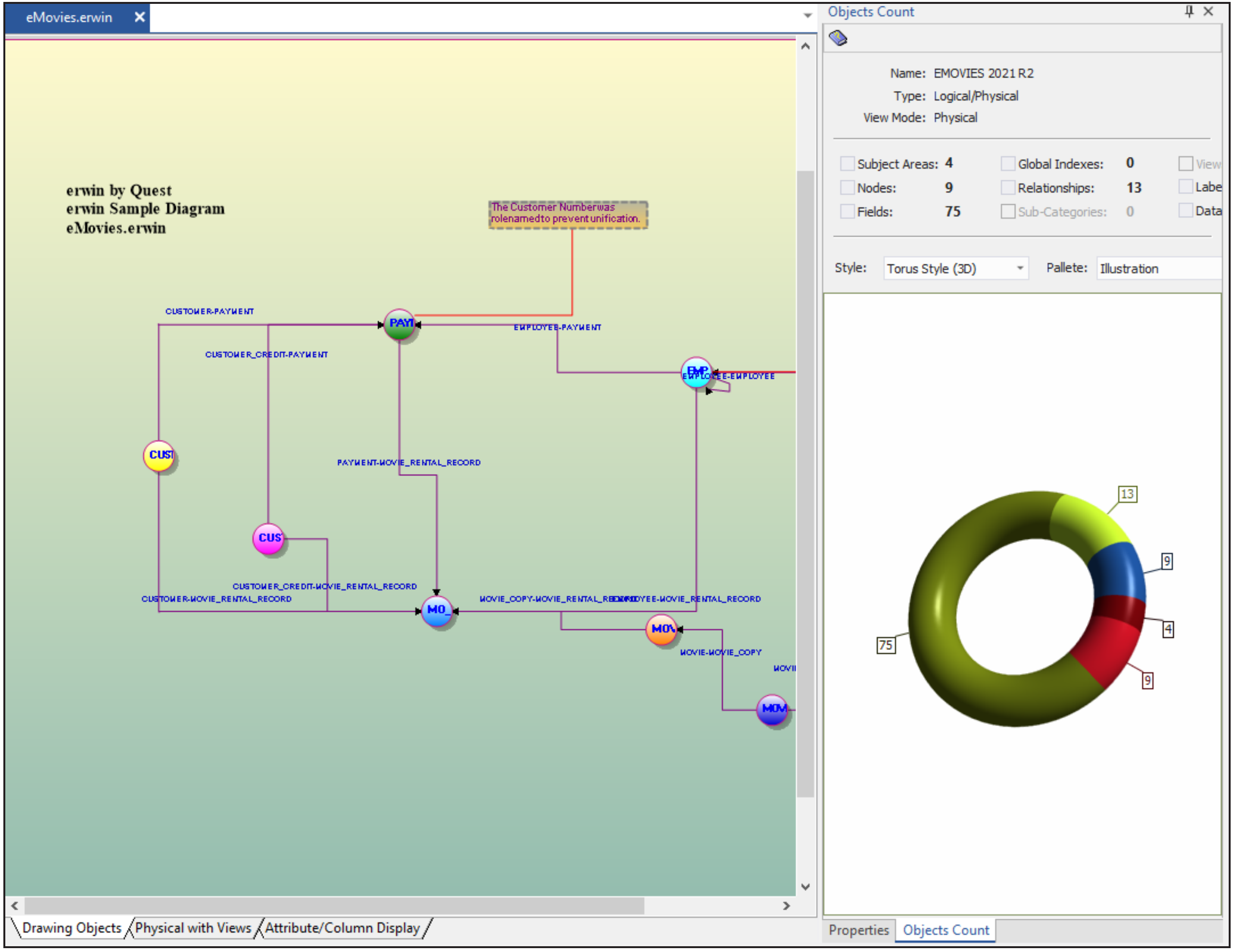
Default Neo4j Datatype: String

Default Non-Key Null Option

□ NOT NULL

4. Click **OK**.

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



Objects Count

Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 4	□ Global Indexes: 0	□ View
□ Nodes: 9	□ Relationships: 13	□ Labe
□ Fields: 75	□ Sub-Categories: 0	□ Data

Style: Torus Style (3D) Pallete: Illustration

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

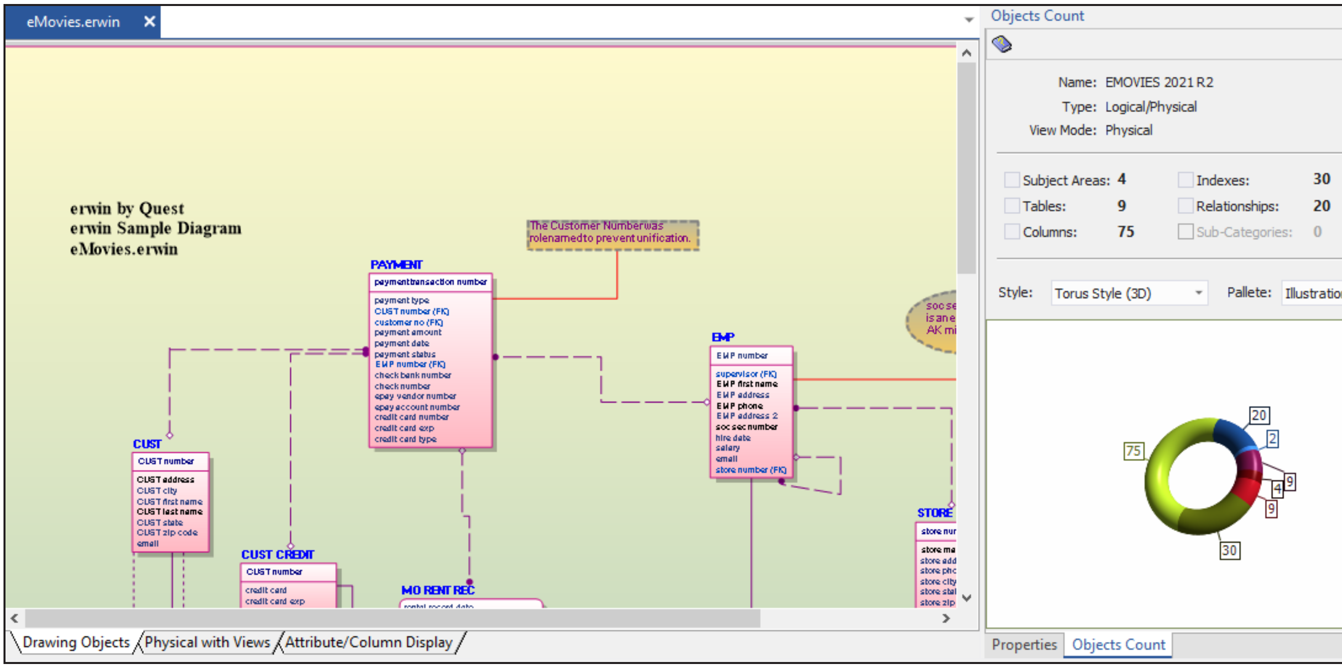
To migrate by deriving a model, follow these steps:

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.



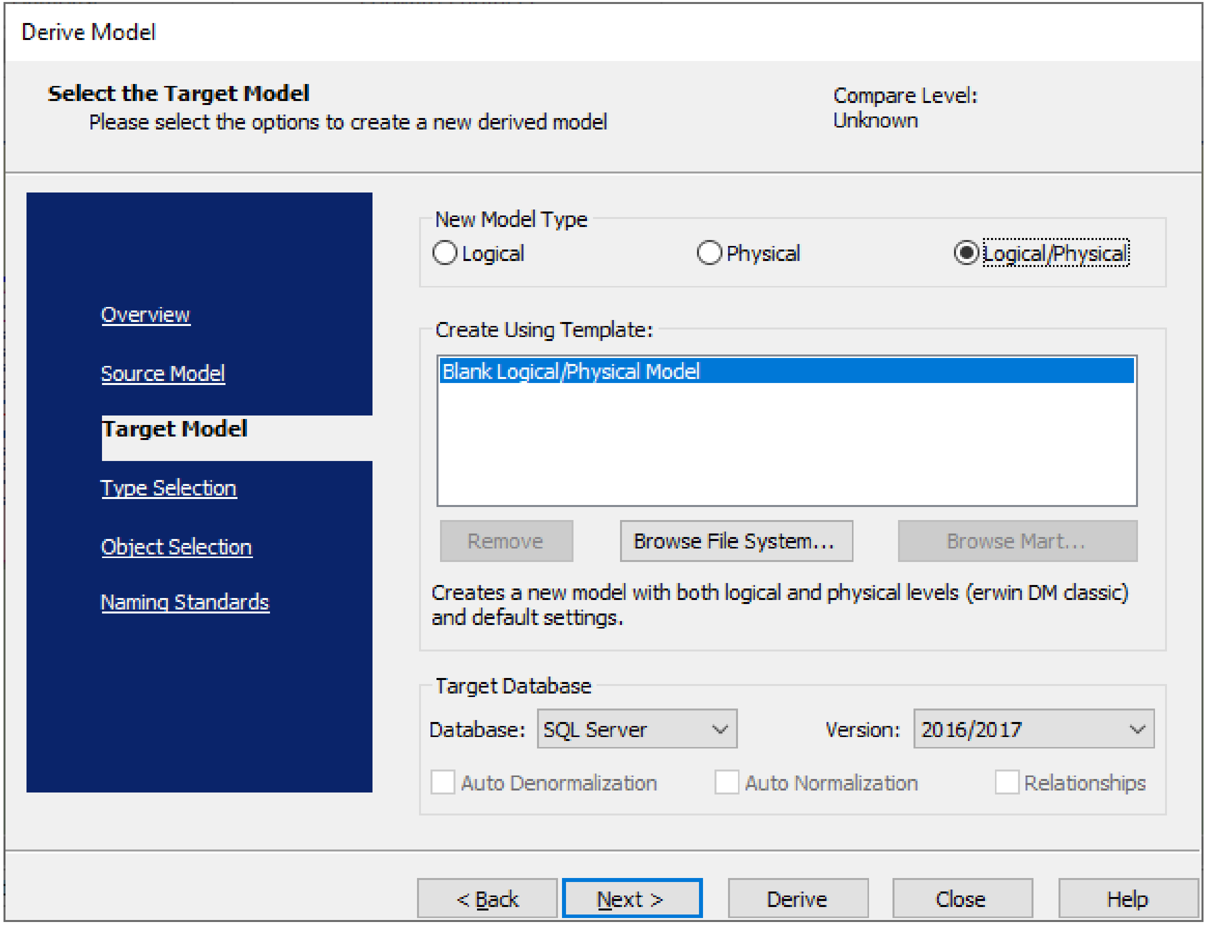
Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	4	□ Indexes:	30
□ Tables:	9	□ Relationships:	20
□ Columns:	75	□ Sub-Categories:	0

Style: Torus Style (3D)　Pallete: Illustratio

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

New Model Type

○ Logical ○ Physical ● Logical/Physical

Create Using Template:

Blank Logical/Physical Model

Creates a new model with both logical and physical levels (erwin DM classic) and default settings.

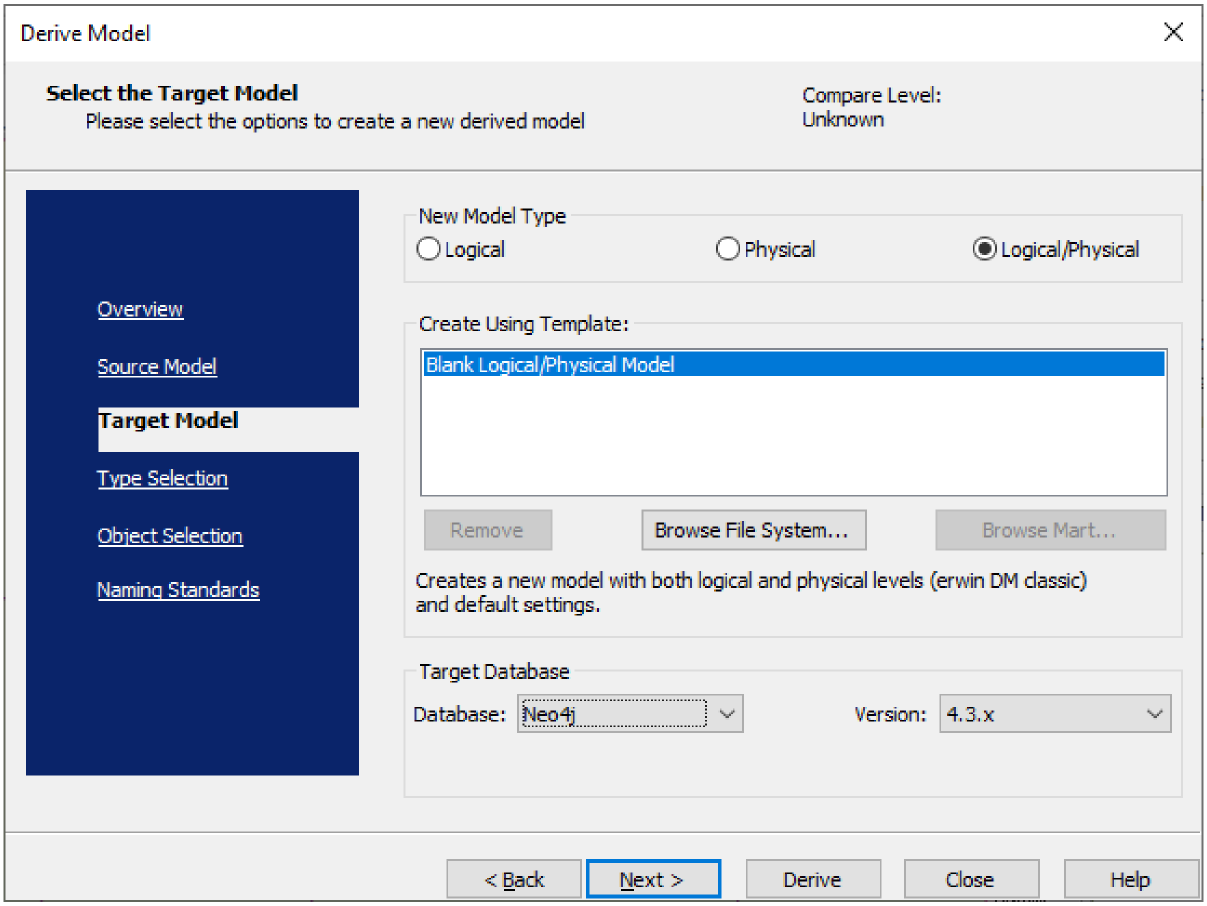
Target Database

Database: SQL Server

Version: 2016/2017

□ Auto Denormalization □ Auto Normalization □ Relationships

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



Derive Model

Select the Target Model
Please select the options to create a new derived model

Compare Level:
Unknown

New Model Type
○ Logical ○ Physical ● Logical/Physical

Create Using Template:
Blank Logical/Physical Model

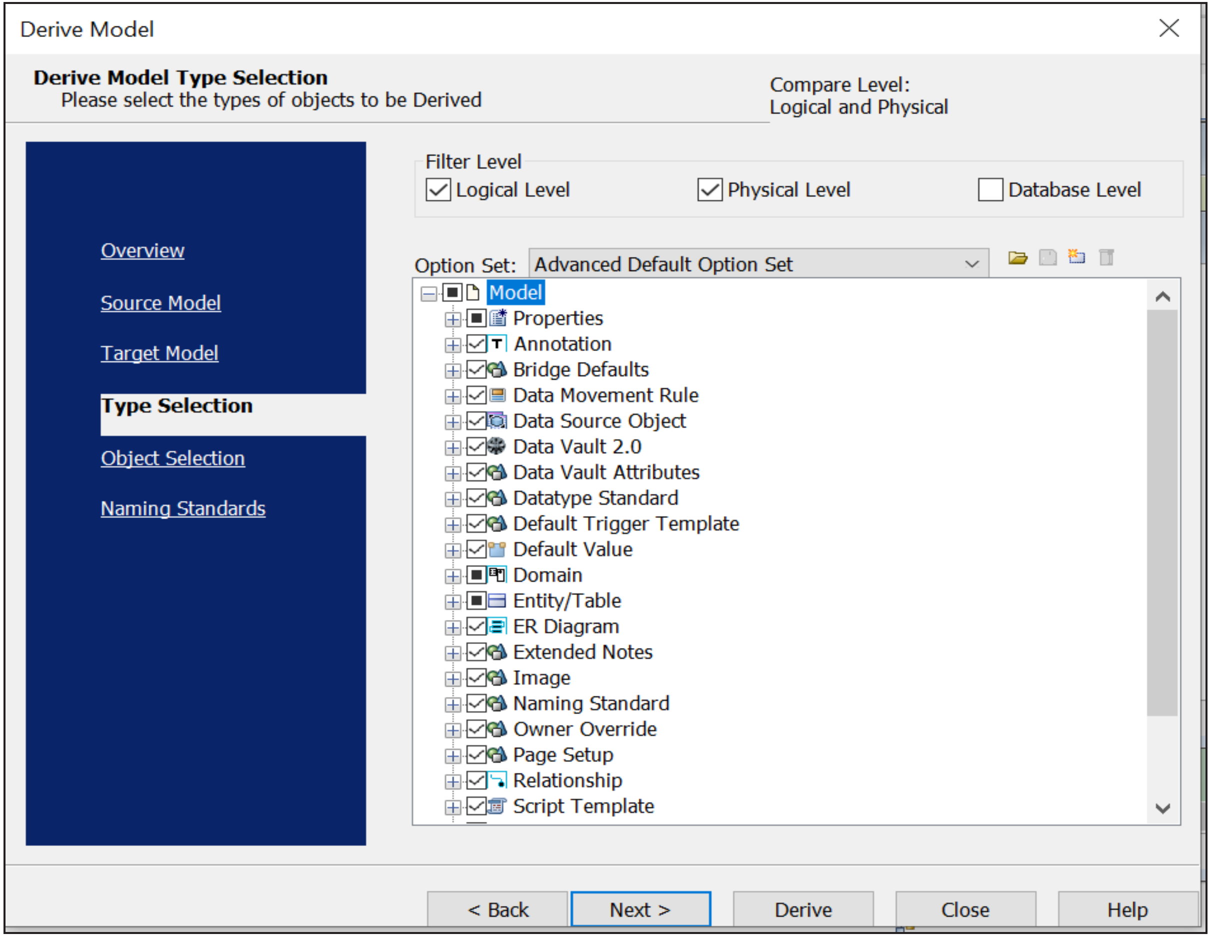
Creates a new model with both logical and physical levels (erwin DM classic) and default settings.

Target Database
Database: Neo4j
Version: 4.3.x

4. Click **Next**.

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

The Type Selection section appears.



Derive Model

Derive Model Type Selection
Please select the types of objects to be Derived

Compare Level:
Logical and Physical

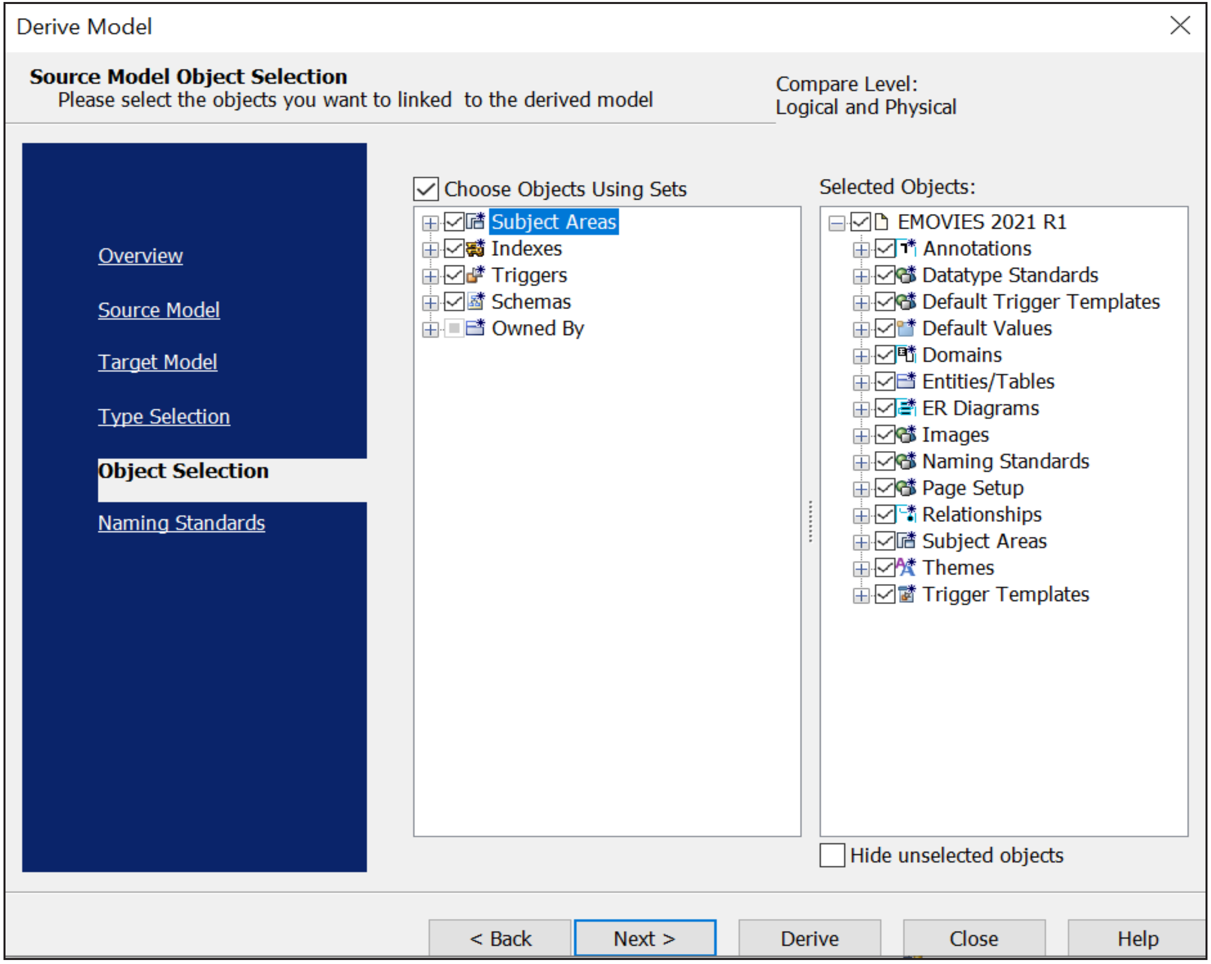
Filter Level
☑ Logical Level ☑ Physical Level □ Database Level

Option Set: Advanced Default Option Set

- ■ Model
  - ■ Properties
  - ☑ Annotation
  - ☑ Bridge Defaults
  - ☑ Data Movement Rule
  - ☑ Data Source Object
  - ☑ Data Vault 2.0
  - ☑ Data Vault Attributes
  - ☑ Datatype Standard
  - ☑ Default Trigger Template
  - ☑ Default Value
  - ■ Domain
  - ■ Entity/Table
  - ☑ ER Diagram
  - ☑ Extended Notes
  - ☑ Image
  - ☑ Naming Standard
  - ☑ Owner Override
  - ☑ Page Setup
  - ☑ Relationship
  - ☑ Script Template

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



Derive Model

Source Model Object Selection
Please select the objects you want to linked to the derived model

Compare Level:
Logical and Physical

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- ☑ Schemas
- □ Owned By

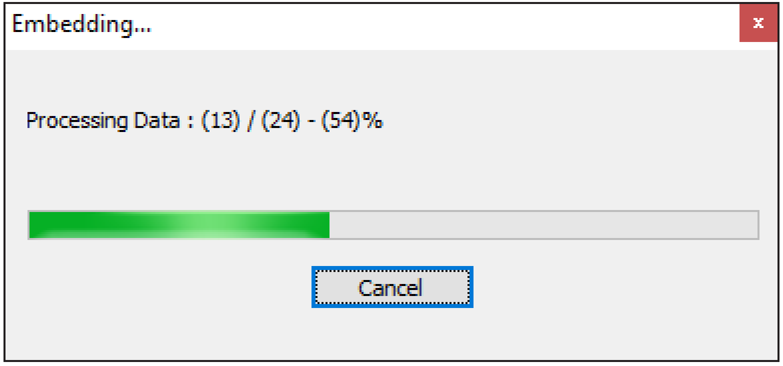
Selected Objects:

- ☑ EMOVIES 2021 R1
  - ☑ Annotations
  - ☑ Datatype Standards
  - ☑ Default Trigger Templates
  - ☑ Default Values
  - ☑ Domains
  - ☑ Entities/Tables
  - ☑ ER Diagrams
  - ☑ Images
  - ☑ Naming Standards
  - ☑ Page Setup
  - ☑ Relationships
  - ☑ Subject Areas
  - ☑ Themes
  - ☑ Trigger Templates

□ Hide unselected objects

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

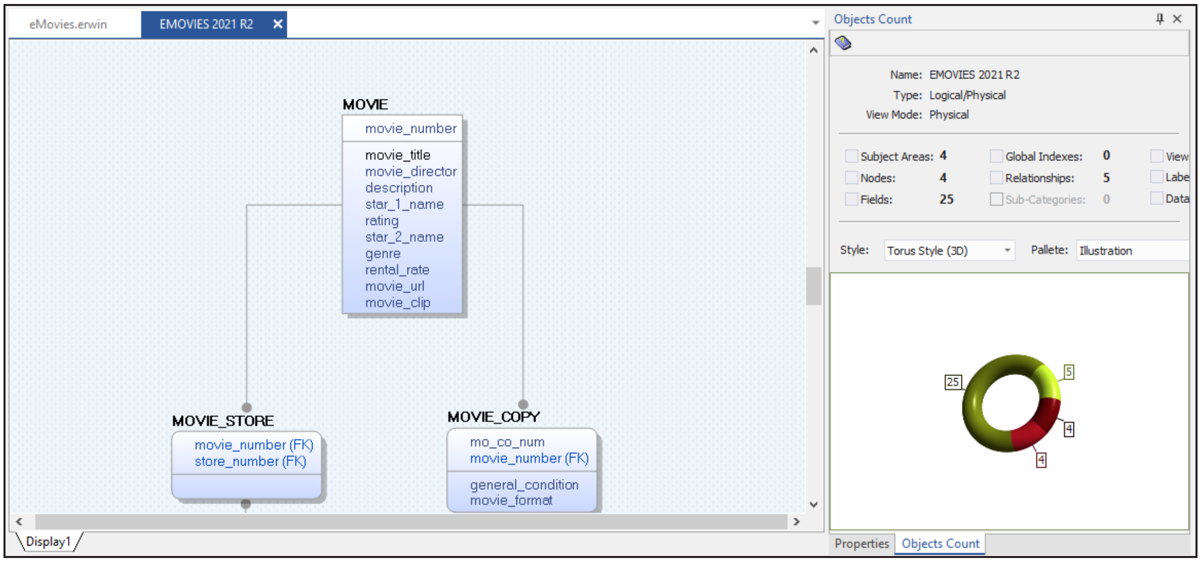
8. Click **Derive**.
   The model derivation process starts.



Embedding...

Processing Data : (13) / (24) - (54)%

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



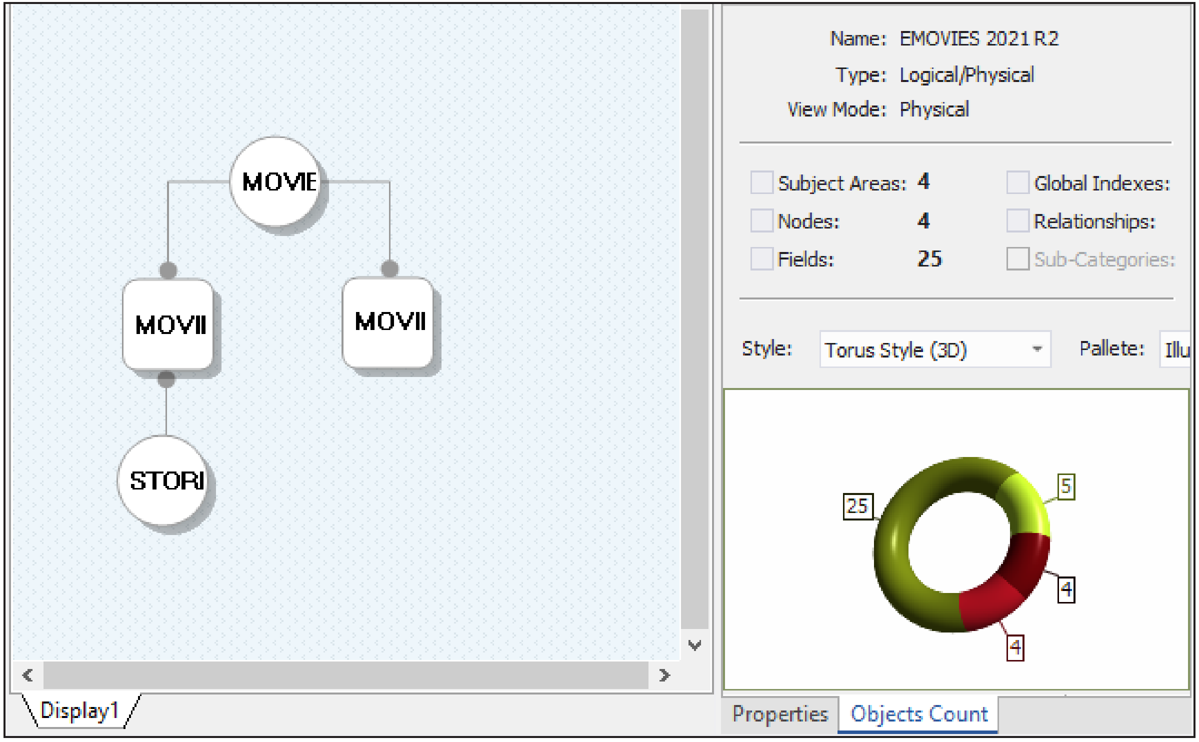
Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	4	□ Global Indexes:	0	□ View	
□ Nodes:	4	□ Relationships:	5	□ Labe	
□ Fields:	25	□ Sub-Categories:	0	□ Data	

Style: Torus Style (3D) Pallete: Illustration

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:



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

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.

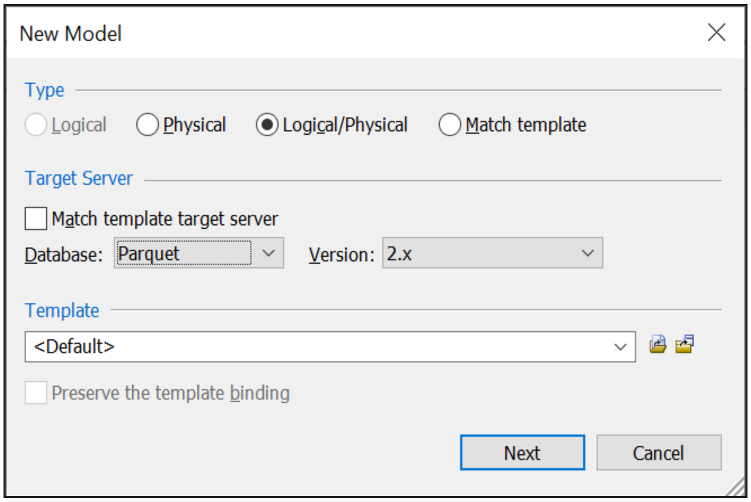
This topic walks you through the steps to reverse engineer a Parquet model. For detailed description of reverse engineering options, refer to the Reverse Engineering Options topic.

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**.



New Model

Type

○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

Database: Parquet Version: 2.x

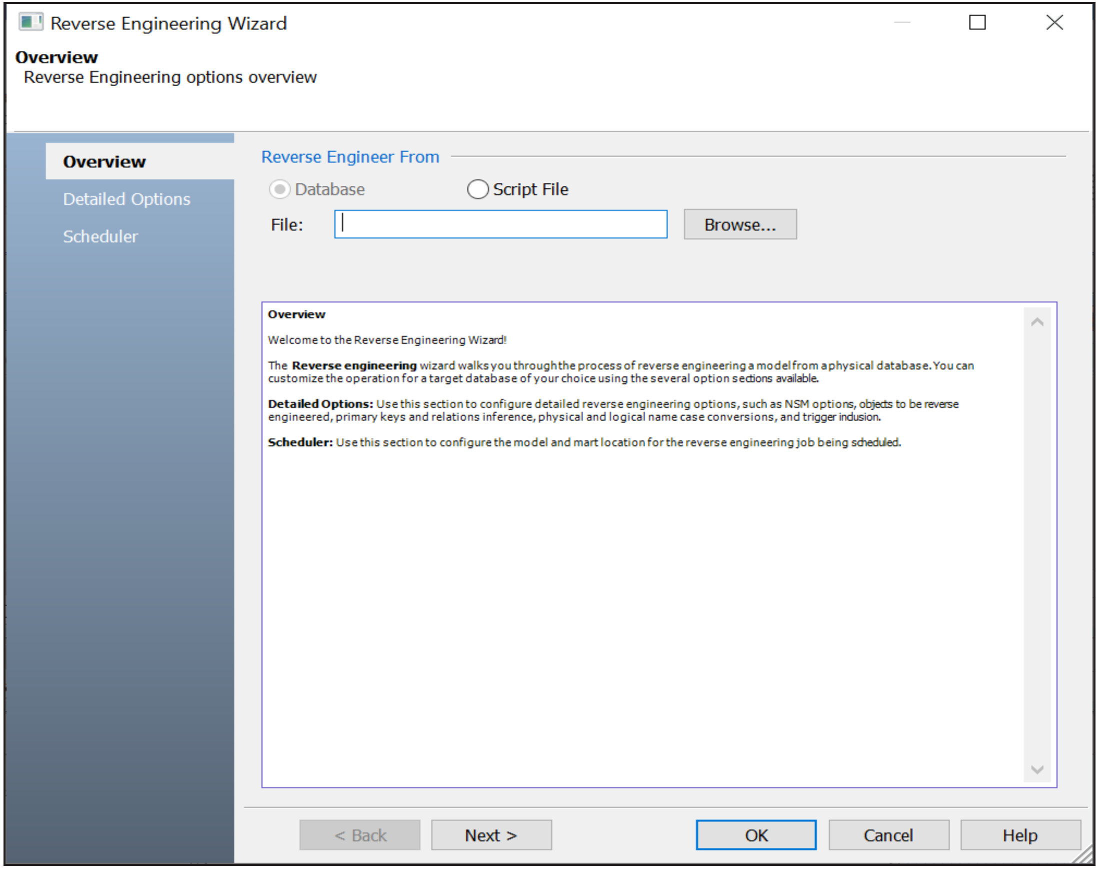
Template

<Default>

□ Preserve the template binding

3. Click **Next**.

   The Reverse Engineering Wizard appears.



Overview
Reverse Engineering options overview

Overview

Welcome to the Reverse Engineering Wizard!

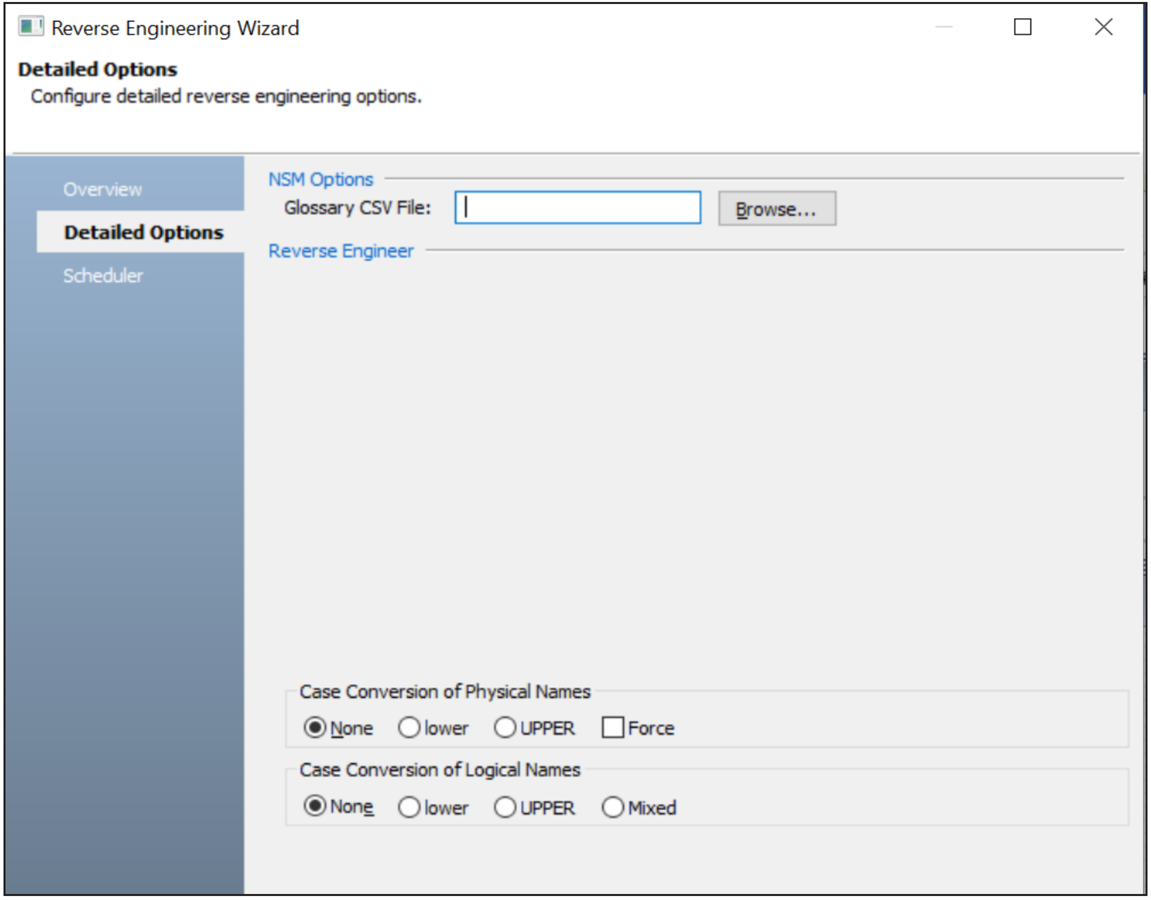
The **Reverse engineering** wizard walks you throughthe process of reverse engineering a modelfrom aphysical database. You can customize the operation for a target database of your choice using the several option sections available.

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

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

   The Detailed Options section appears.



Detailed Options

Configure detailed reverse engineering options.

Glossary CSV File:

Case Conversion of Physical Names

● None ○ lower ○ UPPER □ Force

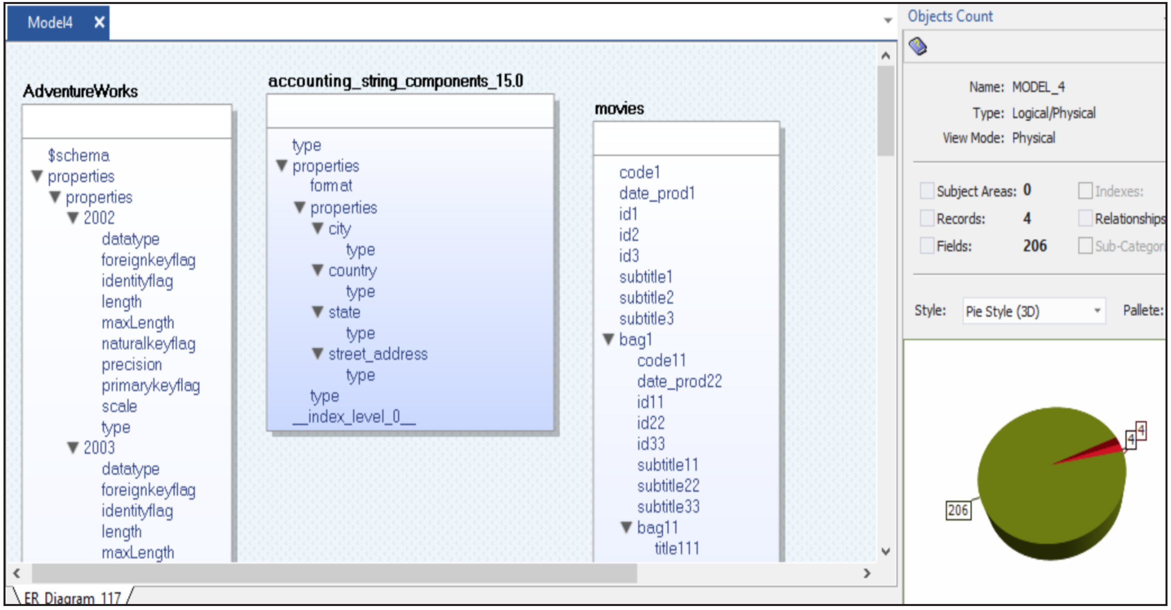
Case Conversion of Logical Names

● None ○ lower ○ UPPER ○ Mixed

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.



Name: MODEL_4
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 0 □ Indexes:
□ Records: 4 □ Relationships
□ Fields: 206 □ Sub-Categor

Style: Pie Style (3D) Pallete:

Reverse Engineering Options for Parquet

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

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 where the reverse engineered model should be saved and its name	When you schedule a job on a remote server, ensure the model path is same for remote and local server. For example: C:\Scheduler\<Model Name>.erwin
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.

Forward Engineering Models

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

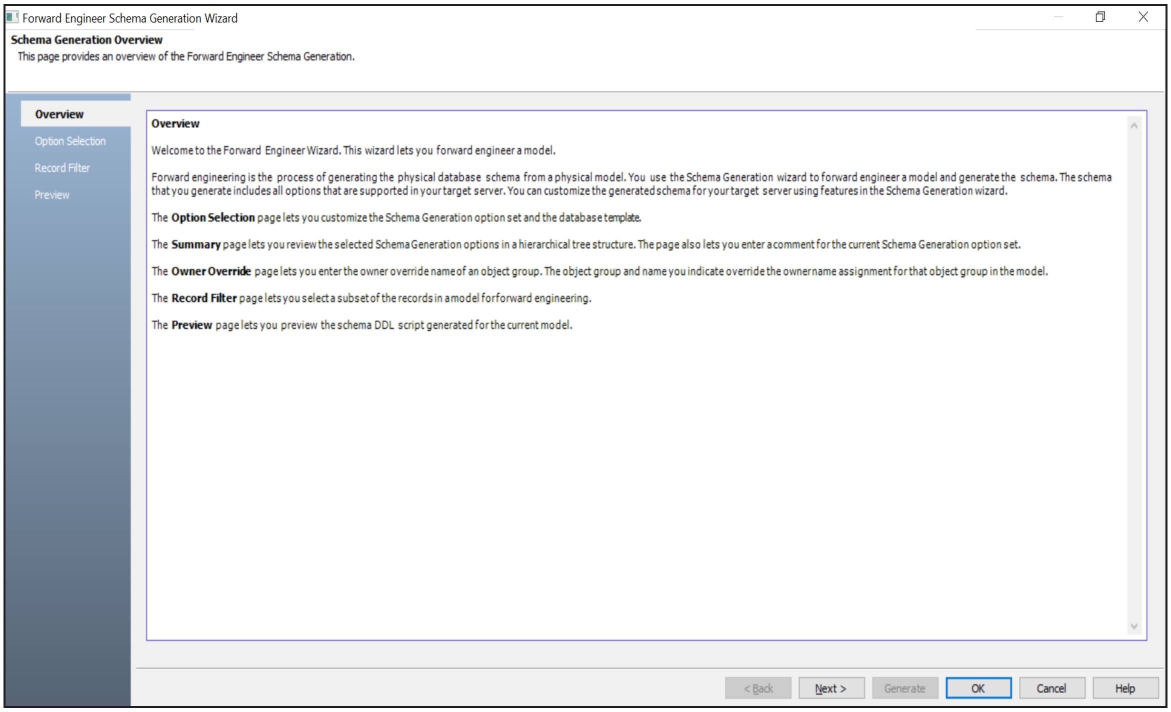
This topic walks you through the steps to forward engineer a Parquet model. For detailed of forward engineering options, refer to the Forward Engineering Options topic.

To forward engineer a Parquet model:

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

   > Ensure that you are in the Physical mode.

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



Schema Generation Overview

This page provides an overview of the Forward Engineer Schema Generation.

Overview

Welcome to the Forward Engineer Wizard. This wizard lets you forward engineer a model.

Forward engineering is the process of generating the physical database schema from a physical model. You use the Schema Generation wizard to forward engineer a model and generate the schema. The schema that you generate includes all options that are supported in your target server. You can customize the generated schema for your target server using features in the Schema Generation wizard.

The **Option Selection** page lets you customize the Schema Generation option set and the database template.

The **Summary** page lets you review the selected Schema Generation options in a hierarchical tree structure. The page also lets you enter a comment for the current Schema Generation option set.

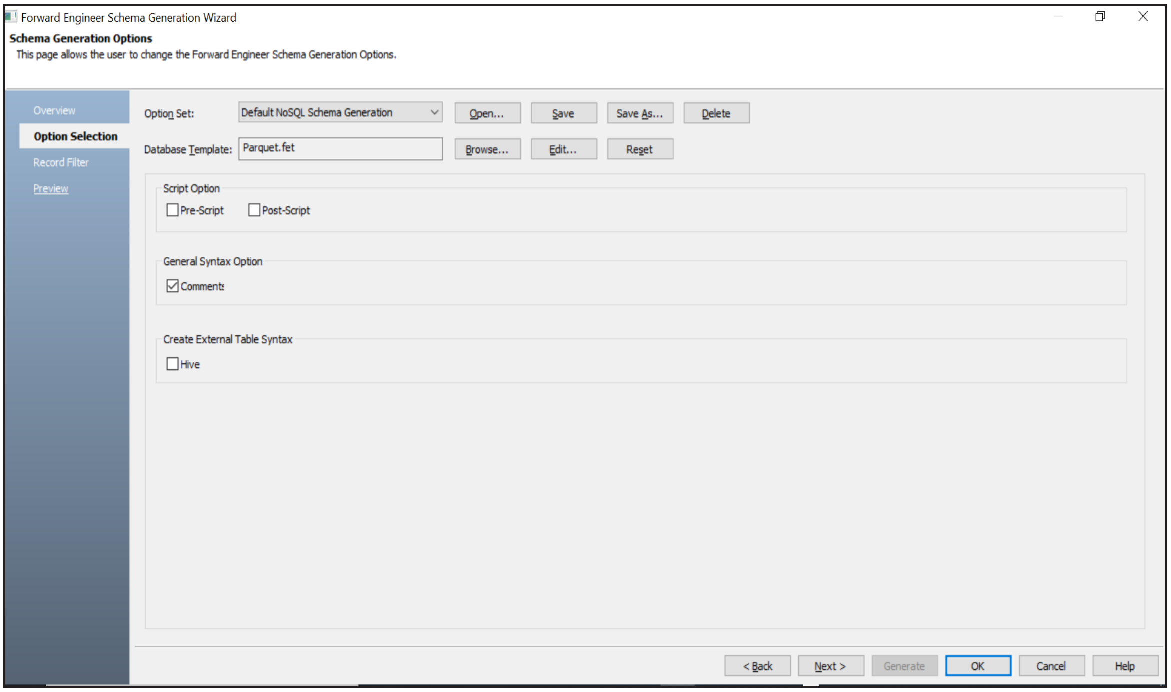
The **Owner Override** page lets you enter the owner override name of an object group. The object group and name you indicate override the ownername assignment for that object group in the model.

The **Record Filter** page lets you select a subset of the records in a model forforward engineering.

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

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

options.



Forward Engineer Schema Generation Wizard

Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default NoSQL Schema Generation

Database Template: Parquet.fet

Script Option

□ Pre-Script □ Post-Script

General Syntax Option

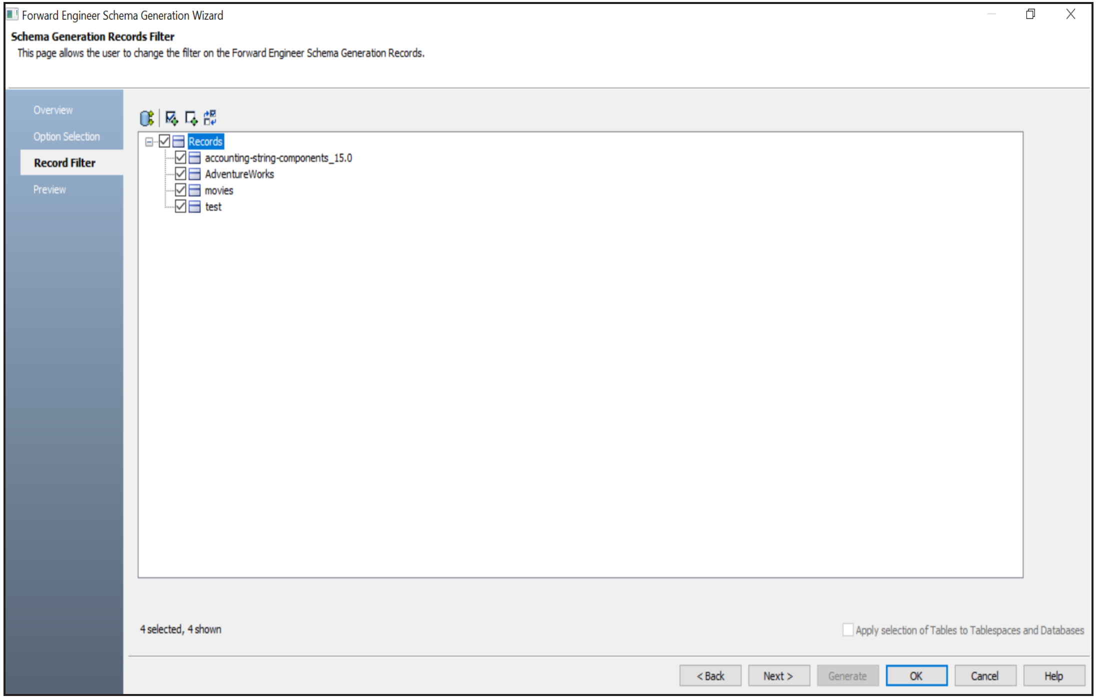
☑ Comments

Create External Table Syntax

□ Hive

4. Click **Next**.

   The Record Filter section appears. It displays a list of records available in your model.



Forward Engineer Schema Generation Wizard

Schema Generation Records Filter

This page allows the user to change the filter on the Forward Engineer Schema Generation Records.

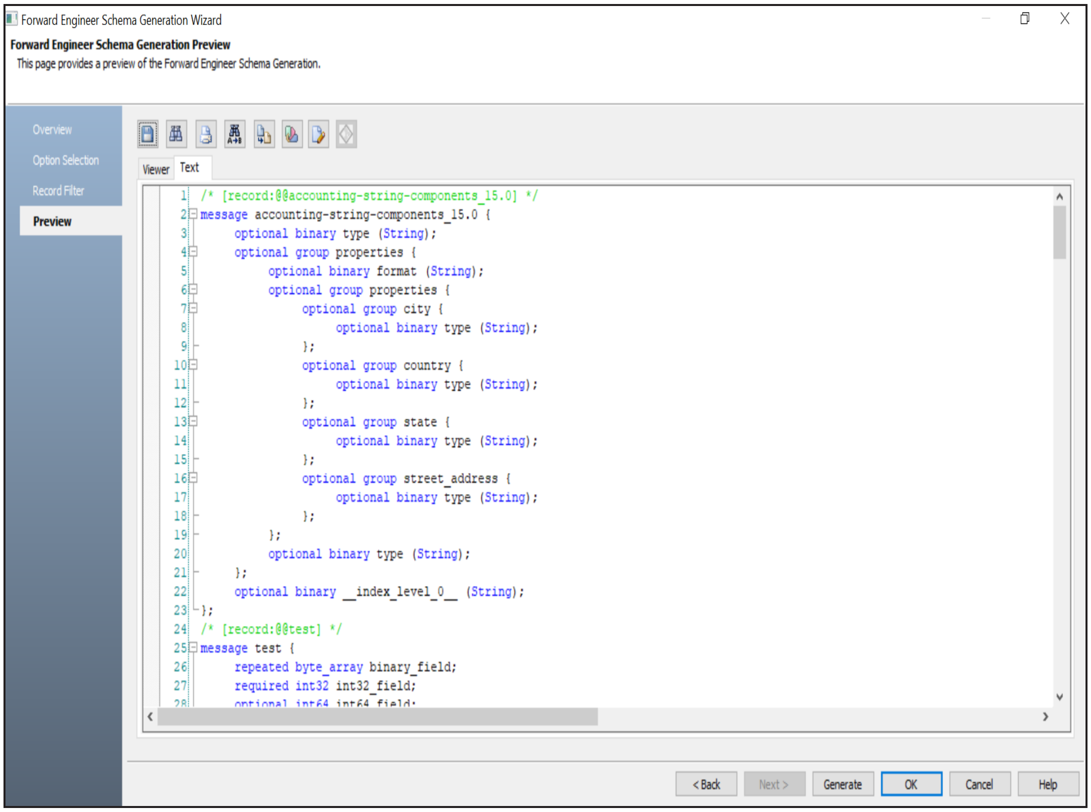
- ☑ Records
  - ☑ accounting-string-components_15.0
  - ☑ AdventureWorks
  - ☑ movies
  - ☑ test

4 selected, 4 shown

□ Apply selection of Tables to Tablespaces and Databases

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

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



Forward Engineer Schema Generation Wizard

Forward Engineer Schema Generation Preview

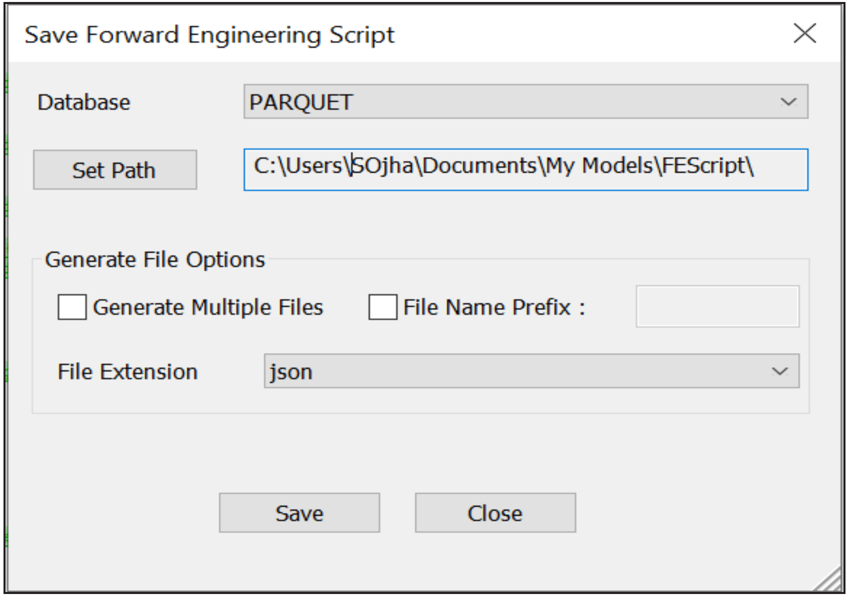
This page provides a preview of the Forward Engineer Schema Generation.

```
/* [record:@@accounting-string-components_15.0] */
message accounting-string-components_15.0 {
     optional binary type (String);
     optional group properties {
          optional binary format (String);
          optional group properties {
               optional group city {
                    optional binary type (String);
               };
               optional group country {
                    optional binary type (String);
               };
               optional group state {
                    optional binary type (String);
               };
               optional group street_address {
                    optional binary type (String);
               };
          };
          optional binary type (String);
     };
     optional binary __index_level_0__ (String);
};
/* [record:@@test] */
message test {
     repeated byte_array binary_field;
     required int32 int32_field;
     optional int64 int64_field;
```

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.



Database: PARQUET

Set Path: C:\Users\SOjha\Documents\My Models\FEScript\

□ Generate Multiple Files

□ File Name Prefix :

File Extension: json

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

The following are the forward engineering options for Parquet in erwin DM.

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.

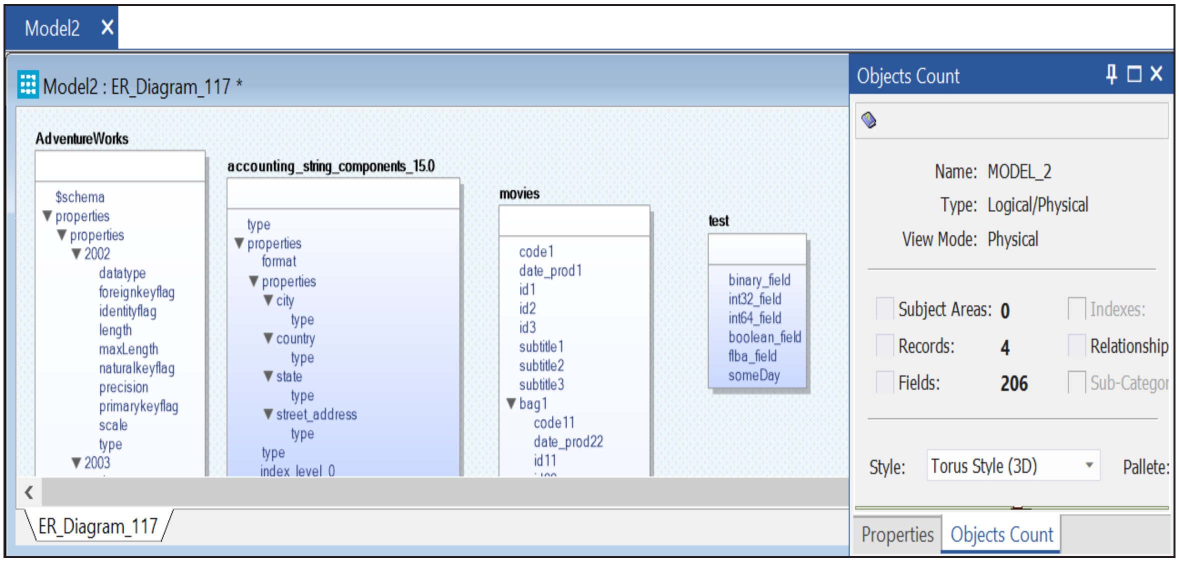
This topic walks you through the steps to compare a Parquet model with script.

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.

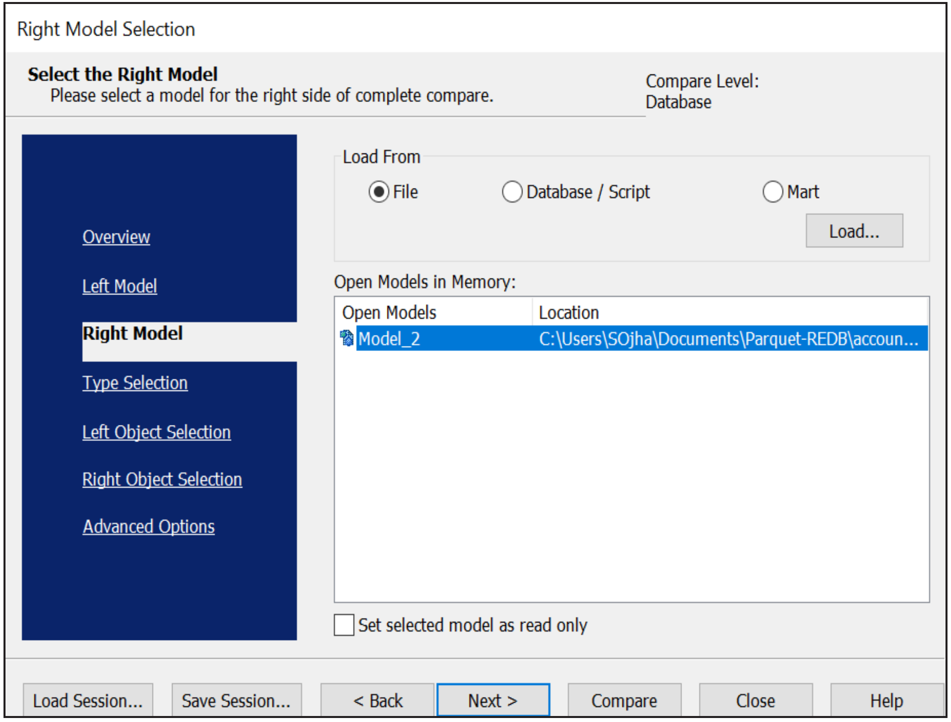


Name: MODEL_2
Type: Logical/Physical
View Mode: Physical

Subject Areas: 0
Records: 4
Fields: 206

Style: Torus Style (3D)

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.



Right Model Selection

Select the Right Model
Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

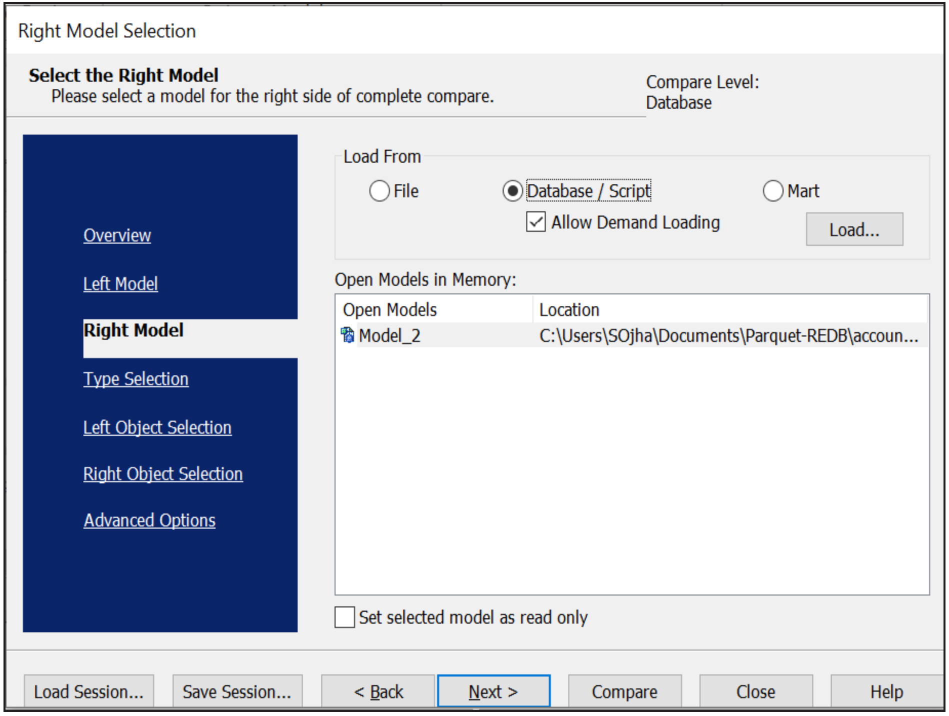
● File ○ Database / Script ○ Mart

Open Models in Memory:

Open Models	Location
Model_2	C:\Users\SOjha\Documents\Parquet-REDB\accoun...

□ Set selected model as read only

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



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

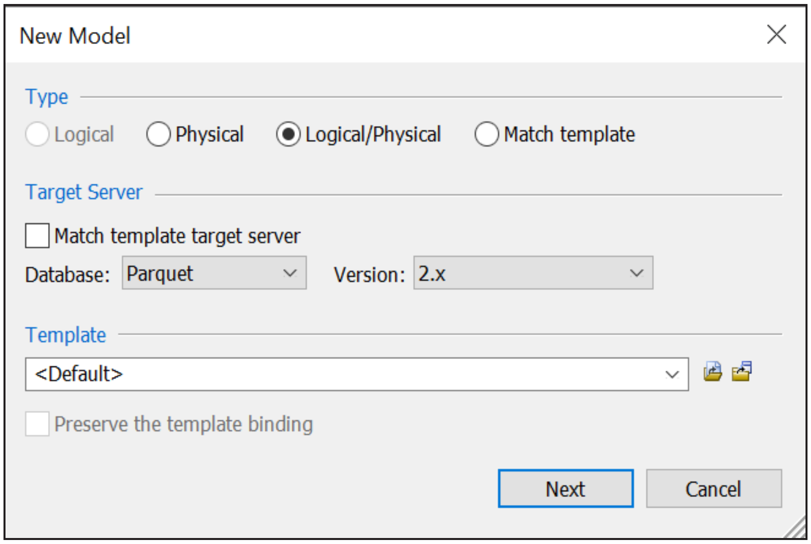
Open Models in Memory:

Open Models	Location
Model_2	C:\Users\SOjha\Documents\Parquet-REDB\accoun...

□ Set selected model as read only

4. Click **Load**.

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



New Model

Type

○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

Database: Parquet Version: 2.x

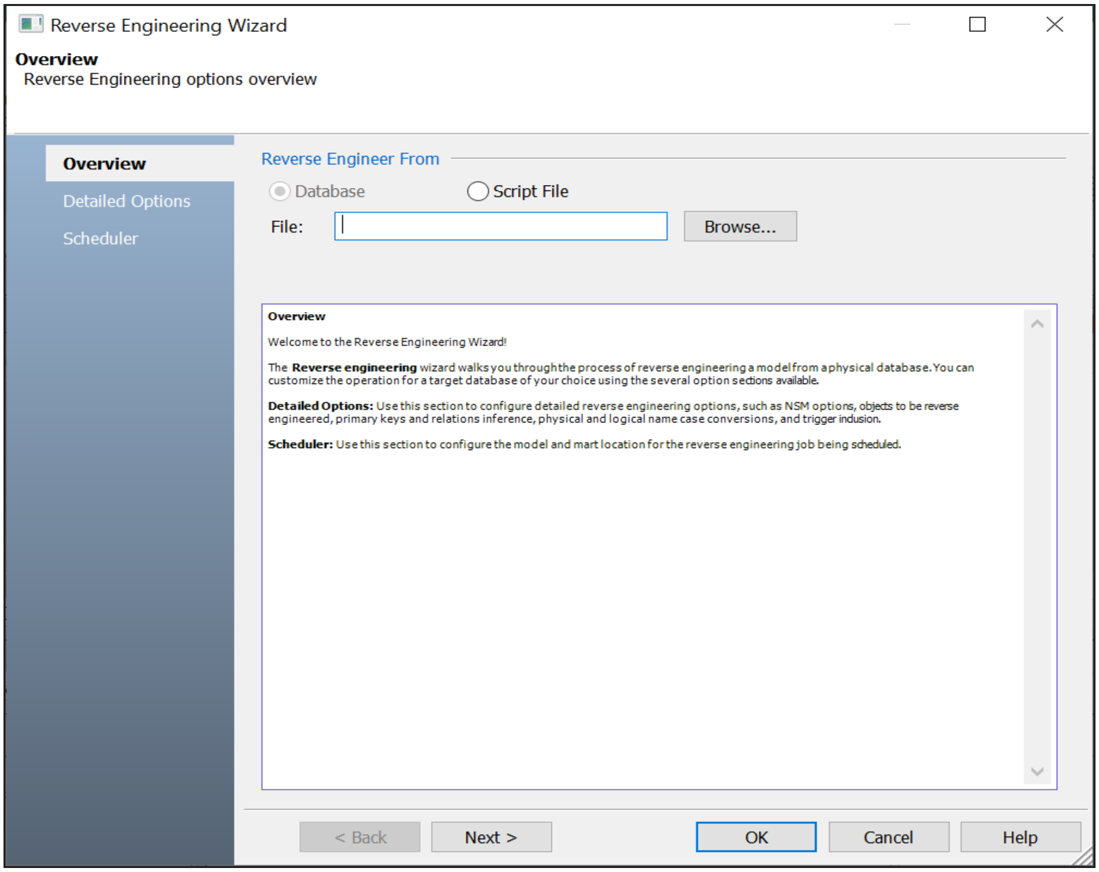
Template

<Default>

□ Preserve the template binding

5. Ensure that the Database is set to Parquet. Then, click **Next**.

   The Reverse Engineer Process Wizard appears.



Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you throughthe process of reverse engineering a modelfrom aphysical database. You can customize the operation for a target database of your choice using the several option sections available.

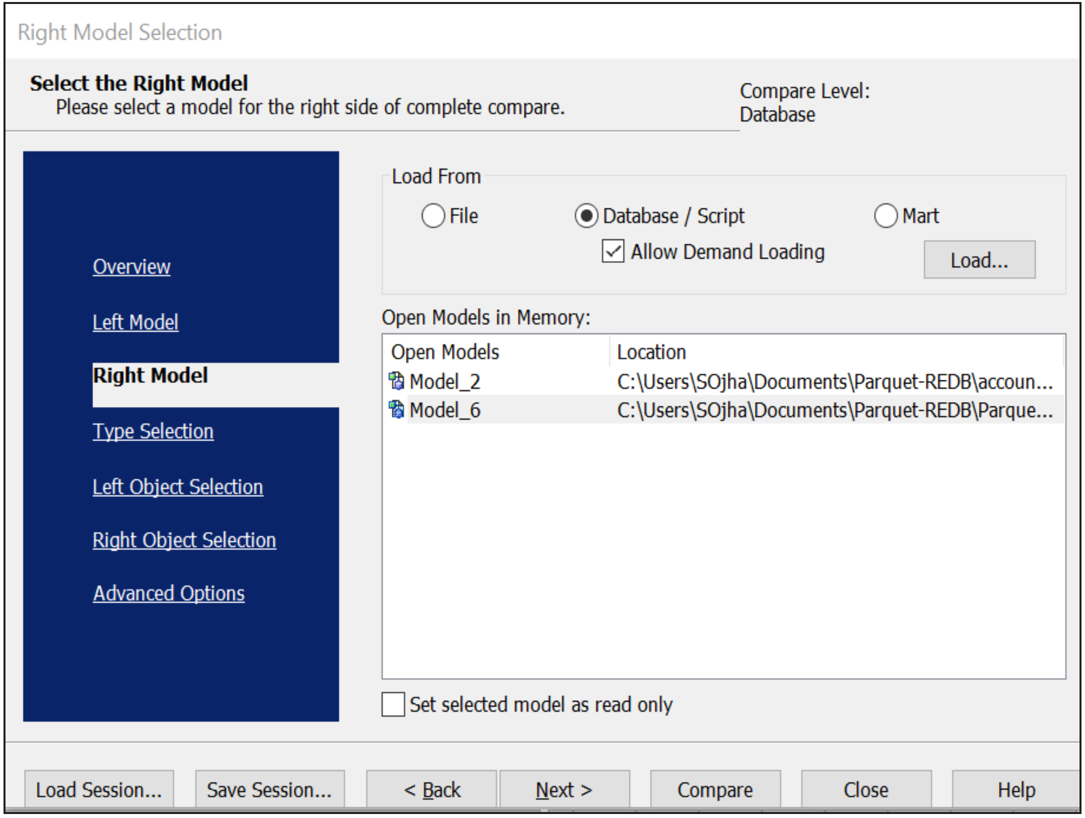
Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

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.



Right Model Selection

Select the Right Model
Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

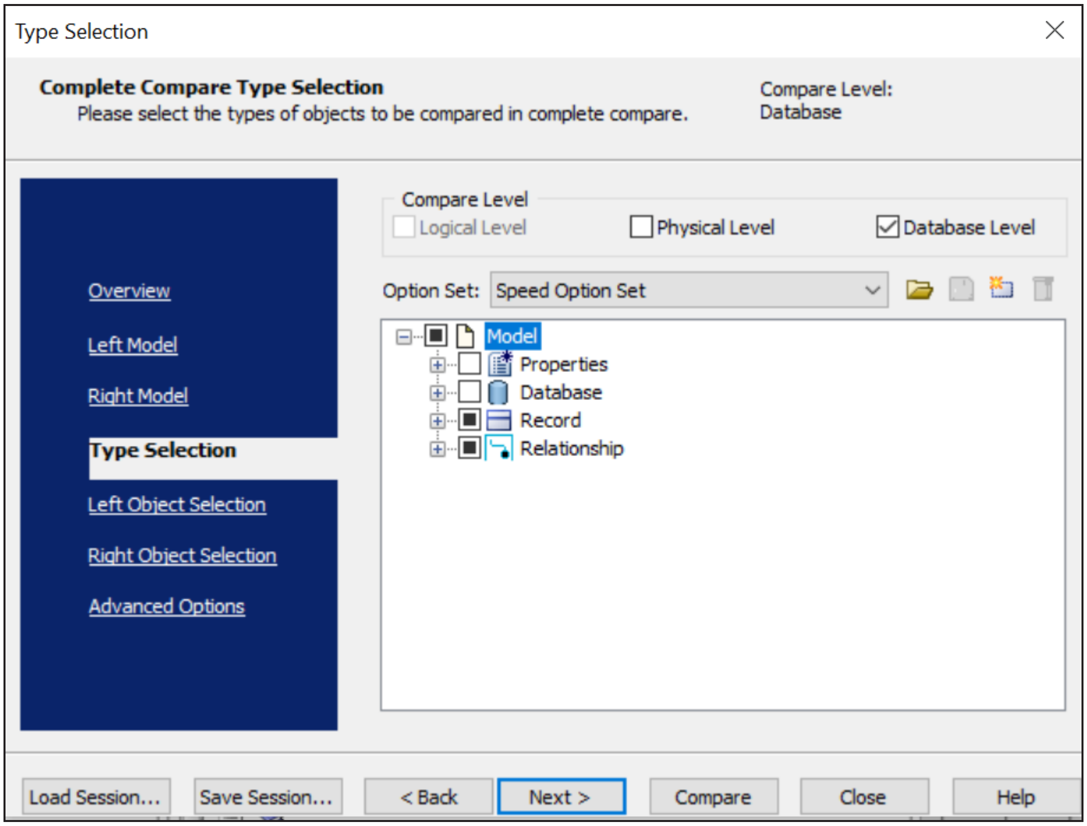
Open Models in Memory:

Open Models	Location
Model_2	C:\Users\SOjha\Documents\Parquet-REDB\accoun...
Model_6	C:\Users\SOjha\Documents\Parquet-REDB\Parque...

□ Set selected model as read only

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

For example, the following image shows the default options.



Type Selection

Complete Compare Type Selection

Please select the types of objects to be compared in complete compare.

Compare Level: Database

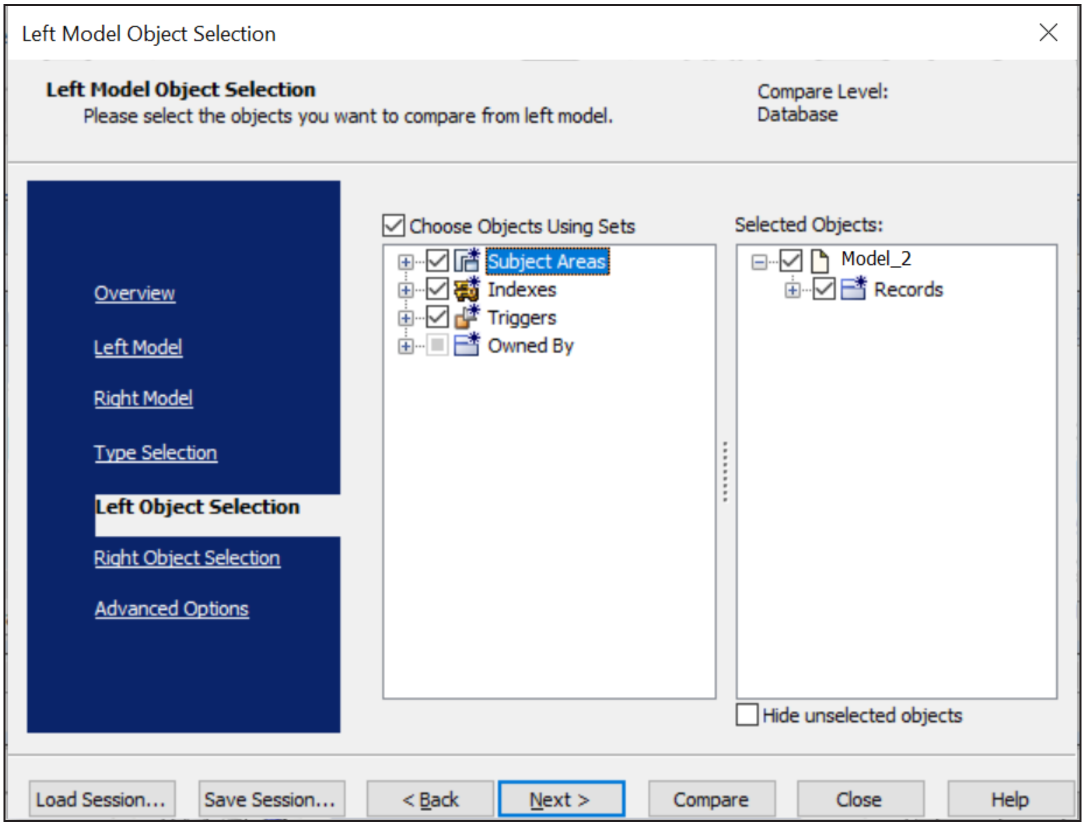
Compare Level: □ Logical Level □ Physical Level ☑ Database Level

Option Set: Speed Option Set

- ■ Model
  - □ Properties
  - □ Database
  - ■ Record
  - ■ Relationship

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

For example, the following image shows the default options.



Left Model Object Selection

Left Model Object Selection

Please select the objects you want to compare from left model.

Compare Level:
Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

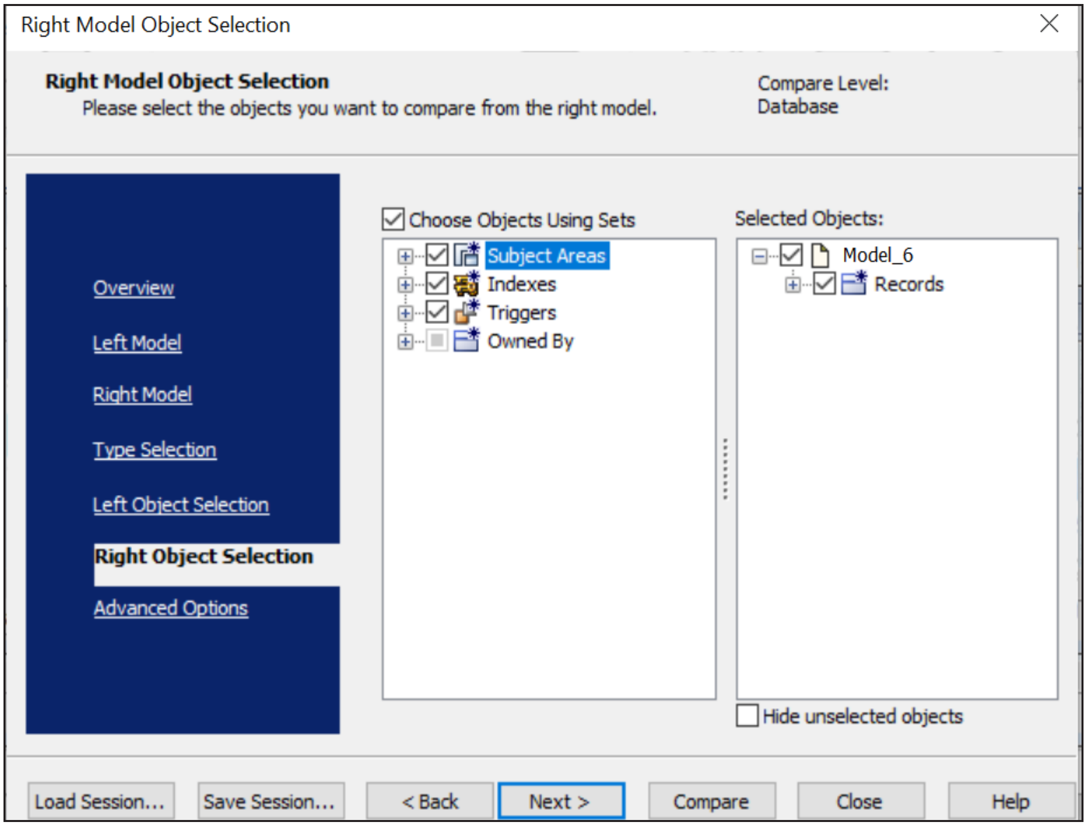
Selected Objects:

- ☑ Model_2
  - ☑ Records

□ Hide unselected objects

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

For example, the following image shows the default options.



Right Model Object Selection

Right Model Object Selection

Please select the objects you want to compare from the right model.

Compare Level:
Database

☑ Choose Objects Using Sets

☑ Subject Areas
☑ Indexes
☑ Triggers
□ Owned By

Selected Objects:

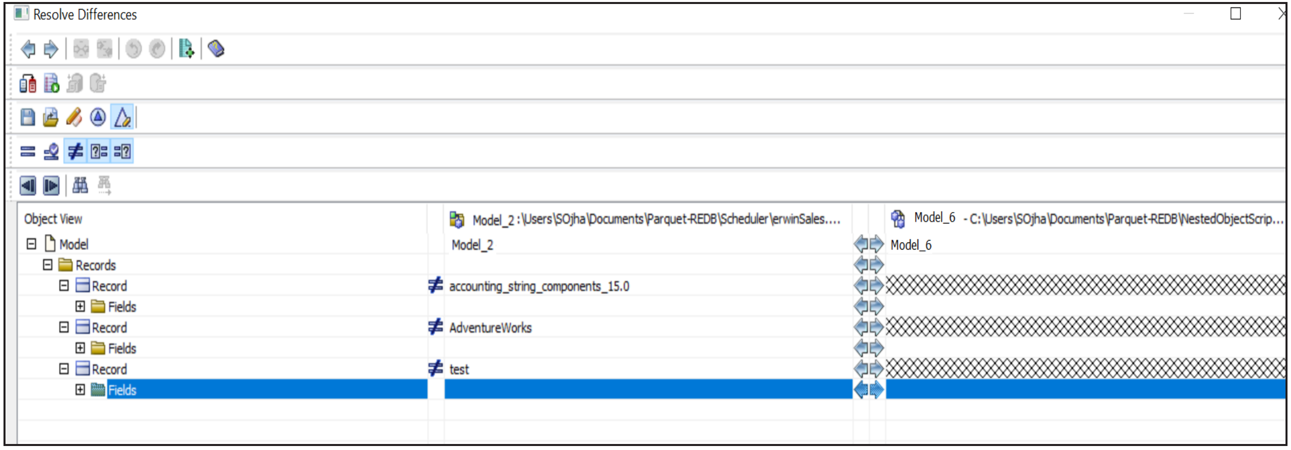
☑ Model_6
☑ Records

□ Hide unselected objects

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.

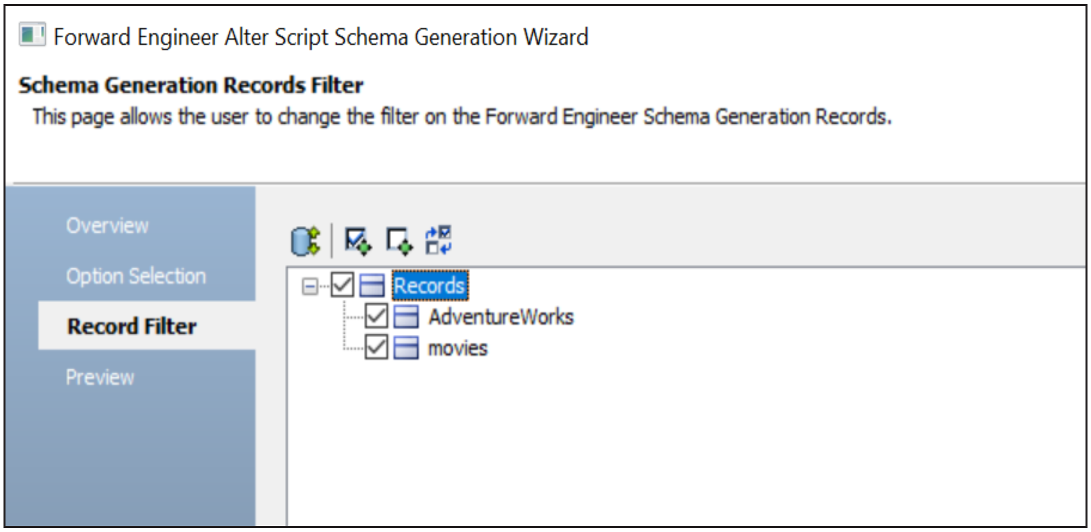


Object View		Model_2 : \Users\SOjha\Documents\Parquet-REDB\Scheduler\erwinSales....	Model_6 - C:\Users\SOjha\Documents\Parquet-REDB\NestedObjectScrip...
⊟ Model		Model_2	Model_6
⊟ Records			
⊟ Record	≠	accounting_string_components_15.0	
⊞ Fields			
⊟ Record	≠	AdventureWorks	
⊞ Fields			
⊟ Record	≠	test	
⊞ Fields			

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.


Forward Engineer Alter Script Schema Generation Wizard

Schema Generation Records Filter

This page allows the user to change the filter on the Forward Engineer Schema Generation Records.

⊟ ☑ Records
- ☑ AdventureWorks
- ☑ movies

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.

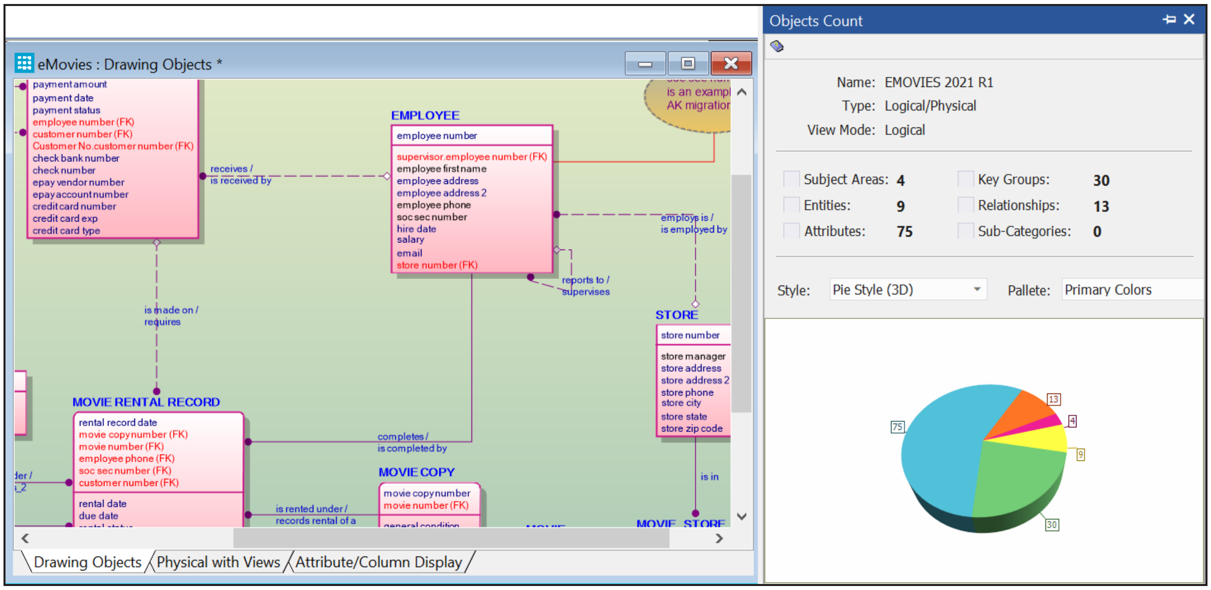
Migration by Changing the Target Database

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.



Objects Count

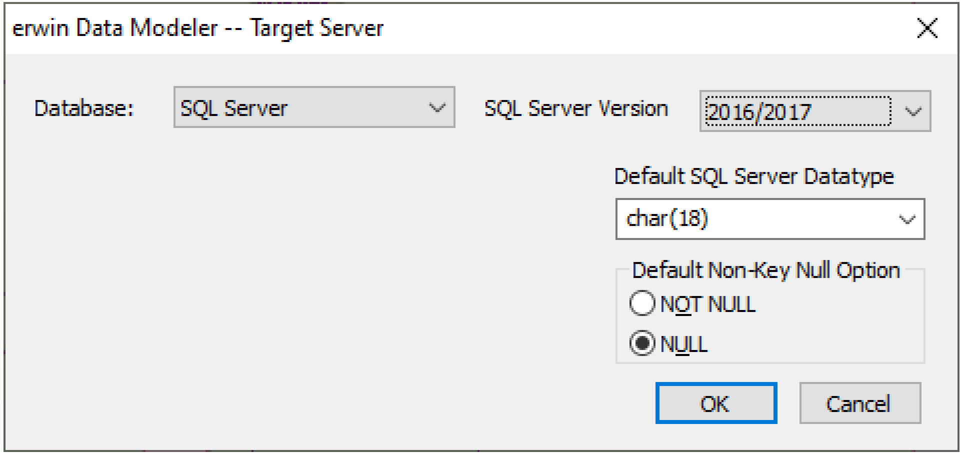
Name: EMOVIES 2021 R1
Type: Logical/Physical
View Mode: Logical

□ Subject Areas:	4	□ Key Groups:	30
□ Entities:	9	□ Relationships:	13
□ Attributes:	75	□ Sub-Categories:	0

Style: Pie Style (3D)   Pallete: Primary Colors

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.



erwin Data Modeler -- Target Server

Database: SQL Server

SQL Server Version: 2016/2017

Default SQL Server Datatype: char(18)

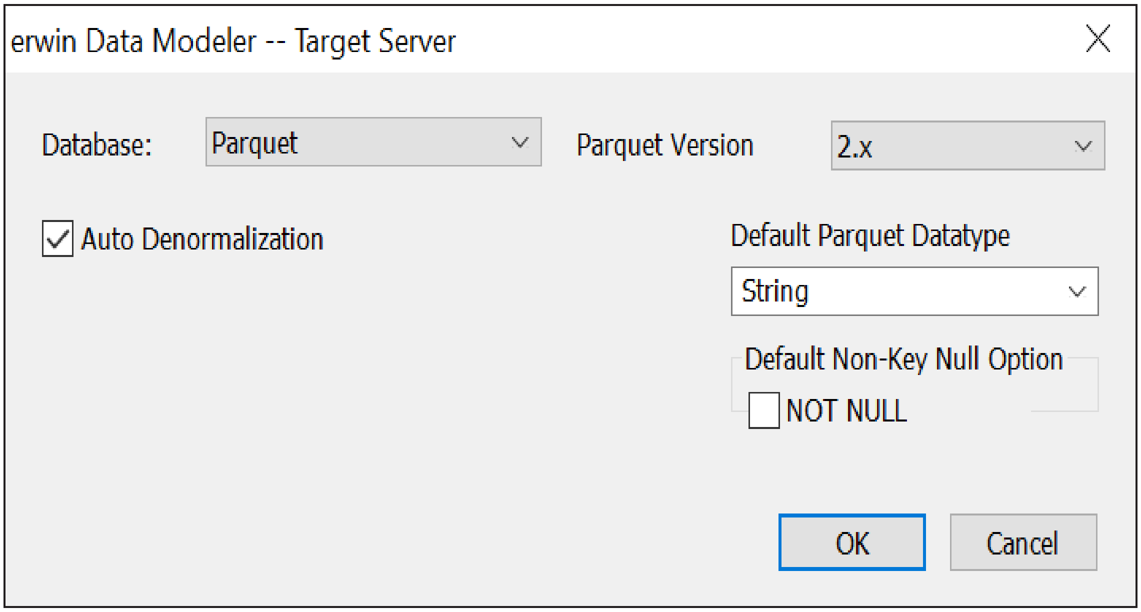
Default Non-Key Null Option

○ NOT NULL

● NULL

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

   By default, the Auto Denormalization check box is selected. Keep it selected.



erwin Data Modeler -- Target Server

Database: Parquet

Parquet Version: 2.x

☑ Auto Denormalization

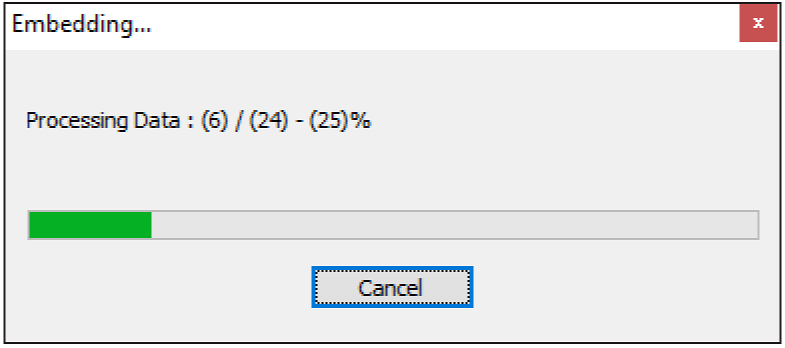
Default Parquet Datatype: String

Default Non-Key Null Option

□ NOT NULL

4. Click **OK**.

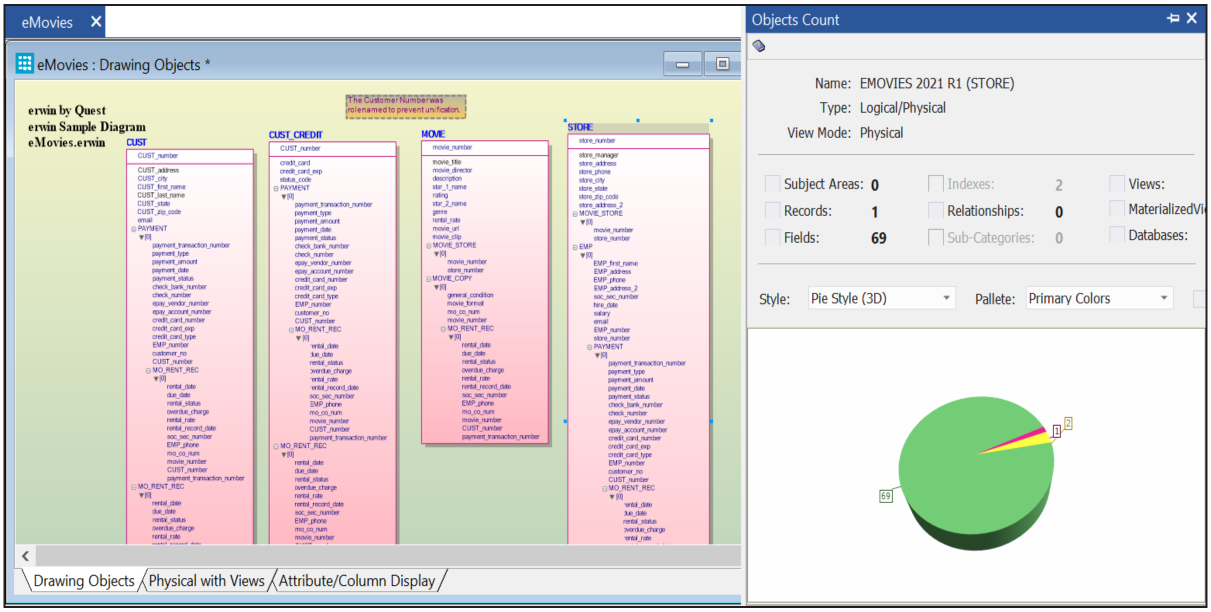
   The conversion process starts.



Embedding...

Processing Data : (6) / (24) - (25)%

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



Name: EMOVIES 2021 R1 (STORE)
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 0 □ Indexes: 2 □ Views:
□ Records: 1 □ Relationships: 0 □ MaterializedVi
□ Fields: 69 □ Sub-Categories: 0 □ Databases:

Style: Pie Style (3D) Pallete: Primary Colors

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

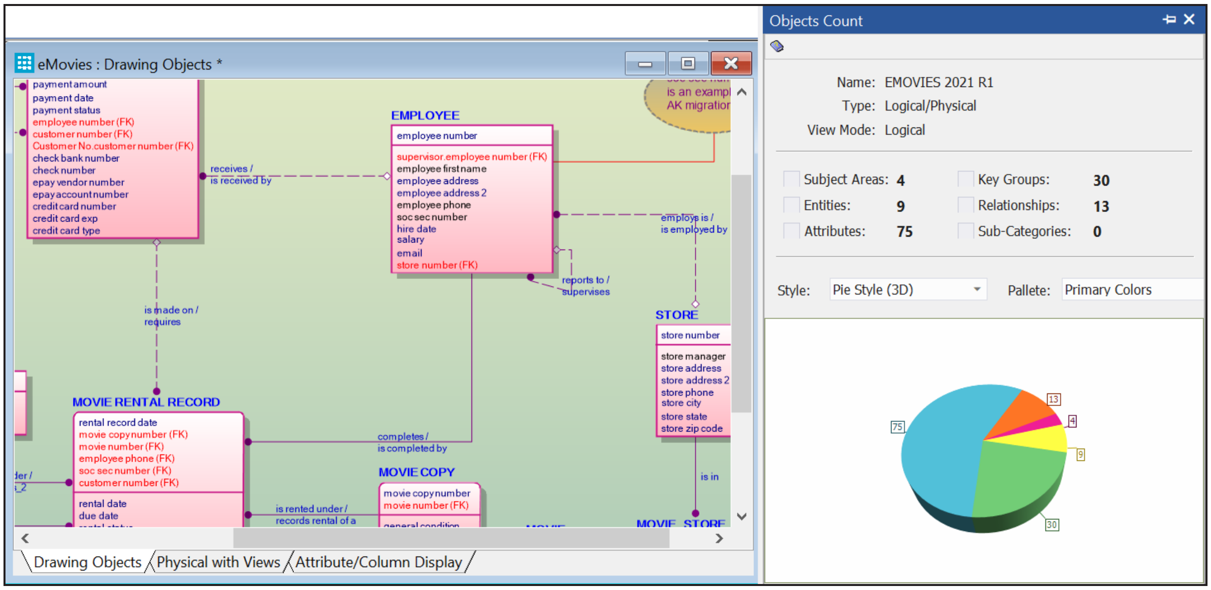
To migrate by deriving a model, follow these steps:

Migrating Relational Models to Parquet Models

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.



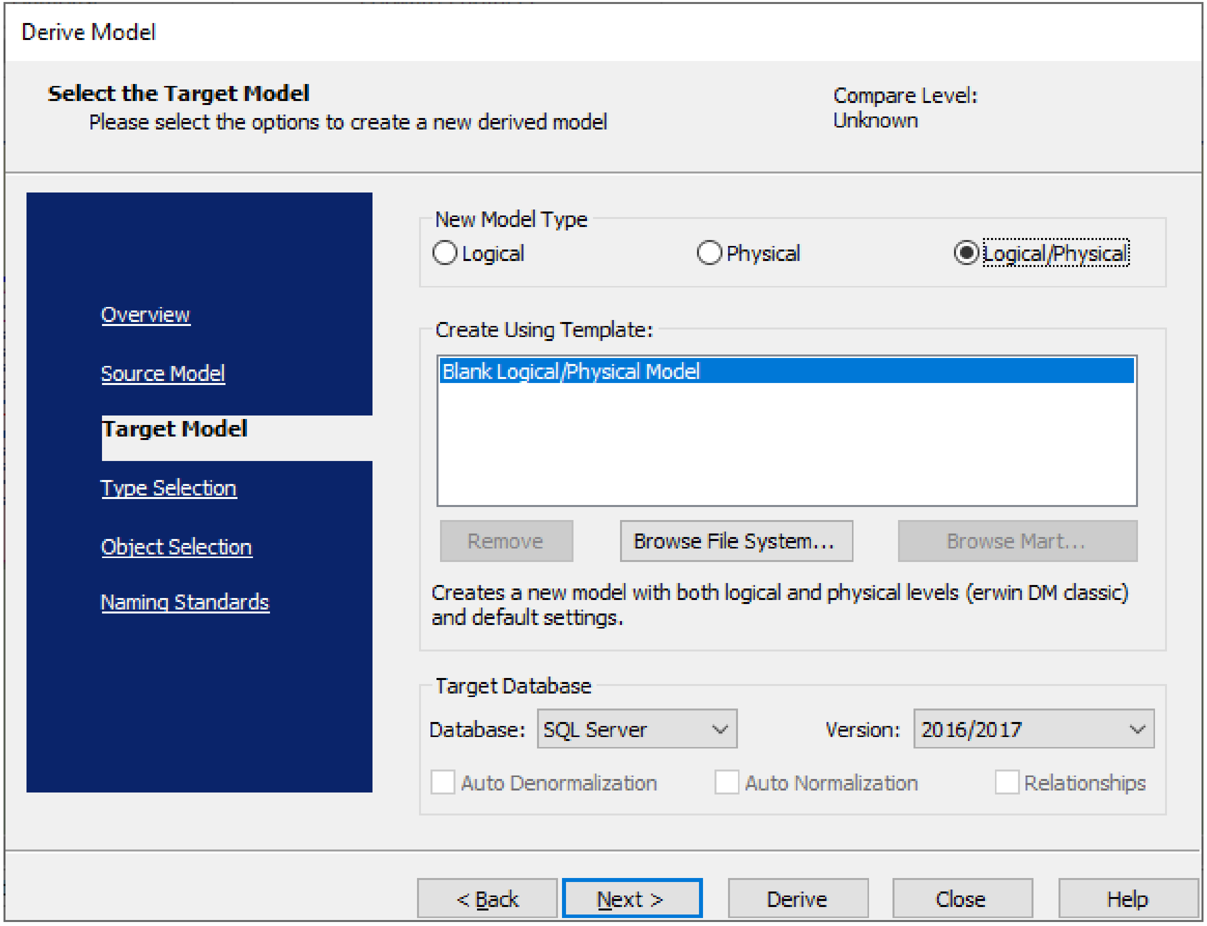
Name: EMOVIES 2021 R1
Type: Logical/Physical
View Mode: Logical

Subject Areas:	4	Key Groups:	30
Entities:	9	Relationships:	13
Attributes:	75	Sub-Categories:	0

Style: Pie Style (3D) Pallete: Primary Colors

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

New Model Type

○ Logical ○ Physical ● Logical/Physical

Create Using Template:

Blank Logical/Physical Model

Creates a new model with both logical and physical levels (erwin DM classic) and default settings.

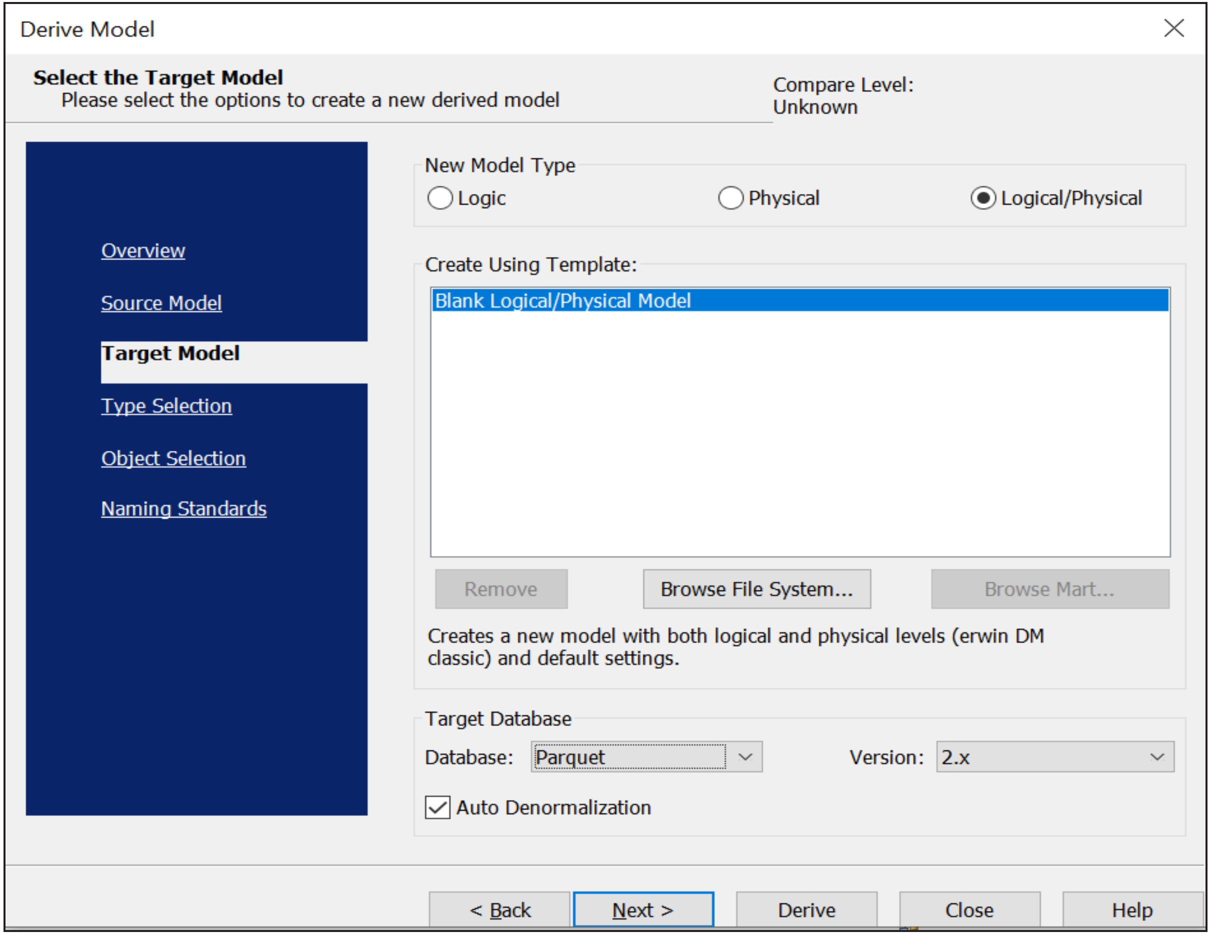
Target Database

Database: SQL Server Version: 2016/2017

□ Auto Denormalization □ Auto Normalization □ Relationships

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

   By default, the Auto Denormalization check box is selected. Keep it selected.



Derive Model

Select the Target Model
Please select the options to create a new derived model

Compare Level:
Unknown

New Model Type
○ Logic ○ Physical ● Logical/Physical

Create Using Template:
Blank Logical/Physical Model

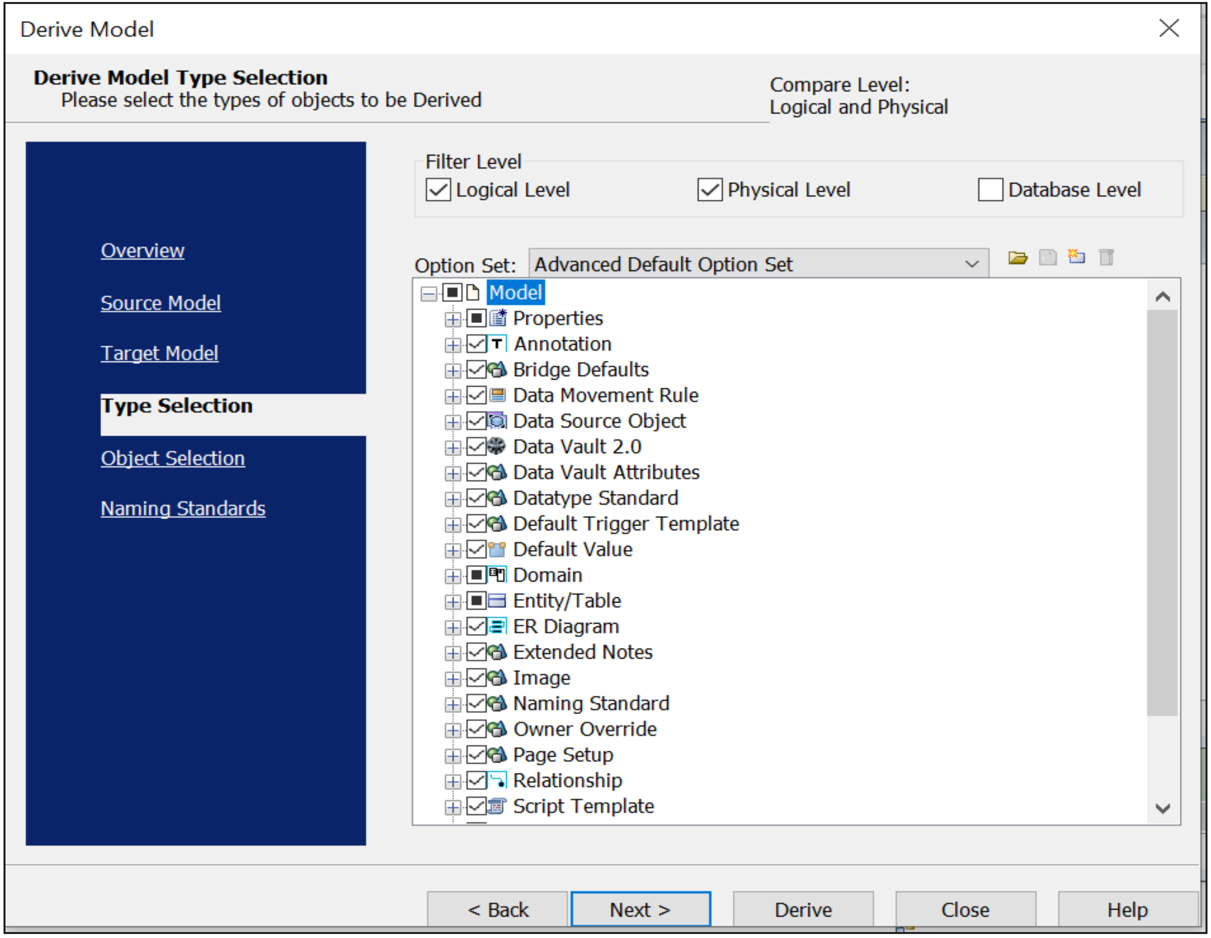
Creates a new model with both logical and physical levels (erwin DM classic) and default settings.

Target Database
Database: Parquet Version: 2.x
☑ Auto Denormalization

4. Click **Next**.

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

The Type Selection section appears.



Derive Model Type Selection
Please select the types of objects to be Derived

Compare Level:
Logical and Physical

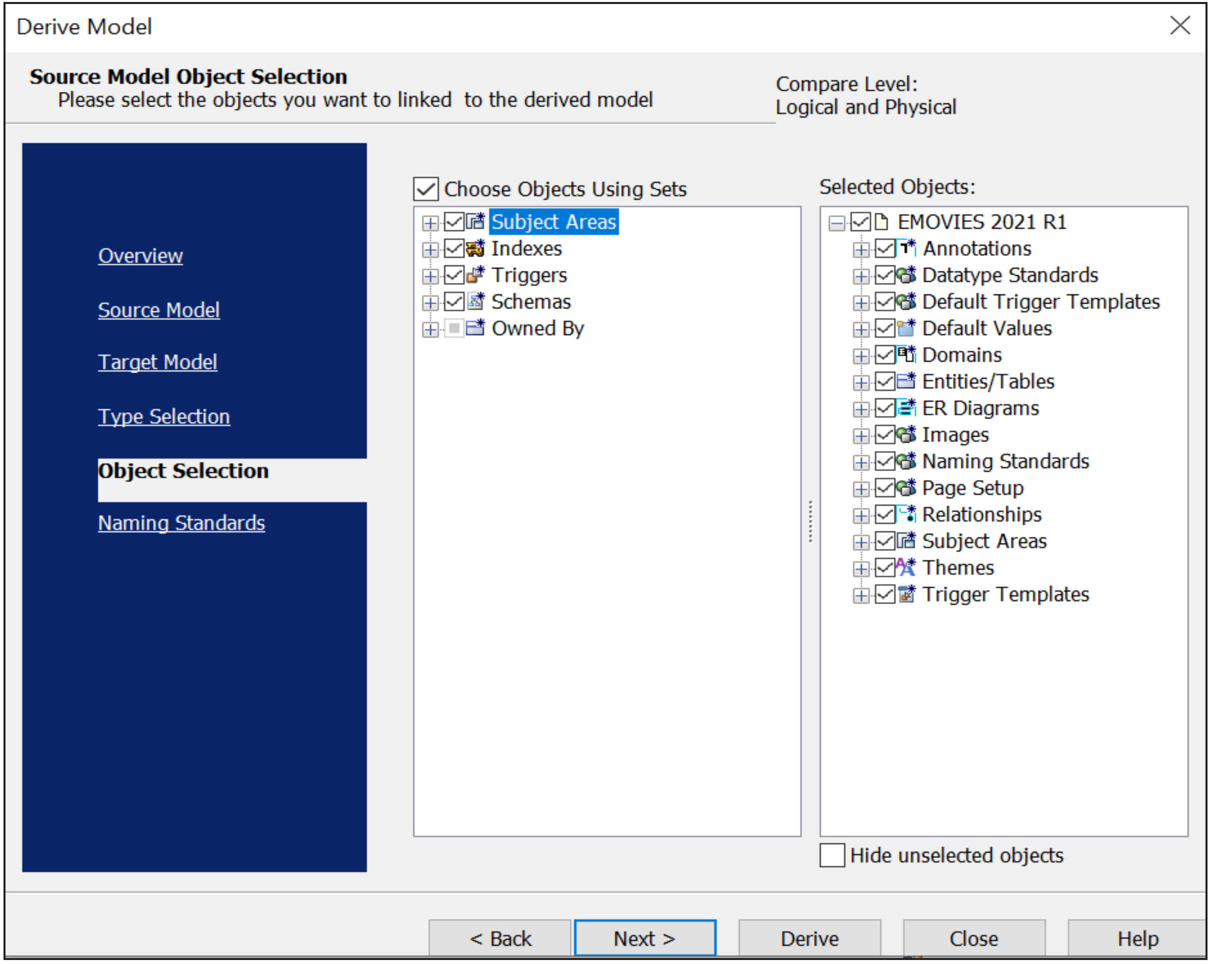
Filter Level
☑ Logical Level ☑ Physical Level ☐ Database Level

Option Set: Advanced Default Option Set

- ■ Model
  - ■ Properties
  - ☑ Annotation
  - ☑ Bridge Defaults
  - ☑ Data Movement Rule
  - ☑ Data Source Object
  - ☑ Data Vault 2.0
  - ☑ Data Vault Attributes
  - ☑ Datatype Standard
  - ☑ Default Trigger Template
  - ☑ Default Value
  - ■ Domain
  - ■ Entity/Table
  - ☑ ER Diagram
  - ☑ Extended Notes
  - ☑ Image
  - ☑ Naming Standard
  - ☑ Owner Override
  - ☑ Page Setup
  - ☑ Relationship
  - ☑ Script Template

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.



Source Model Object Selection
Please select the objects you want to linked to the derived model

Compare Level:
Logical and Physical

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- ☑ Schemas
- ☐ Owned By

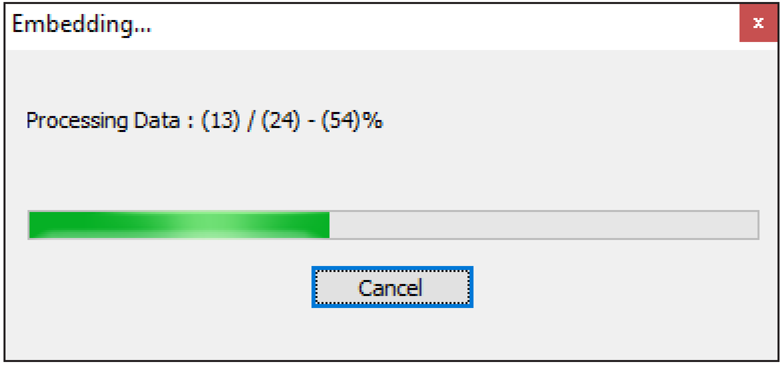
Selected Objects:

- ☑ EMOVIES 2021 R1
  - ☑ Annotations
  - ☑ Datatype Standards
  - ☑ Default Trigger Templates
  - ☑ Default Values
  - ☑ Domains
  - ☑ Entities/Tables
  - ☑ ER Diagrams
  - ☑ Images
  - ☑ Naming Standards
  - ☑ Page Setup
  - ☑ Relationships
  - ☑ Subject Areas
  - ☑ Themes
  - ☑ Trigger Templates

☐ Hide unselected objects

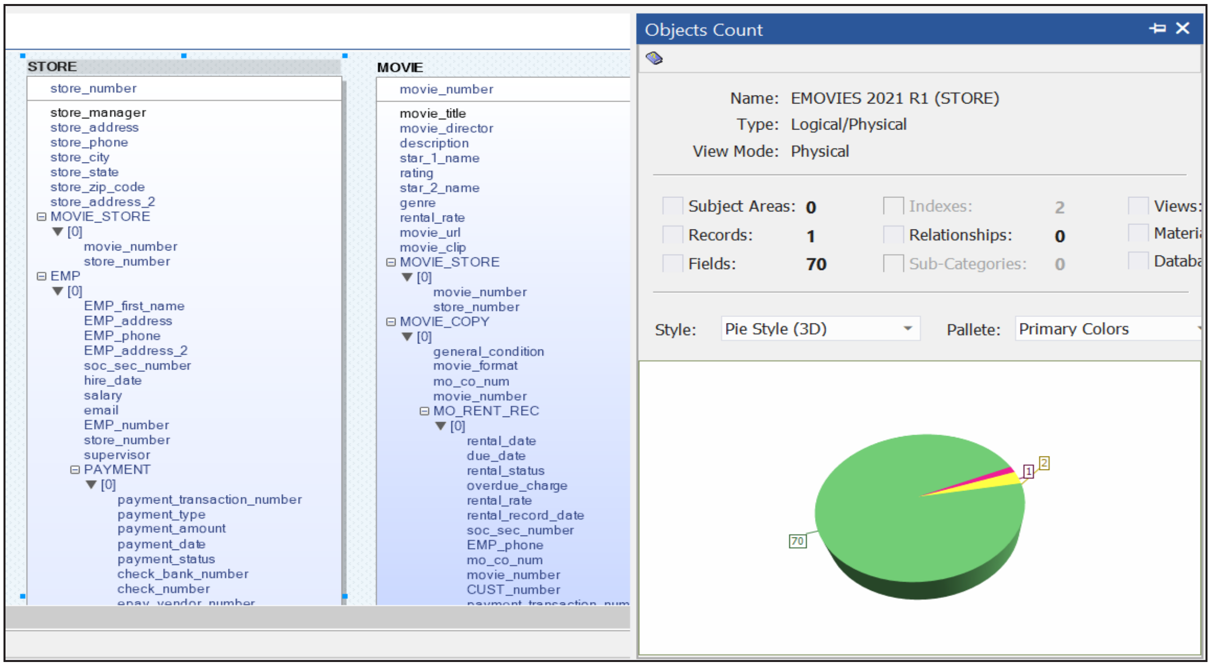
7. Select the objects that you want to derive into the target Parquet model.

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



Processing Data : (13) / (24) - (54)%

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



Name: EMOVIES 2021 R1 (STORE)
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 0
□ Records: 1
□ Fields: 70
□ Indexes: 2
□ Relationships: 0
□ Sub-Categories: 0
□ Views:
□ Materi
□ Databa

Style: Pie Style (3D)
Pallete: Primary Colors

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.

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

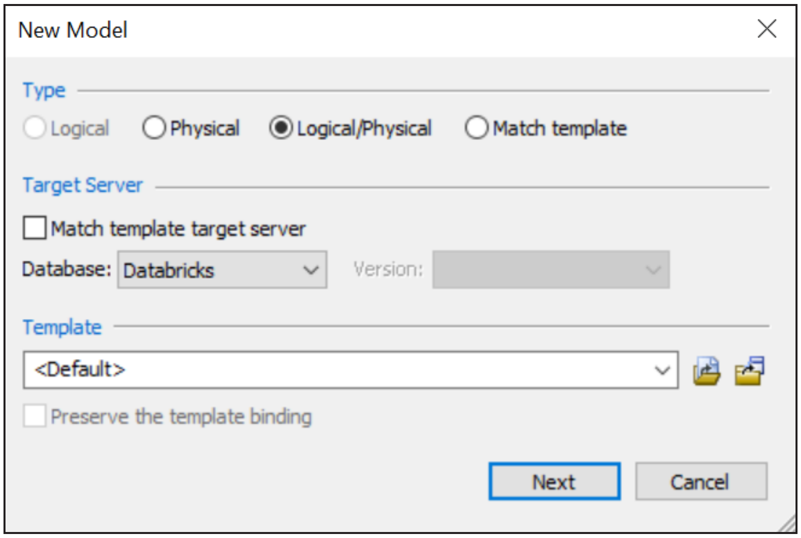
Reverse Engineering Models

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.



New Model

Type

○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

Database: Databricks

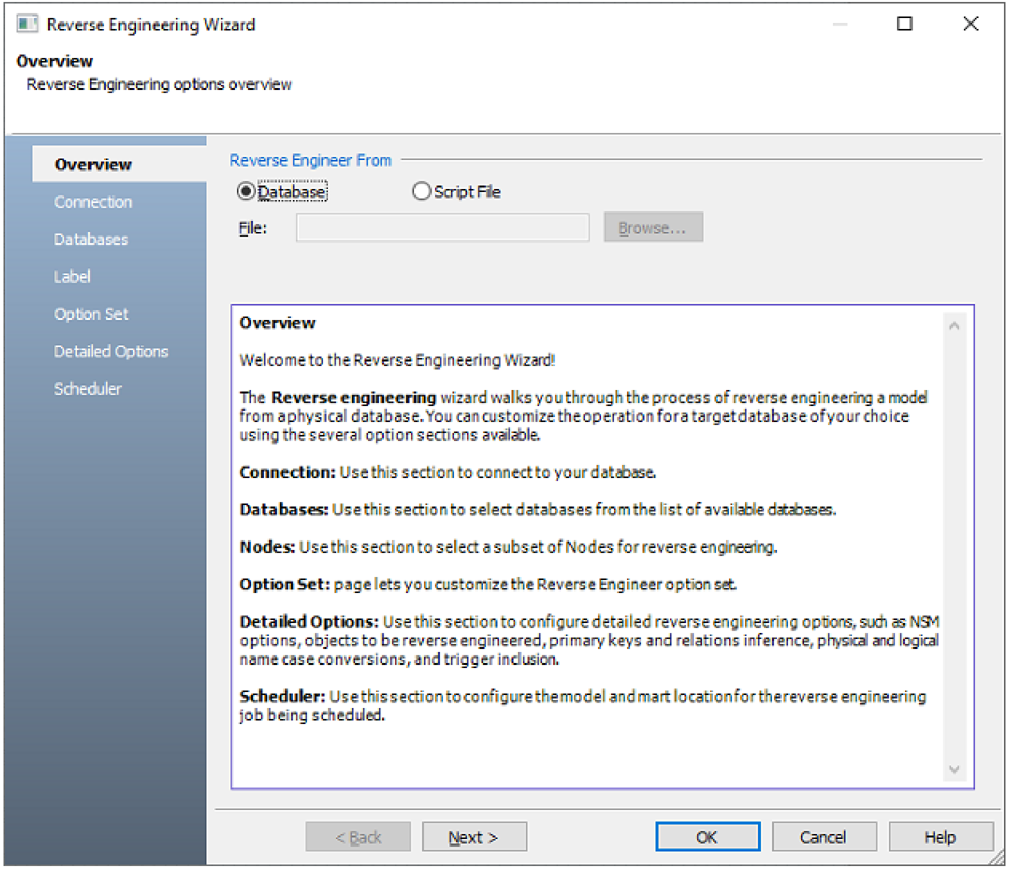
Version:

Template

<Default>

□ Preserve the template binding

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



Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Databases: Use this section to select databases from the list of available databases.

Nodes: Use this section to select a subset of Nodes for reverse engineering.

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

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

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

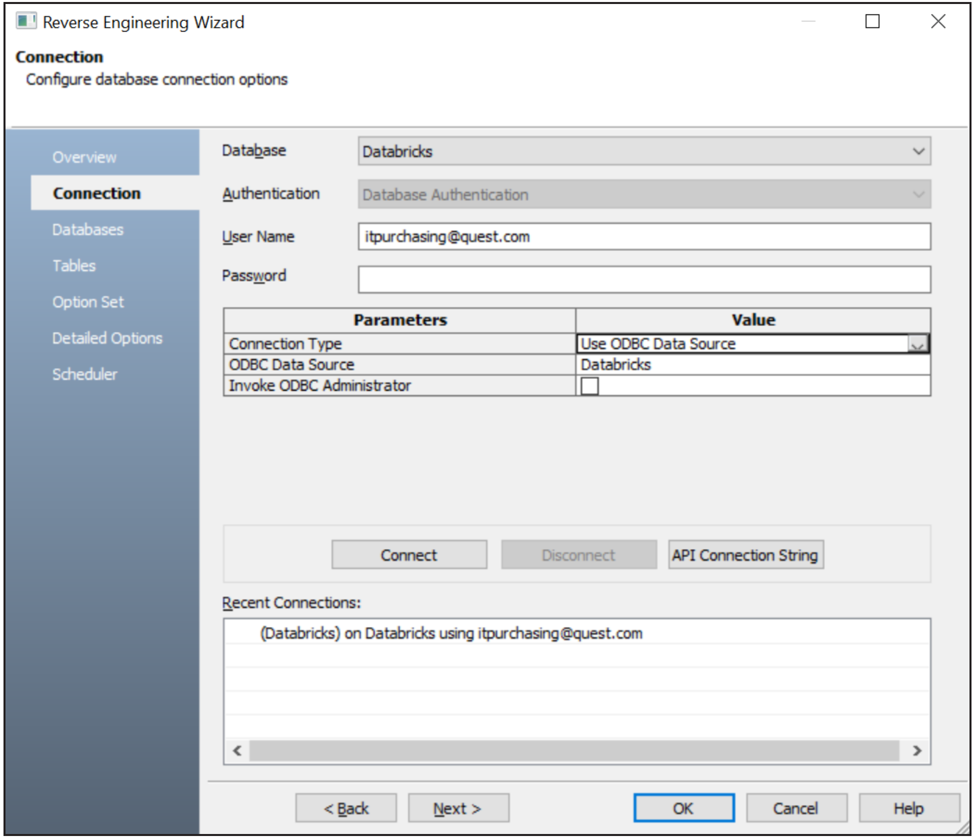
     If you click **Database**, continue to step 5.

   - **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 section appears.



Reverse Engineering Wizard

Connection

Configure database connection options

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	□

Recent Connections:

(Databricks) on Databricks using itpurchasing@quest.com

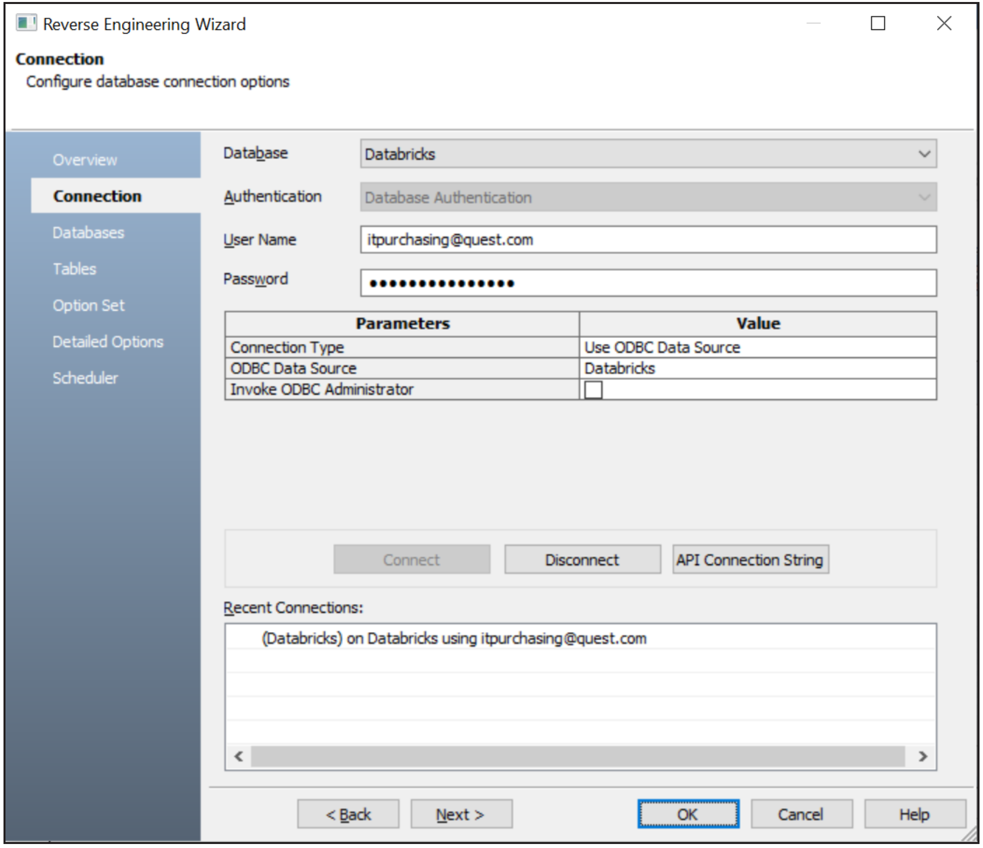
6. Enter your **User Name** and **Password**.

   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.	
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.	
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 method is set to JDBC Connection. For example, jdbc:spark://dbc-64e36c82-9e5d.cloud.databricks.com:443/default;transportMode=http;ssl=1;httpPathh=sql/protocolv1/o/2132616201277612/1108-064928-9gy4v7gf

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



Connection

Configure database connection options

Database: Databricks

Authentication: Database Authentication

User Name: itpurchasing@quest.com

Password: •••••••••••••••

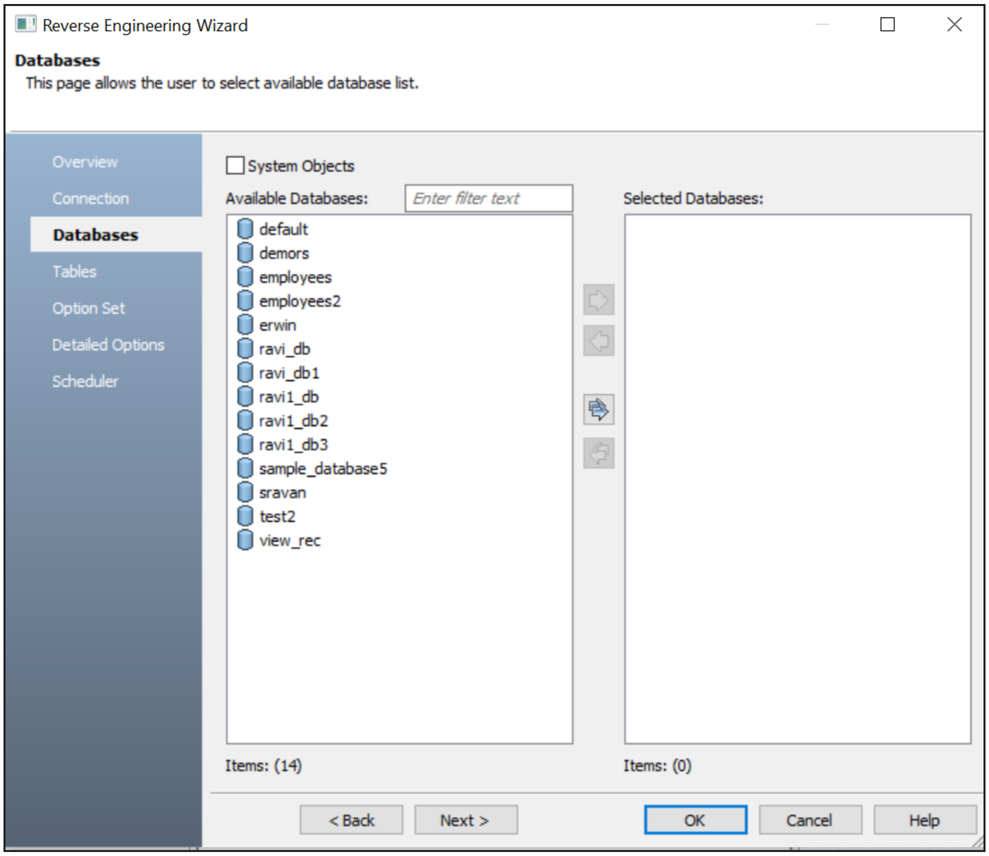
Parameters	Value
Connection Type	Use ODBC Data Source
ODBC Data Source	Databricks
Invoke ODBC Administrator	□

Recent Connections:

(Databricks) on Databricks using itpurchasing@quest.com

8. Click **Next**.

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



Reverse Engineering Wizard

Databases

This page allows the user to select available database list.

☐ System Objects

Available Databases: Enter filter text

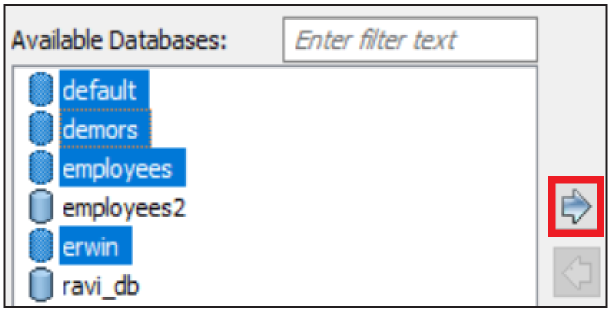
- default
- demors
- employees
- employees2
- erwin
- ravi_db
- ravi_db1
- ravi1_db
- ravi1_db2
- ravi1_db3
- sample_database5
- sravan
- test2
- view_rec

Items: (14)

Selected Databases:

Items: (0)

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

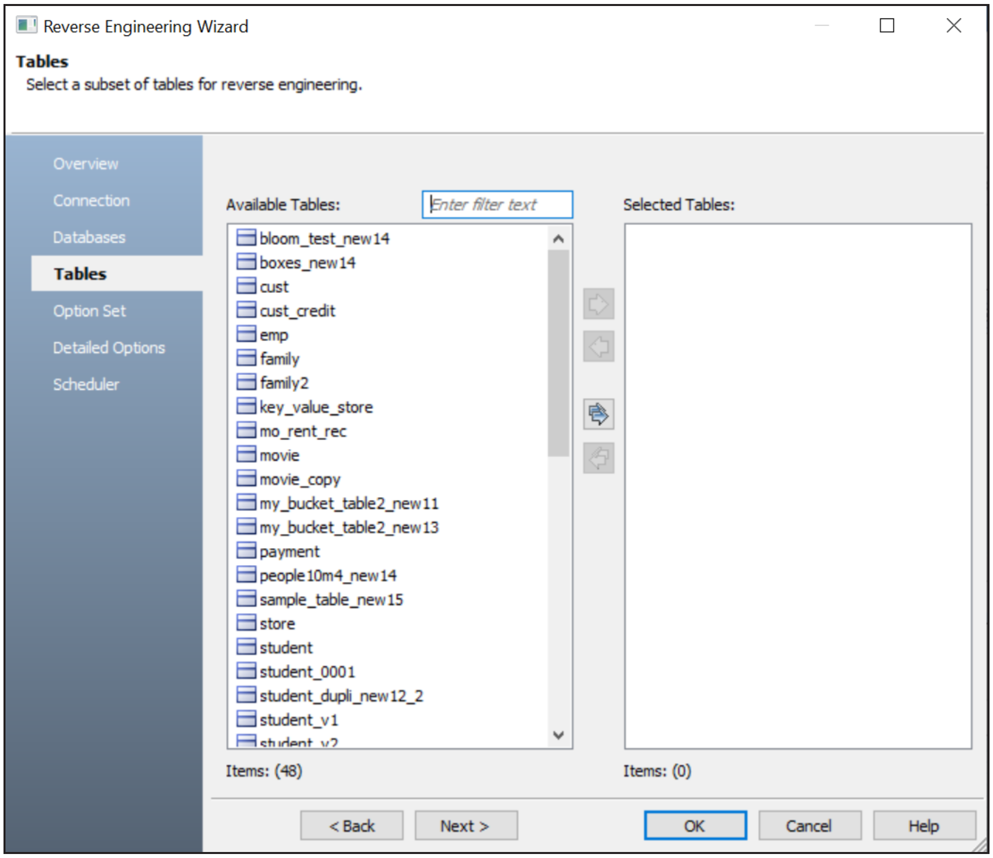


Available Databases: Enter filter text

- default
- demors
- employees
- employees2
- erwin
- ravi_db

10. Click **Next**.

    The Tables section appears. It displays a list of available tables in the databases that you selected in step 9.



Tables

Select a subset of tables for reverse engineering.

Available Tables: *Enter filter text*

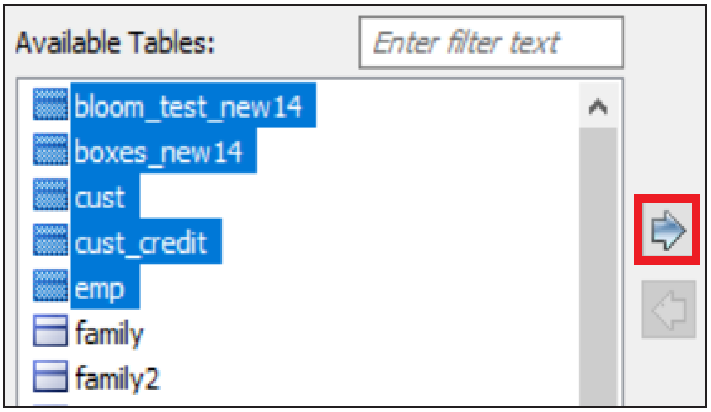
bloom_test_new14
boxes_new14
cust
cust_credit
emp
family
family2
key_value_store
mo_rent_rec
movie
movie_copy
my_bucket_table2_new11
my_bucket_table2_new13
payment
people10m4_new14
sample_table_new15
store
student
student_0001
student_dupli_new12_2
student_v1
student_v2

Items: (48)

Selected Tables:

Items: (0)

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

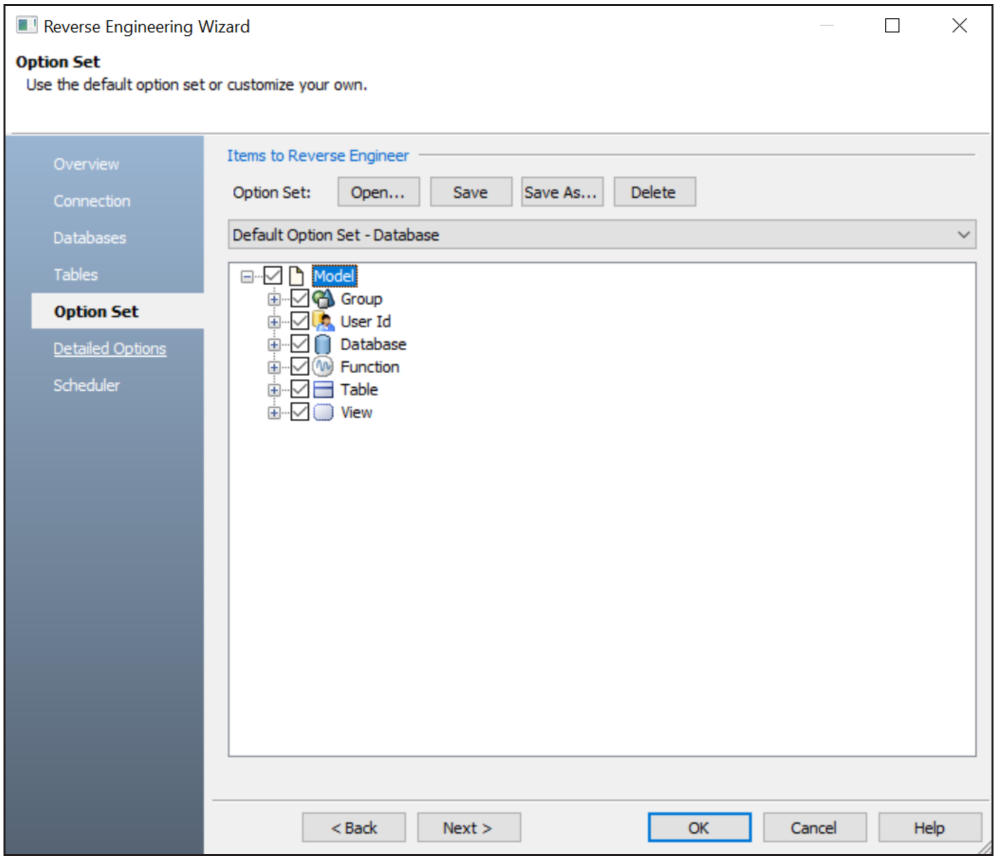


Available Tables: *Enter filter text*

bloom_test_new14
boxes_new14
cust
cust_credit
emp
family
family2

12. Click **Next**.

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



Reverse Engineering Wizard

Option Set

Use the default option set or customize your own.

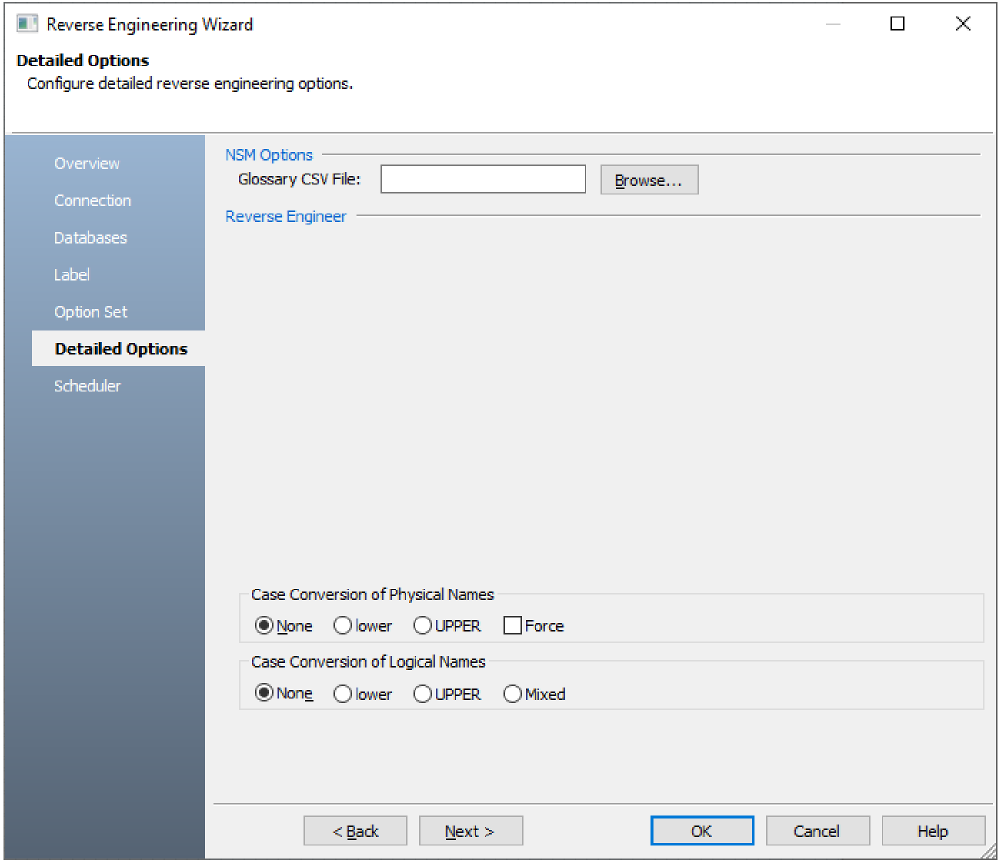
Items to Reverse Engineer

Option Set: Default Option Set - Database

- ☑ Model
  - ☑ Group
  - ☑ User Id
  - ☑ Database
  - ☑ Function
  - ☑ Table
  - ☑ View

13. Click **Next**.

    The Detailed Options section appears. Set up appropriate options based on your requirement.



Reverse Engineering Wizard

Detailed Options

Configure detailed reverse engineering options.

NSM Options

Glossary CSV File:

Reverse Engineer

Case Conversion of Physical Names

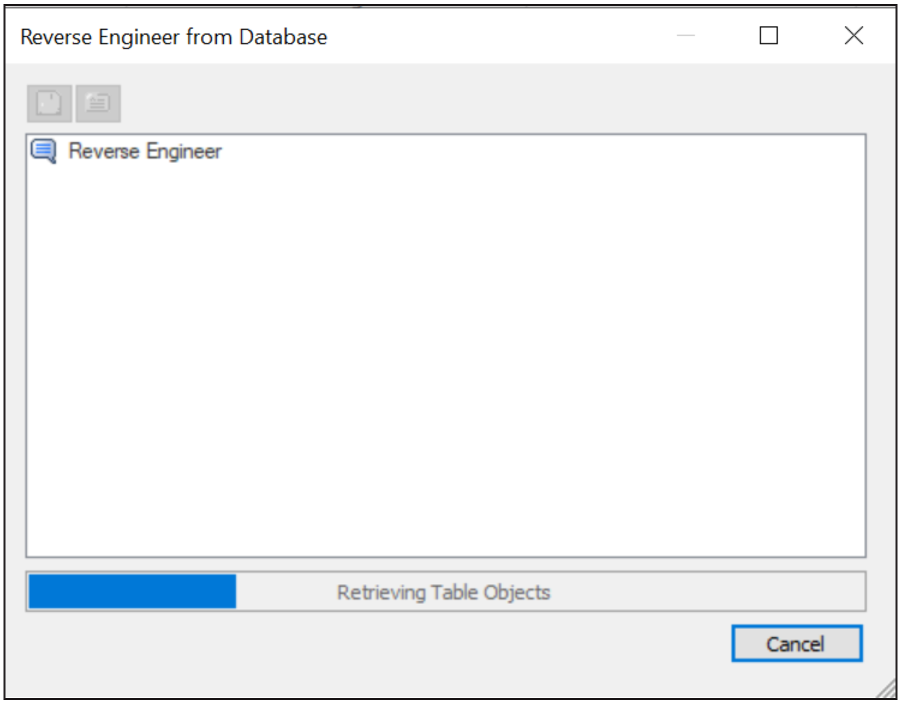
● None ○ lower ○ UPPER □ Force

Case Conversion of Logical Names

● None ○ lower ○ UPPER ○ Mixed

14. Click **OK**.

The reverse engineering process starts.

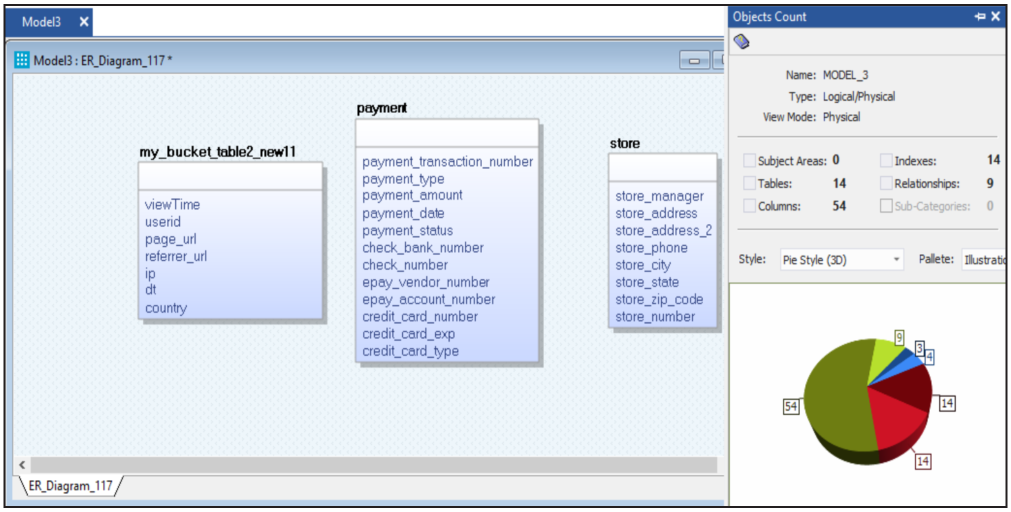


Reverse Engineer from Database

Reverse Engineer

Retrieving Table Objects

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



my_bucket_table2_new11
- viewTime
- userid
- page_url
- referrer_url
- ip
- dt
- country

payment
- payment_transaction_number
- payment_type
- payment_amount
- payment_date
- payment_status
- check_bank_number
- check_number
- epay_vendor_number
- epay_account_number
- credit_card_number
- credit_card_exp
- credit_card_type

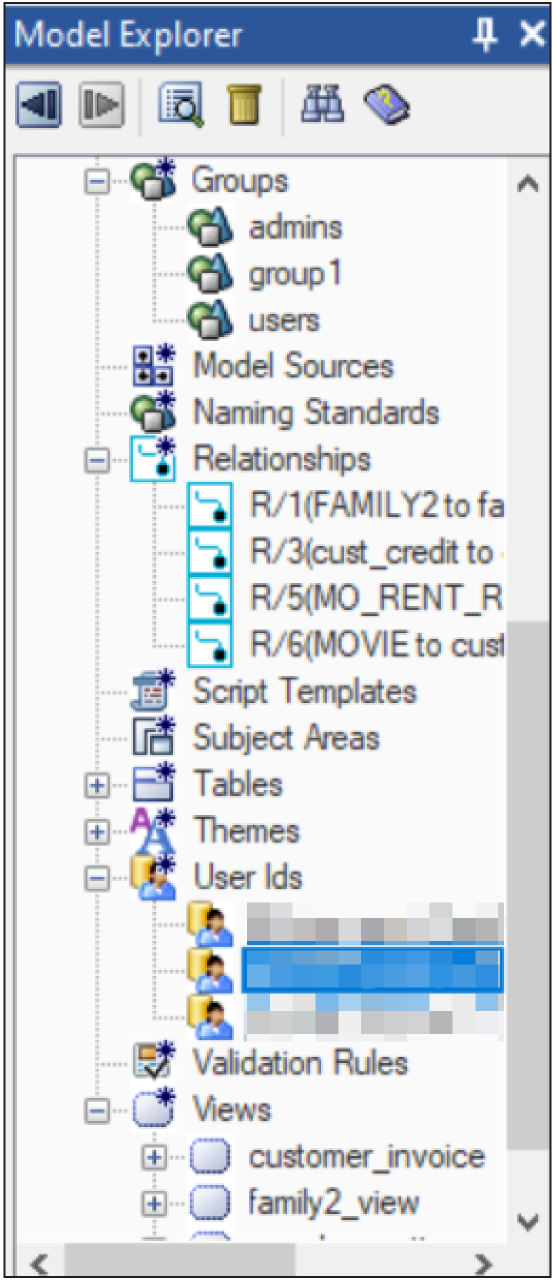
store
- store_manager
- store_address
- store_address_2
- store_phone
- store_city
- store_state
- store_zip_code
- store_number

Name: MODEL_3
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	0	□ Indexes:	14
□ Tables:	14	□ Relationships:	9
□ Columns:	54	□ Sub-Categories:	0

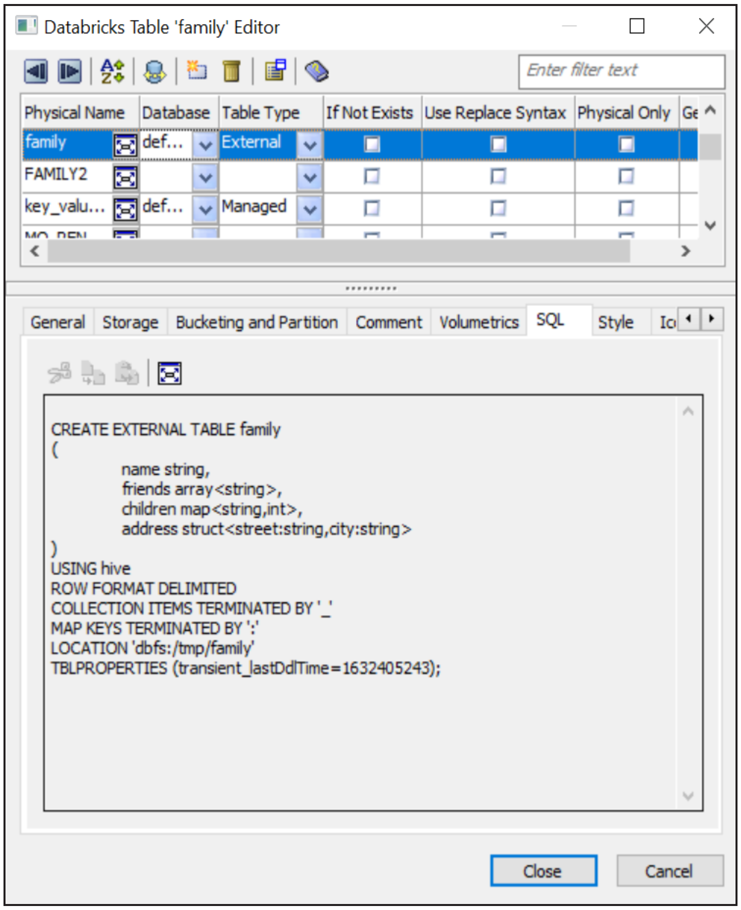
Style: Pie Style (3D)   Pallete: Illustrati

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.



Physical Name	Database	Table Type	If Not Exists	Use Replace Syntax	Physical Only	Ge
family	def...	External	□	□	□	
FAMILY2			□	□	□	
key_valu...	def...	Managed	□	□	□	

```
CREATE EXTERNAL TABLE family
(
            name string,
            friends array<string>,
            children map<string,int>,
            address struct<street:string,city:string>
)
USING hive
ROW FORMAT DELIMITED
COLLECTION ITEMS TERMINATED BY '_'
MAP KEYS TERMINATED BY ':'
LOCATION 'dbfs:/tmp/family'
TBLPROPERTIES (transient_lastDdlTime=1632405243);
```

Reverse Engineering Options for Databricks

The following are the reverse engineering options for Databricks 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 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.	
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.	
Connection	Specifies the connection string based on your JDBC instance in the following	This option is available only when Connection method is set to JDBC Connection.

String	format: *jdbc:spark://<server-host-name>:443/default;transportModee=http;ssl=1;httpPath=<http-path>*	For example, jdbc:spark://dbc-64e36c82-9e5d.cloud.databricks.com:443/default;transportMode=http;ssl=1;httpPath=sql/protocolv1/o/2132616201277612/1108-064928-9gy4v7gf

Databases

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

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	**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 where the reverse engineered model should be saved and its name	When you schedule a job on a remote server, ensure the model path is same for remote and local server. For example: C:\Scheduler\<Model Name>.erwin
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.

Forward Engineering Models

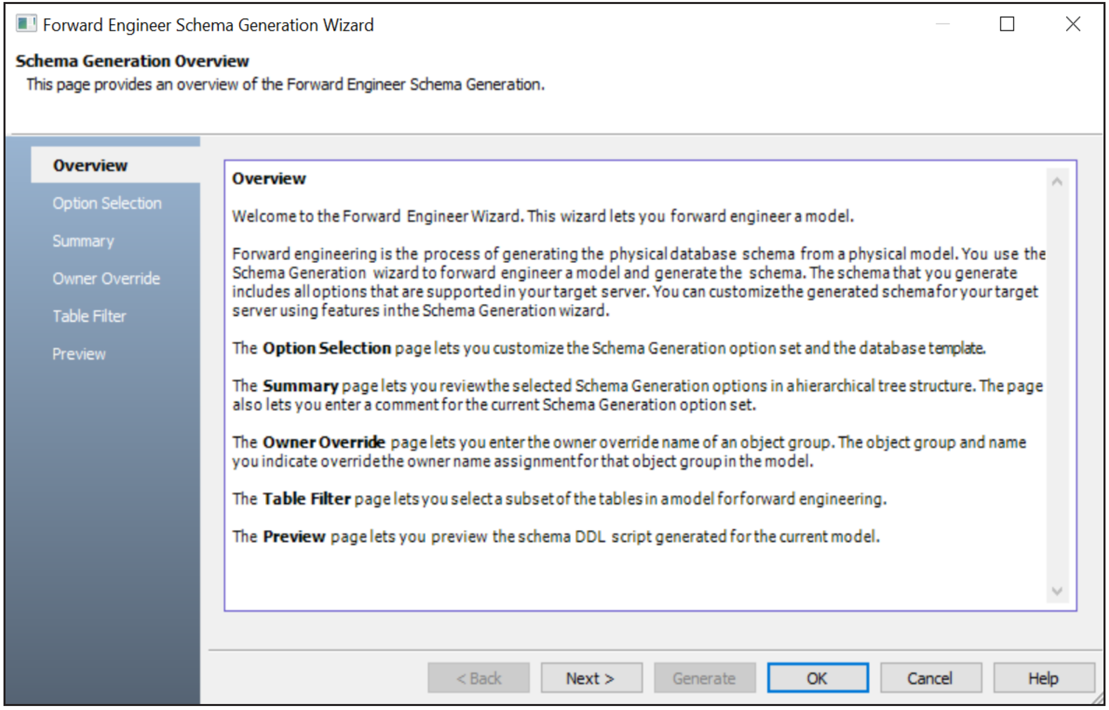
You can generate a physical database schema from a physical model using the Forward Engineering process. This topic walks you through the steps to forward engineer a Databricks model. For detailed description of forward engineering options, refer to the Forward Engineering Options topic.

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.



Schema Generation Overview

This page provides an overview of the Forward Engineer Schema Generation.

Overview

Welcome to the Forward Engineer Wizard. This wizard lets you forward engineer a model.

Forward engineering is the process of generating the physical database schema from a physical model. You use the Schema Generation wizard to forward engineer a model and generate the schema. The schema that you generate includes all options that are supported in your target server. You can customize the generated schema for your target server using features in the Schema Generation wizard.

The **Option Selection** page lets you customize the Schema Generation option set and the database template.

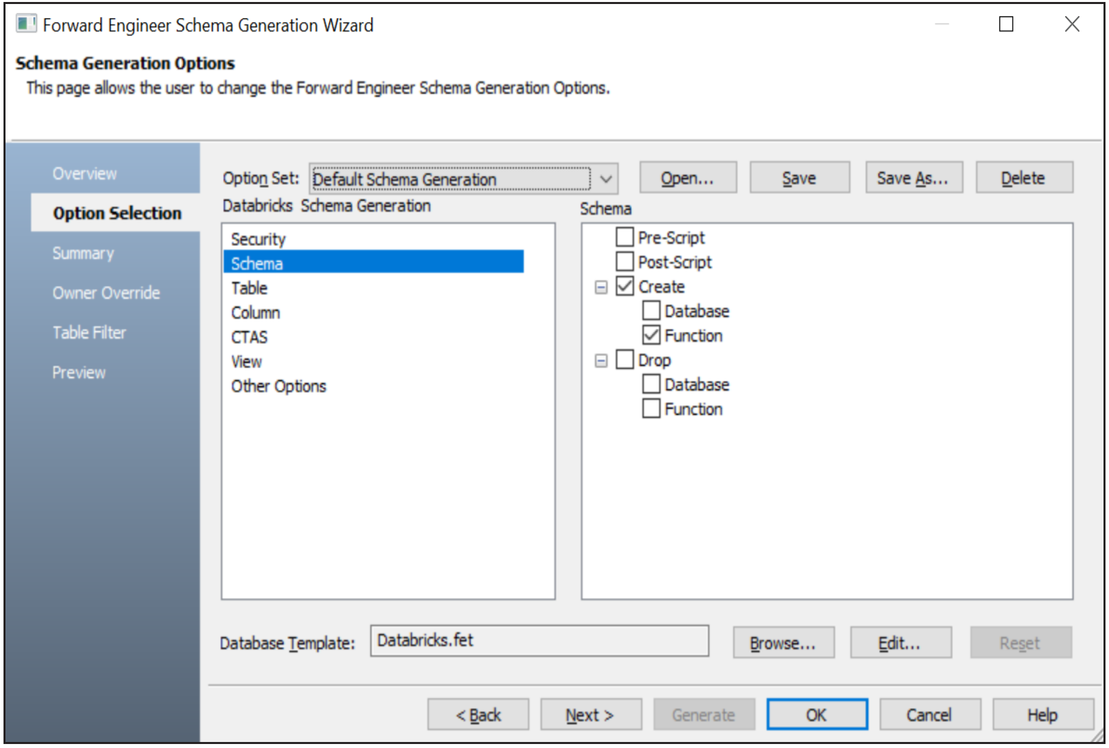
The **Summary** page lets you review the selected Schema Generation options in a hierarchical tree structure. The page also lets you enter a comment for the current Schema Generation option set.

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

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

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

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



Forward Engineer Schema Generation Wizard

Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default Schema Generation

Databricks Schema Generation

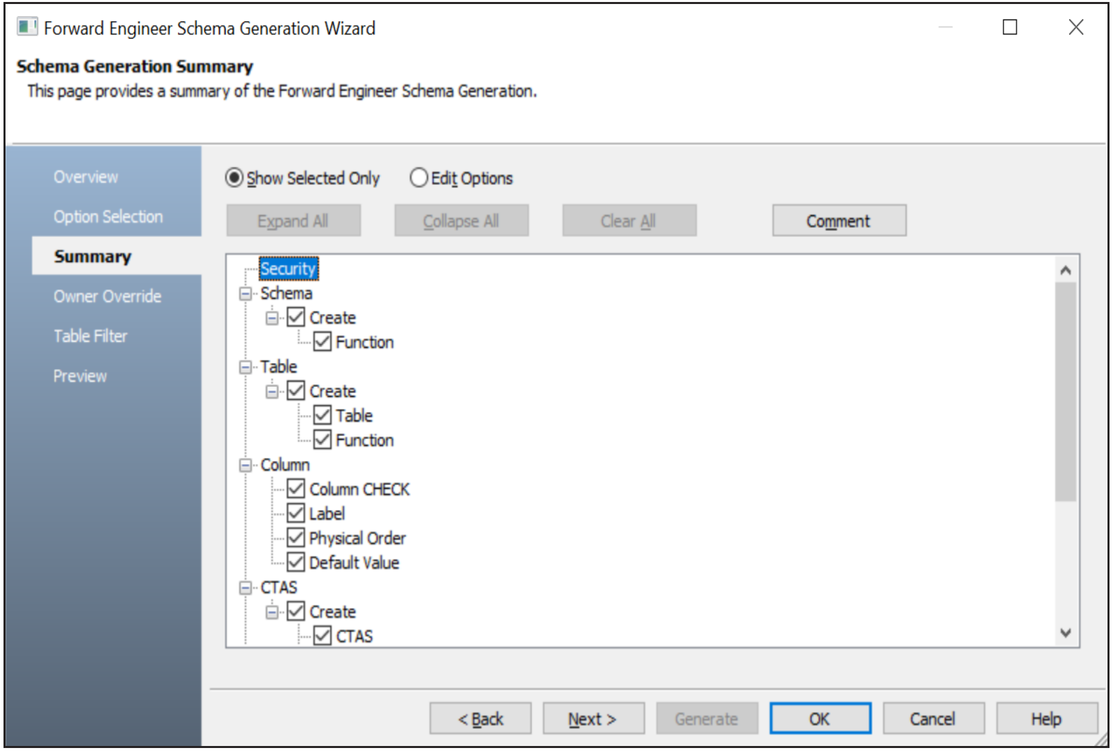
Schema

- □ Pre-Script
- □ Post-Script
- ☑ Create
  - □ Database
  - ☑ Function
- □ Drop
  - □ Database
  - □ Function

Database Template: Databricks.fet

4. Click **Next**.

   The Summary section appears. It displays a list of selected options for the schema generation. Use Edit Options to update selected options.



Forward Engineer Schema Generation Wizard

Schema Generation Summary

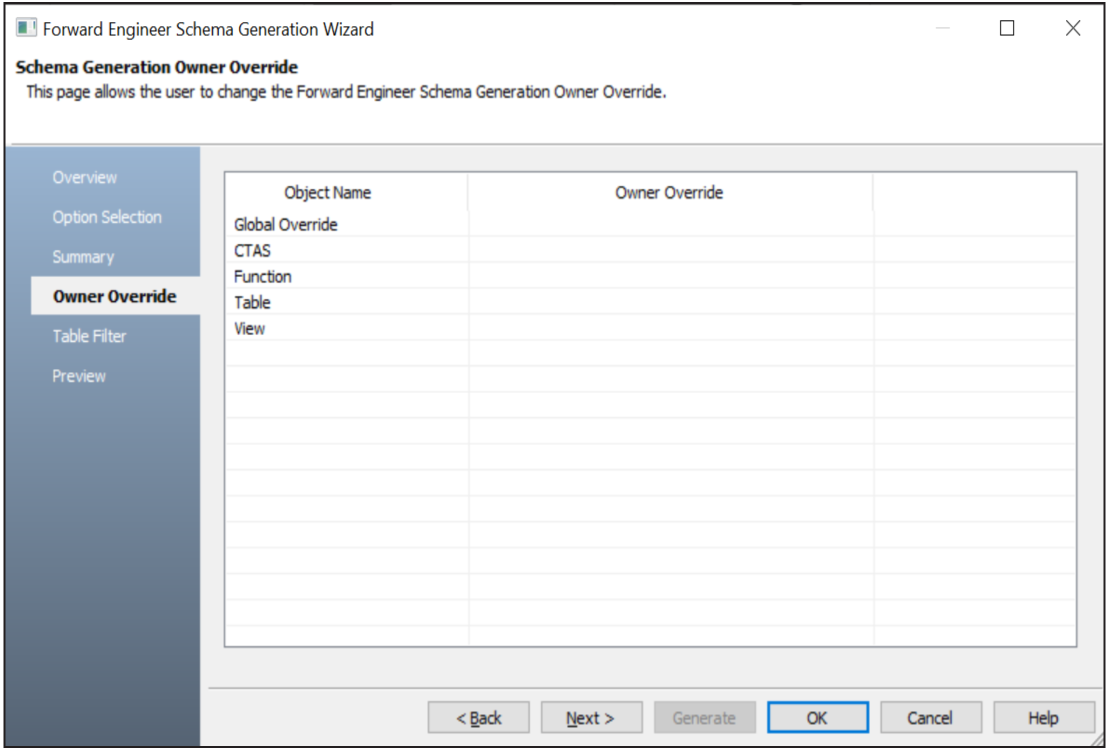
This page provides a summary of the Forward Engineer Schema Generation.

● Show Selected Only ○ Edit Options

- Security
- Schema
  - ☑ Create
    - ☑ Function
- Table
  - ☑ Create
    - ☑ Table
    - ☑ Function
- Column
  - ☑ Column CHECK
  - ☑ Label
  - ☑ Physical Order
  - ☑ Default Value
- CTAS
  - ☑ Create
    - ☑ CTAS

5. Click **Next**.

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



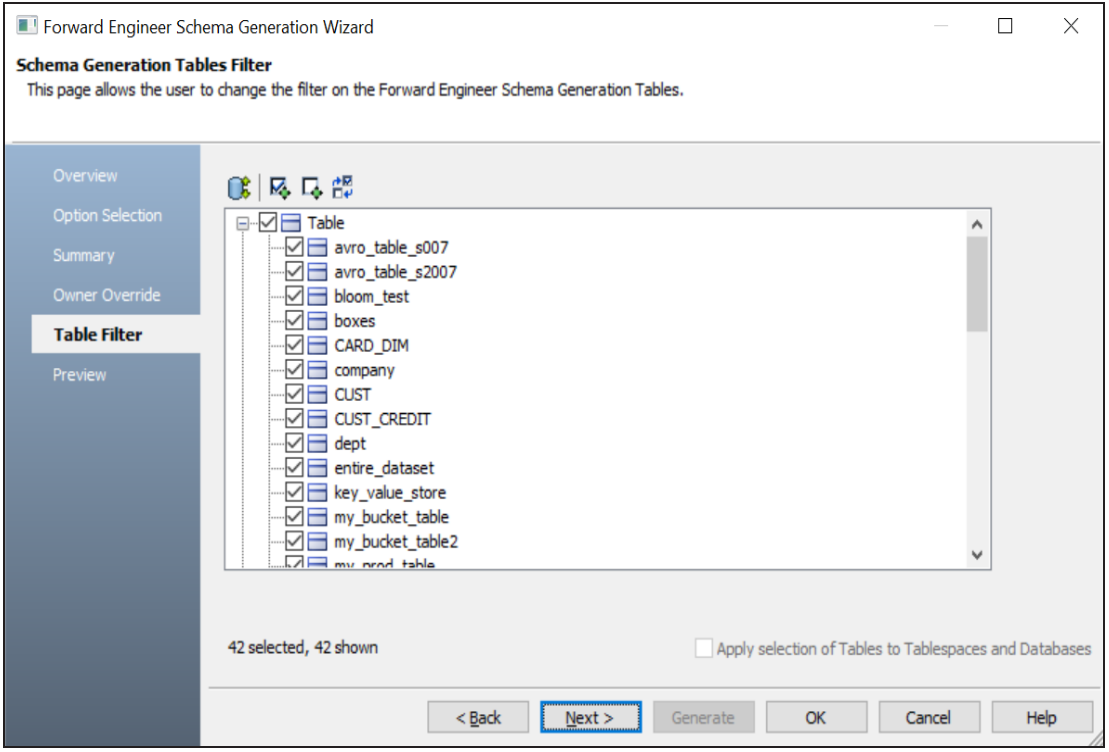
Schema Generation Owner Override

This page allows the user to change the Forward Engineer Schema Generation Owner Override.

Object Name	Owner Override
Global Override	
CTAS	
Function	
Table	
View	

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

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



Forward Engineer Schema Generation Wizard

Schema Generation Tables Filter

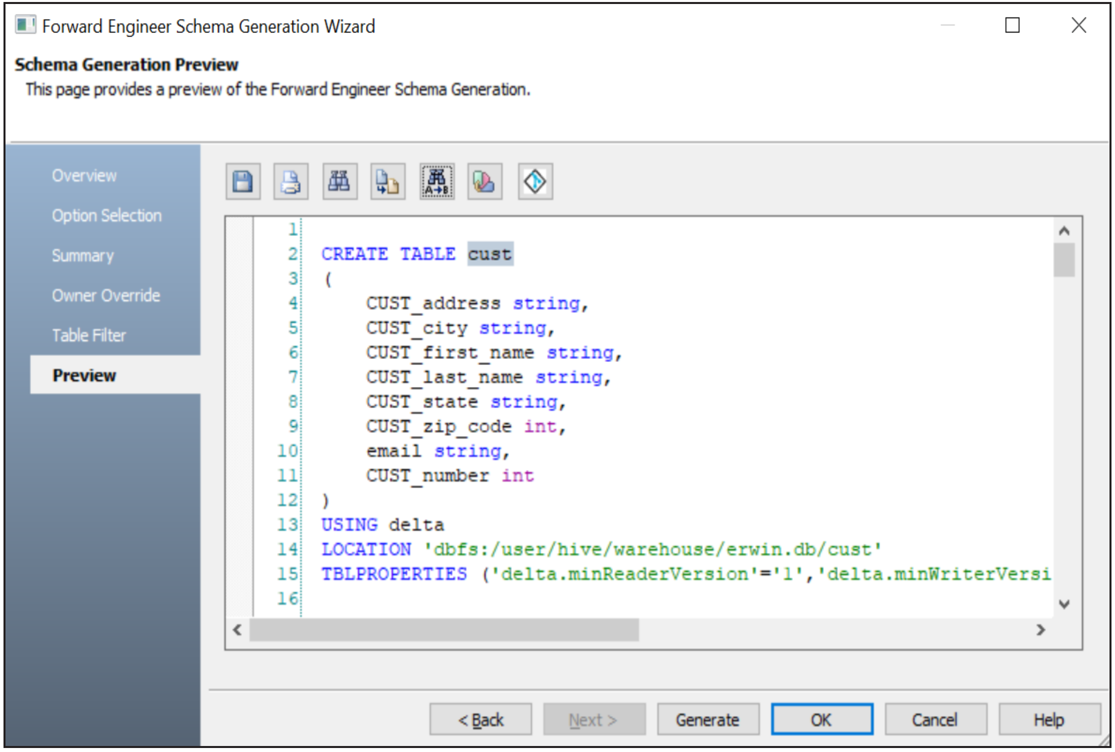
This page allows the user to change the filter on the Forward Engineer Schema Generation Tables.

- ☑ Table
  - ☑ avro_table_s007
  - ☑ avro_table_s2007
  - ☑ bloom_test
  - ☑ boxes
  - ☑ CARD_DIM
  - ☑ company
  - ☑ CUST
  - ☑ CUST_CREDIT
  - ☑ dept
  - ☑ entire_dataset
  - ☑ key_value_store
  - ☑ my_bucket_table
  - ☑ my_bucket_table2
  - ☑ my_prod_table

42 selected, 42 shown

□ Apply selection of Tables to Tablespaces and Databases

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



Schema Generation Preview

This page provides a preview of the Forward Engineer Schema Generation.

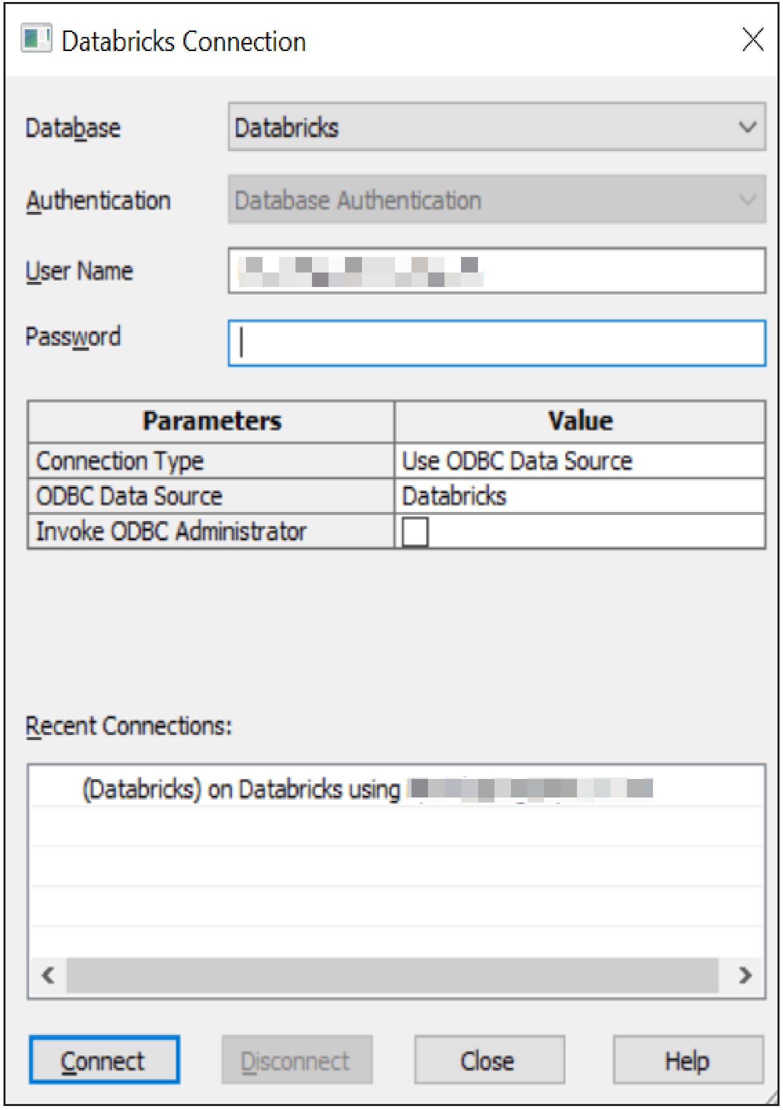
```

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
)
USING delta
LOCATION 'dbfs:/user/hive/warehouse/erwin.db/cust'
TBLPROPERTIES ('delta.minReaderVersion'='1','delta.minWriterVersi

```

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 page appears.



Database: Databricks

Authentication: Database Authentication

User Name: [redacted]

Password:

Parameters	Value
Connection Type	Use ODBC Data Source
ODBC Data Source	Databricks
Invoke ODBC Administrator	□

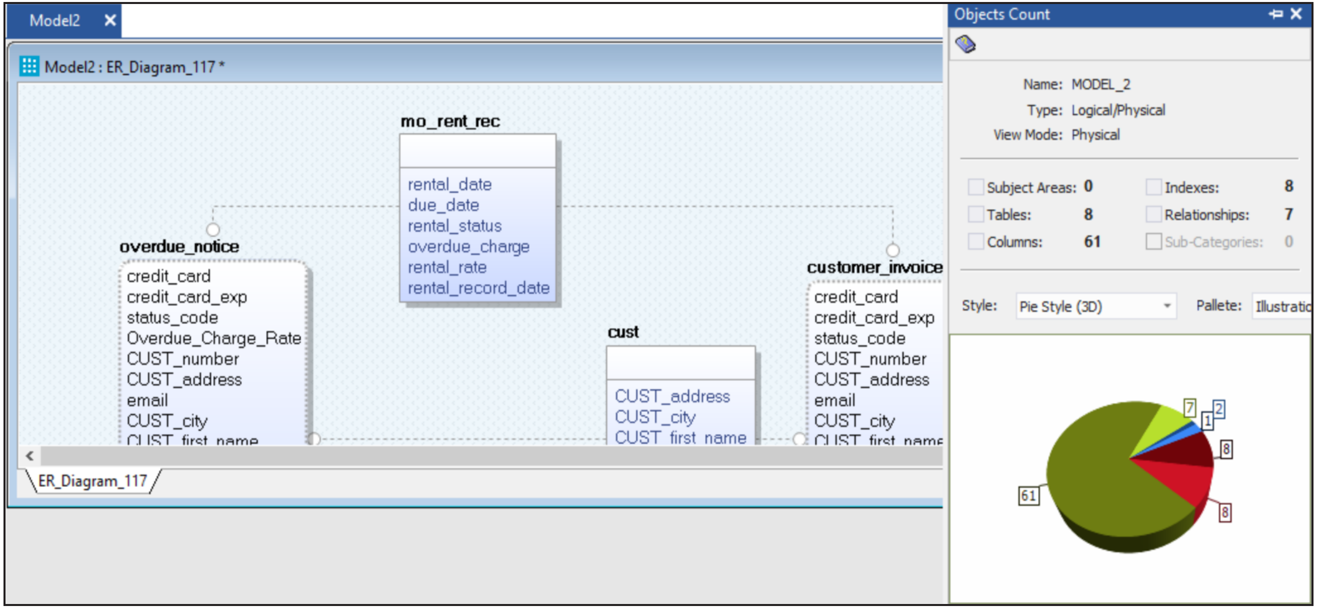
Recent Connections:

(Databricks) on Databricks using [redacted]

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.

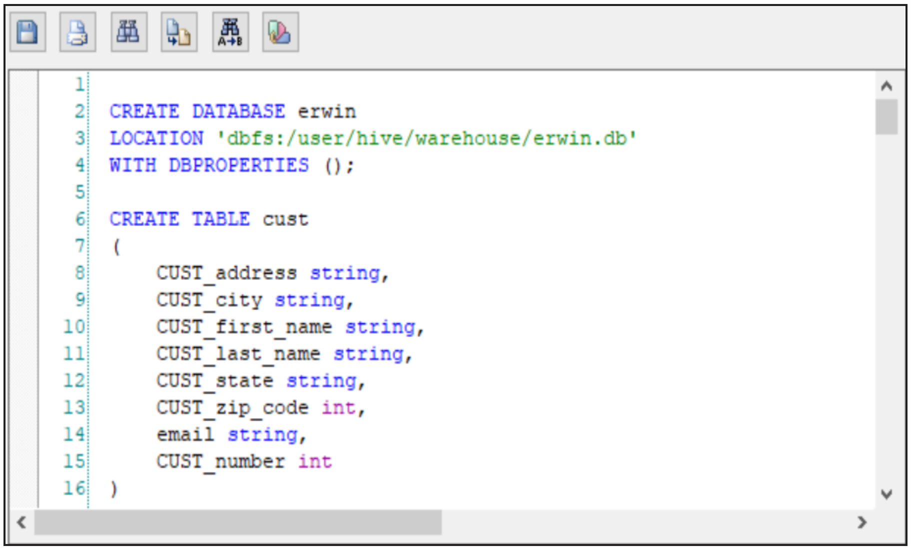


Name: MODEL_2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	0	□ Indexes:	8
□ Tables:	8	□ Relationships:	7
□ Columns:	61	□ Sub-Categories:	0

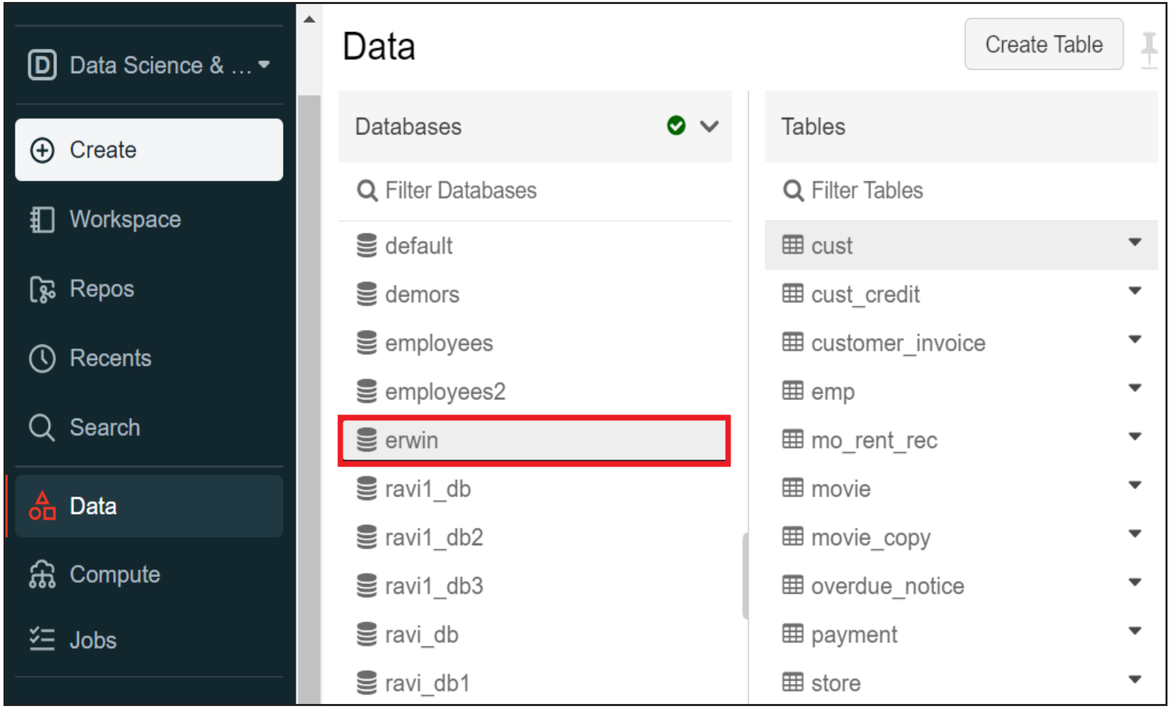
Style: Pie Style (3D) Pallete: Illustratio

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

The following are the forward engineering options for Databricks in erwin DM.

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.

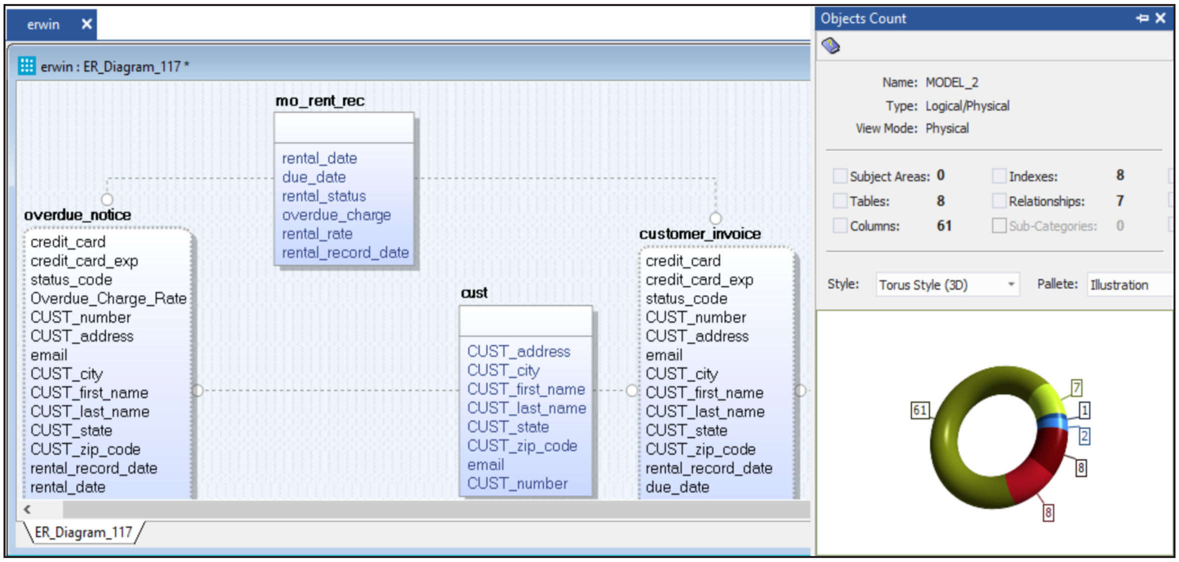
This topic walks you through the steps to compare a Databricks model with database.

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.

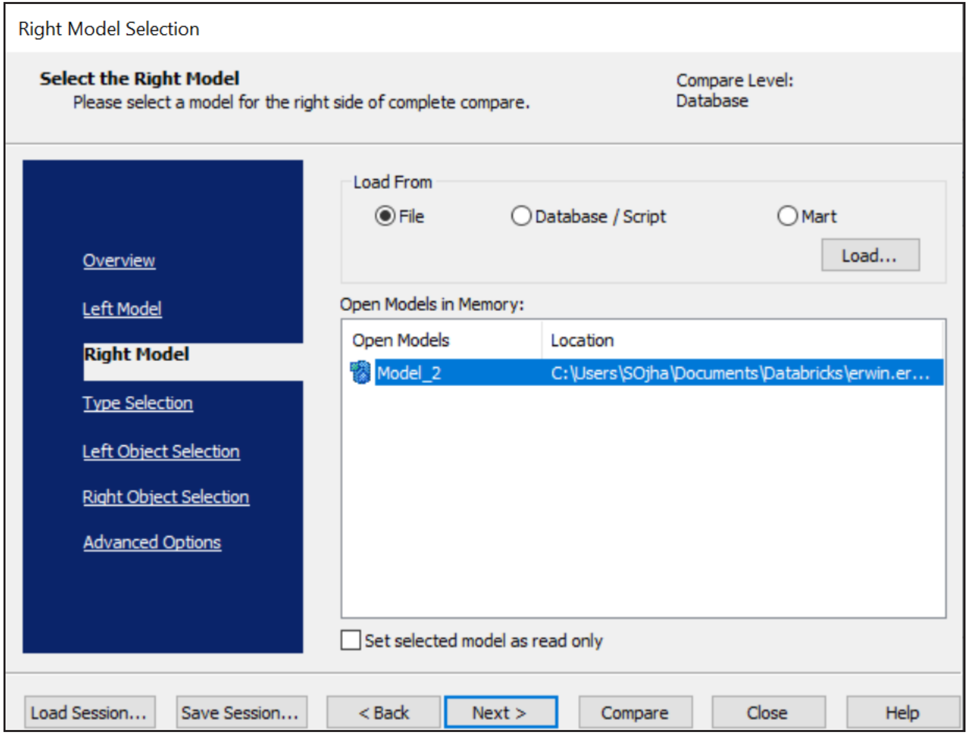


Name: MODEL_2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas:	0	□ Indexes:	8
□ Tables:	8	□ Relationships:	7
□ Columns:	61	□ Sub-Categories:	0

Style: Torus Style (3D) Pallete: Illustration

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.



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

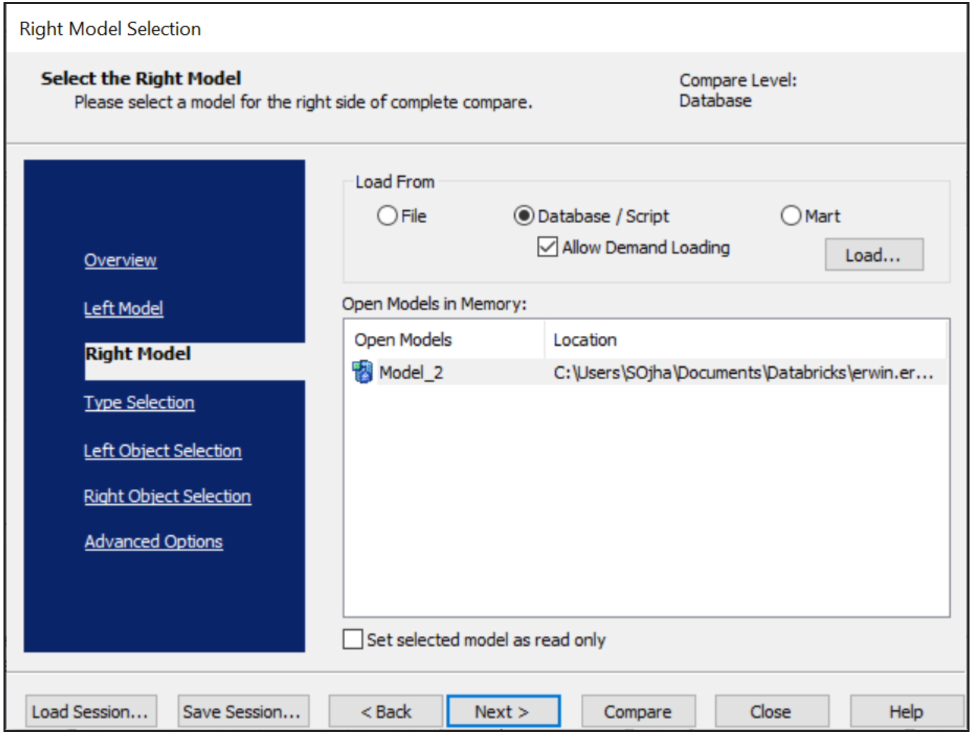
● File ○ Database / Script ○ Mart

Open Models in Memory:

Open Models	Location
Model_2	C:\Users\SOjha\Documents\Databricks\erwin.er...

□ Set selected model as read only

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


Right Model Selection

Select the Right Model
Please select a model for the right side of complete compare.

Compare Level:
Database

Load From
○ File ● Database / Script ○ Mart
☑ Allow Demand Loading

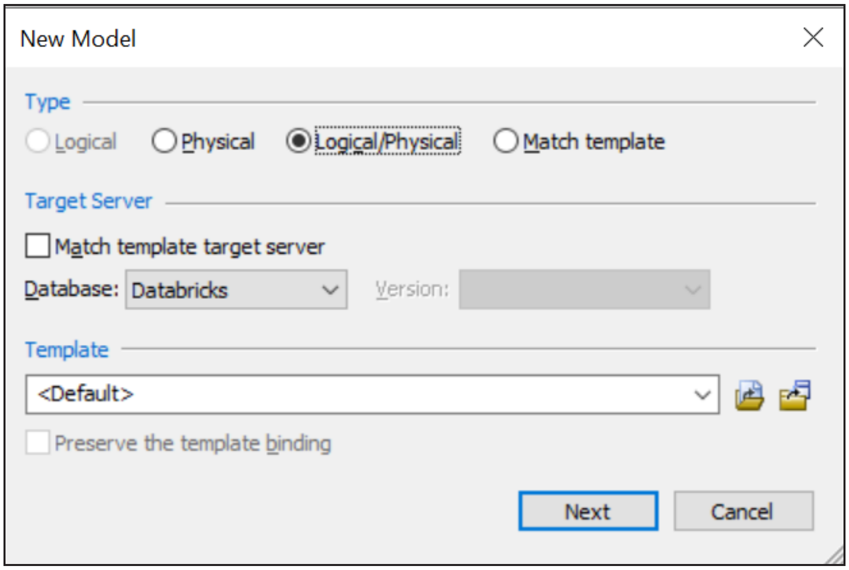
Open Models in Memory:

Open Models	Location
Model_2	C:\Users\SOjha\Documents\Databricks\erwin.er...

□ Set selected model as read only

4. Click **Load**.

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


New Model

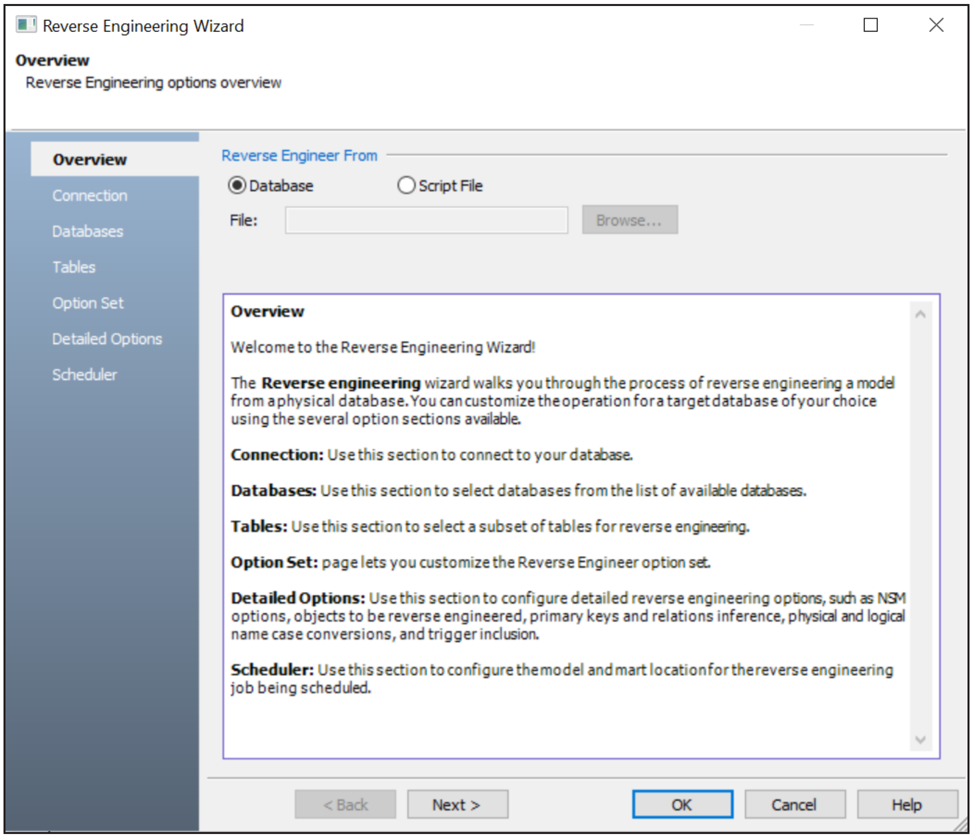
Type
○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server
□ Match template target server
Database: Databricks
Version:

Template
<Default>
□ Preserve the template binding

5. Ensure that the Database is set to Databricks. Then, click **Next**.

   The Reverse Engineer Wizard appears.



Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Databases: Use this section to select databases from the list of available databases.

Tables: Use this section to select a subset of tables for reverse engineering.

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

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

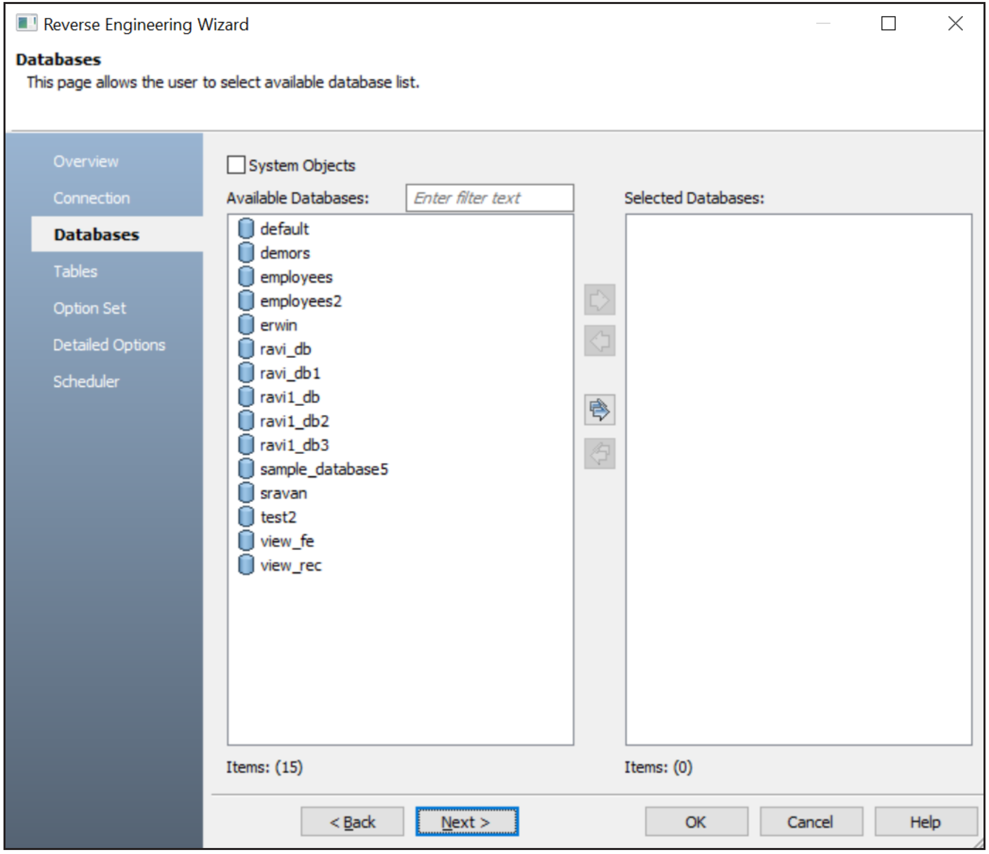
Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

6. Click **Database**. Then, click **Next**.

   The Connection section appears. Use this section to connect to the database from which you want to reverse engineer the model.

7. After connection is established, click **Next**.

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



Reverse Engineering Wizard

Databases

This page allows the user to select available database list.

□ System Objects

Available Databases: Enter filter text

- default
- demors
- employees
- employees2
- erwin
- ravi_db
- ravi_db1
- ravi1_db
- ravi1_db2
- ravi1_db3
- sample_database5
- sravan
- test2
- view_fe
- view_rec

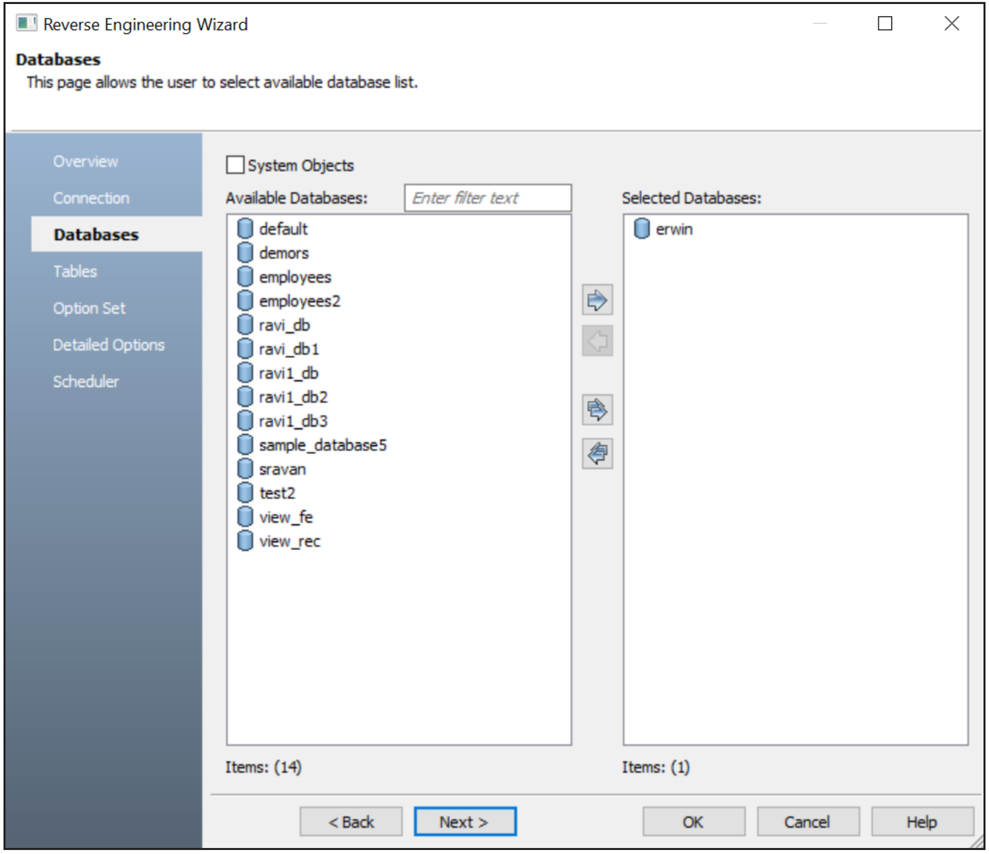
Items: (15)

Selected Databases:

Items: (0)

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

   This moves the selected databases under Selected Databases.



Databases

This page allows the user to select available database list.

□ System Objects

Available Databases: Enter filter text

- default
- demors
- employees
- employees2
- ravi_db
- ravi_db1
- ravi1_db
- ravi1_db2
- ravi1_db3
- sample_database5
- sravan
- test2
- view_fe
- view_rec

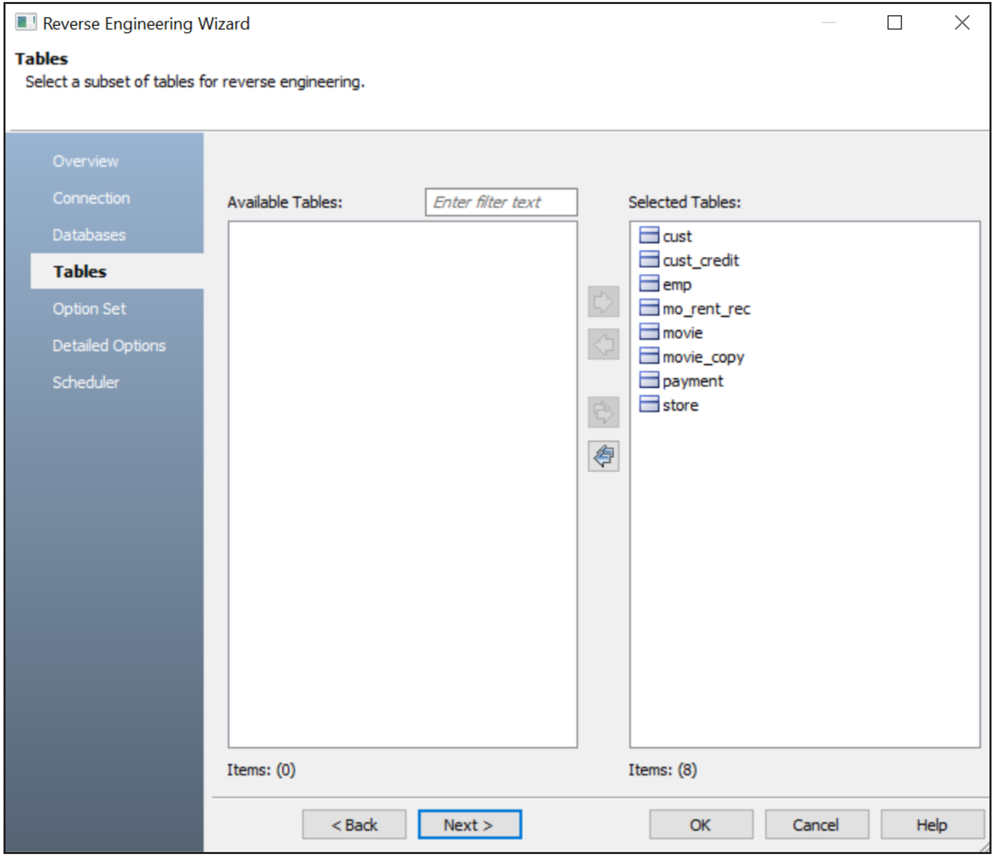
Items: (14)

Selected Databases:

- erwin

Items: (1)

9. Click **Next** and in the Tables section, click ⇛.
   This selects all the available tables.



Tables

Select a subset of tables for reverse engineering.

Available Tables: Enter filter text

Items: (0)

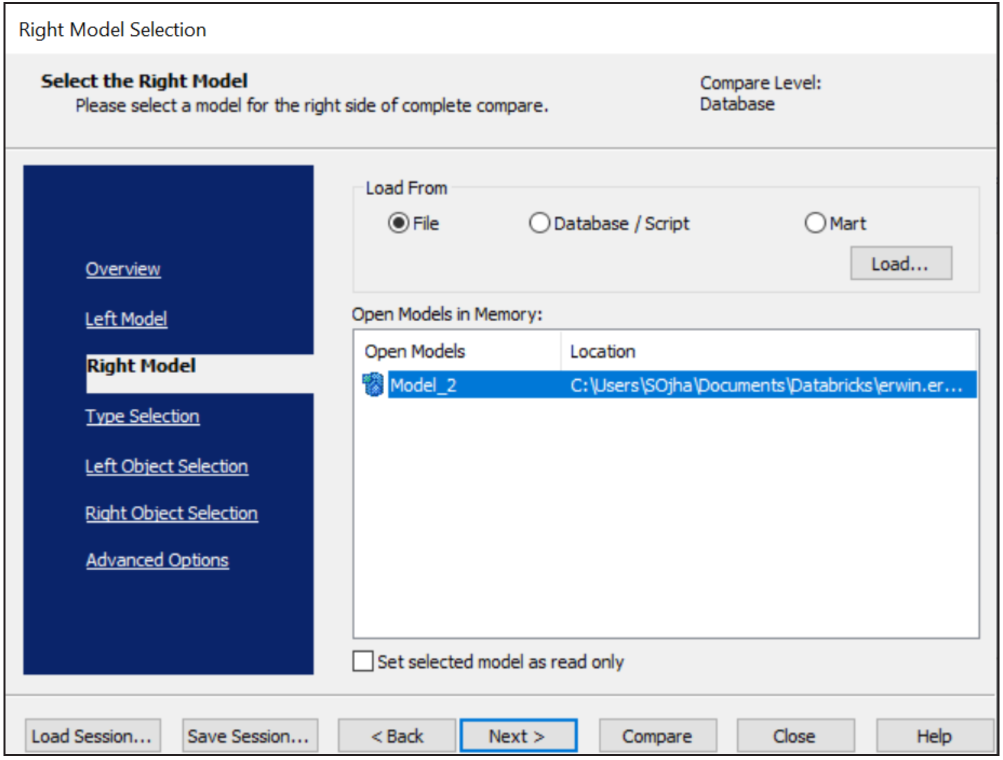
Selected Tables:

- cust
- cust_credit
- emp
- mo_rent_rec
- movie
- movie_copy
- payment
- store

Items: (8)

10. Click **Next** and in the Option Set section, keep the default configuration.
11. Click **Next** and in the Detailed Options section, keep the default configuration.
12. Click **OK**.

    The reverse engineering process starts. Once the process is complete, the Right Model is set to the one that you reverse engineered.



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level:
Database

Load From

● File ○ Database / Script ○ Mart

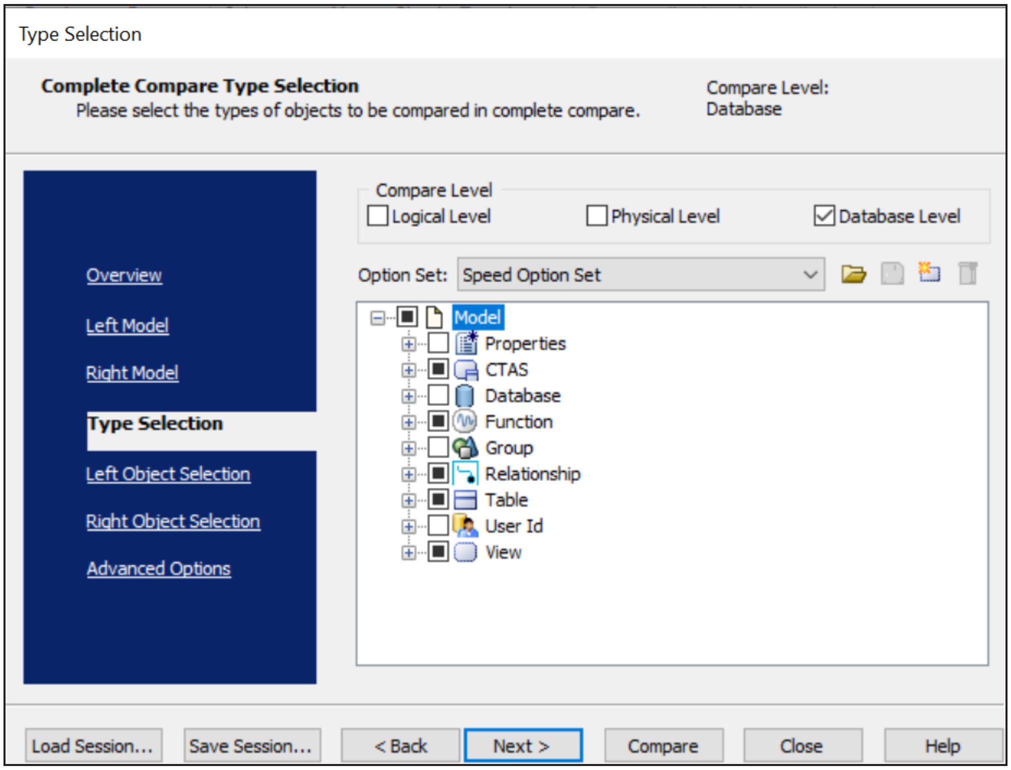
Open Models in Memory:

Open Models	Location
Model_2	C:\Users\SOjha\Documents\Databricks\erwin.er...

□ Set selected model as read only

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

For example, the following image shows the default options.



Type Selection

Complete Compare Type Selection

Please select the types of objects to be compared in complete compare.

Compare Level: Database

Compare Level

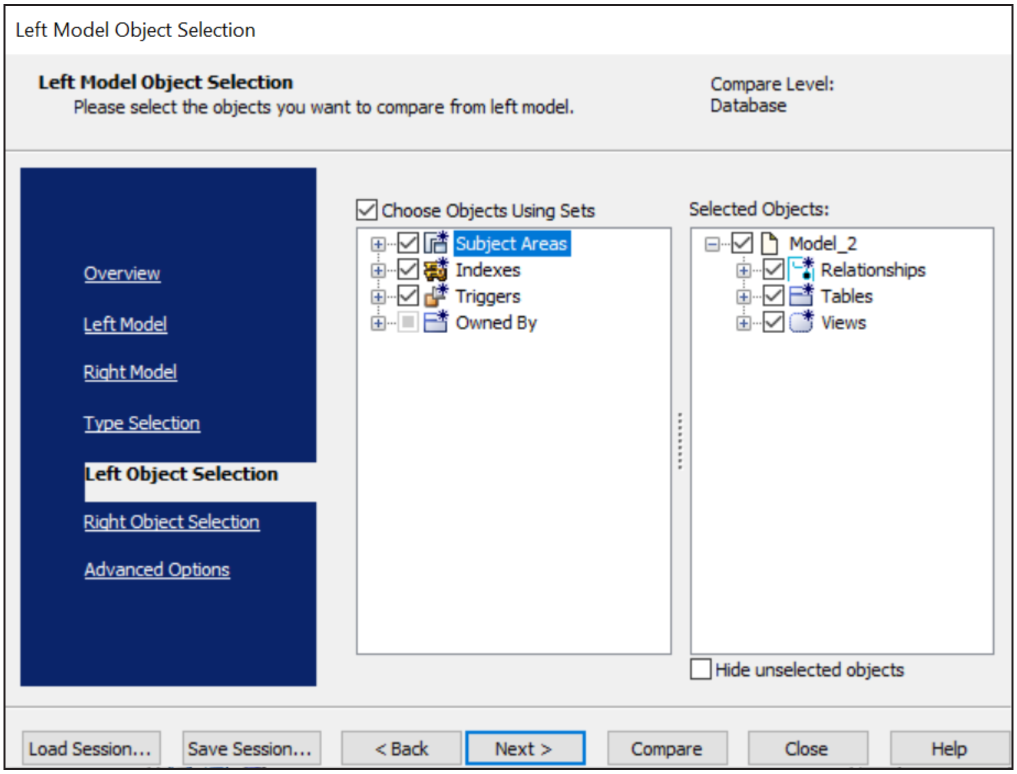
□ Logical Level □ Physical Level ☑ Database Level

Option Set: Speed Option Set

- ■ Model
  - □ Properties
  - ■ CTAS
  - □ Database
  - ■ Function
  - □ Group
  - ■ Relationship
  - ■ Table
  - □ User Id
  - ■ View

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

For example, the following image shows the default options.



Left Model Object Selection

Left Model Object Selection
Please select the objects you want to compare from left model.

Compare Level:
Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

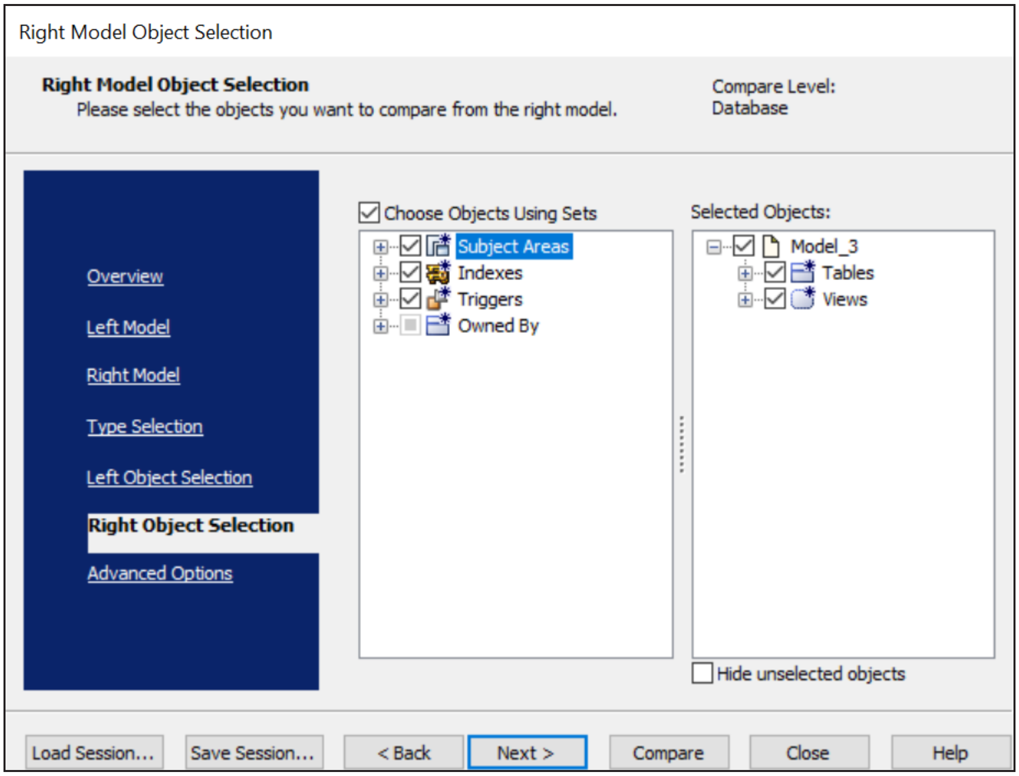
Selected Objects:

- ☑ Model_2
  - ☑ Relationships
  - ☑ Tables
  - ☑ Views

□ Hide unselected objects

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

For example, the following image shows the default options.



Right Model Object Selection

Right Model Object Selection
Please select the objects you want to compare from the right model.

Compare Level:
Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

Selected Objects:

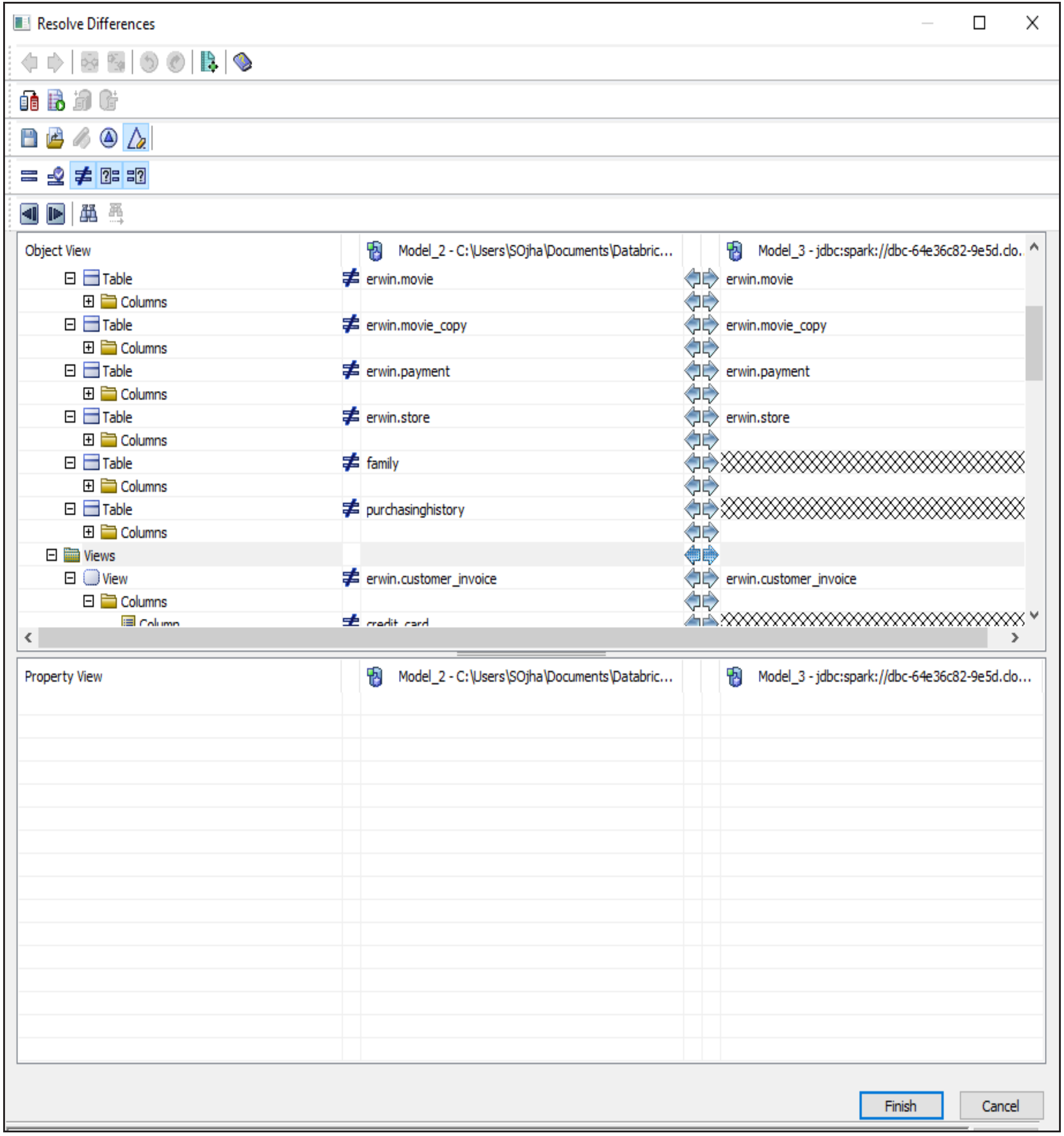
- ☑ Model_3
  - ☑ Tables
  - ☑ Views

□ Hide unselected objects

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.



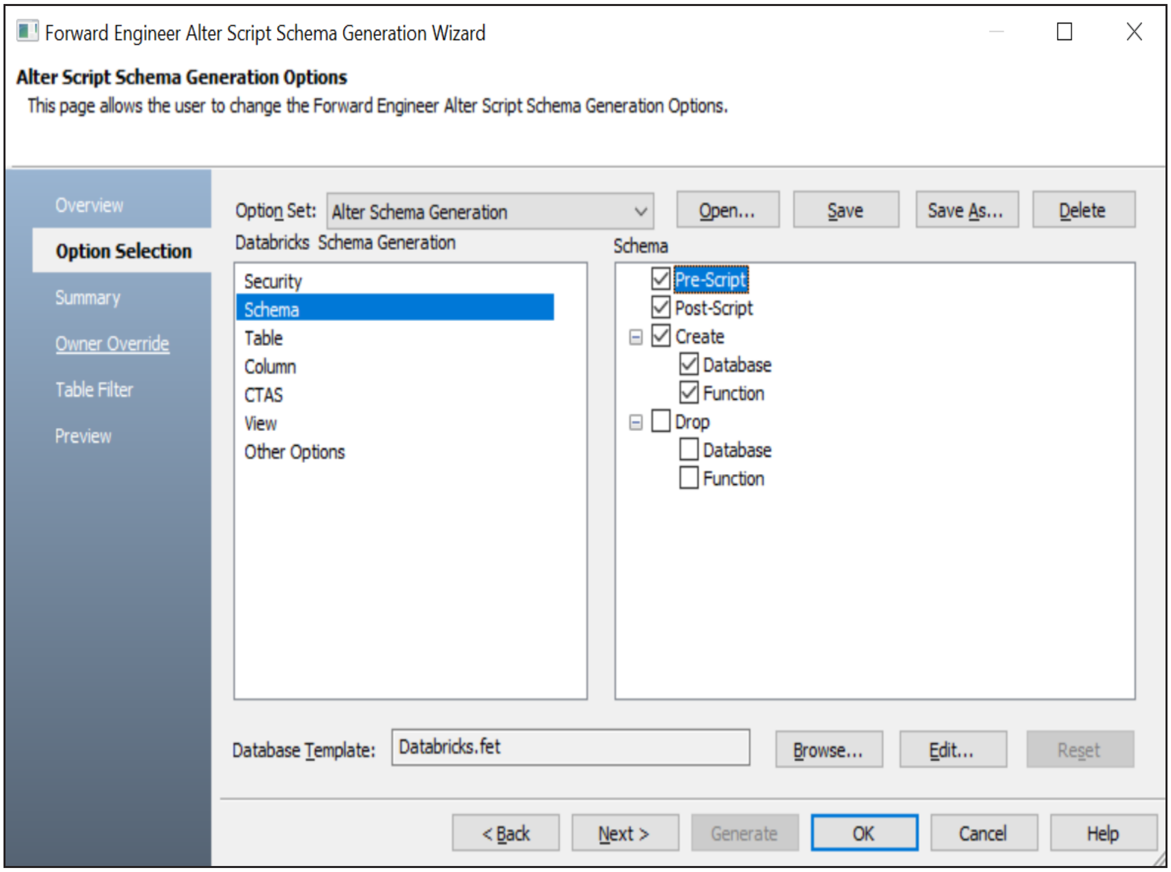
Object View	Model_2 - C:\Users\SOjha\Documents\Databric...	Model_3 - jdbc:spark://dbc-64e36c82-9e5d.clo...
Table	≠ erwin.movie	erwin.movie
Columns		
Table	≠ erwin.movie_copy	erwin.movie_copy
Columns		
Table	≠ erwin.payment	erwin.payment
Columns		
Table	≠ erwin.store	erwin.store
Columns		
Table	≠ family	▨
Columns		
Table	≠ purchasinghistory	▨
Columns		
Views		
View	≠ erwin.customer_invoice	erwin.customer_invoice
Columns		
Column	≠ credit_card	▨

Property View	Model_2 - C:\Users\SOjha\Documents\Databric...	Model_3 - jdbc:spark://dbc-64e36c82-9e5d.clo...

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.



Forward Engineer Alter Script Schema Generation Wizard

Alter Script Schema Generation Options

This page allows the user to change the Forward Engineer Alter Script Schema Generation Options.

Option Set: Alter Schema Generation

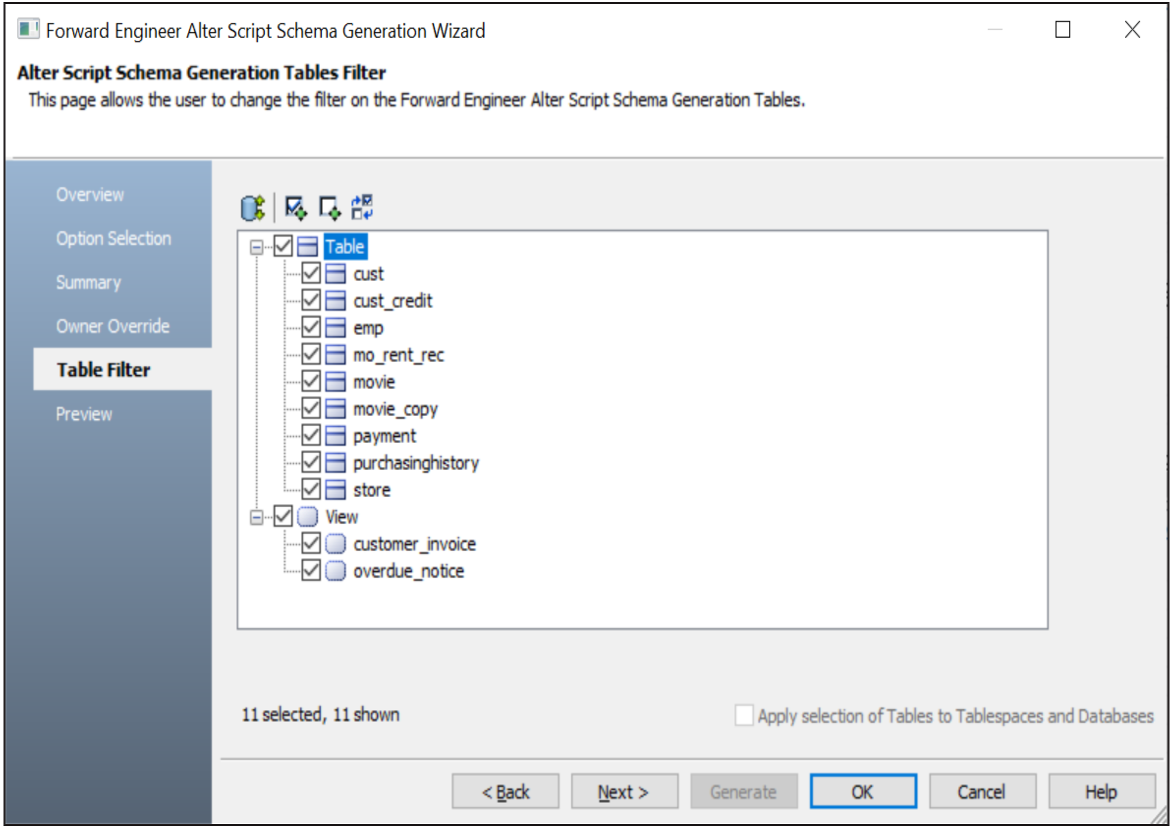
Databricks Schema Generation

Schema

- ☑ Pre-Script
- ☑ Post-Script
- ☑ Create
  - ☑ Database
  - ☑ Function
- □ Drop
  - □ Database
  - □ Function

Database Template: Databricks.fet

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



Forward Engineer Alter Script Schema Generation Wizard

Alter Script Schema Generation Tables Filter

This page allows the user to change the filter on the Forward Engineer Alter Script Schema Generation Tables.

- ☑ Table
  - ☑ cust
  - ☑ cust_credit
  - ☑ emp
  - ☑ mo_rent_rec
  - ☑ movie
  - ☑ movie_copy
  - ☑ payment
  - ☑ purchasinghistory
  - ☑ store
- ☑ View
  - ☑ customer_invoice
  - ☑ overdue_notice

11 selected, 11 shown

□ Apply selection of Tables to Tablespaces and Databases

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

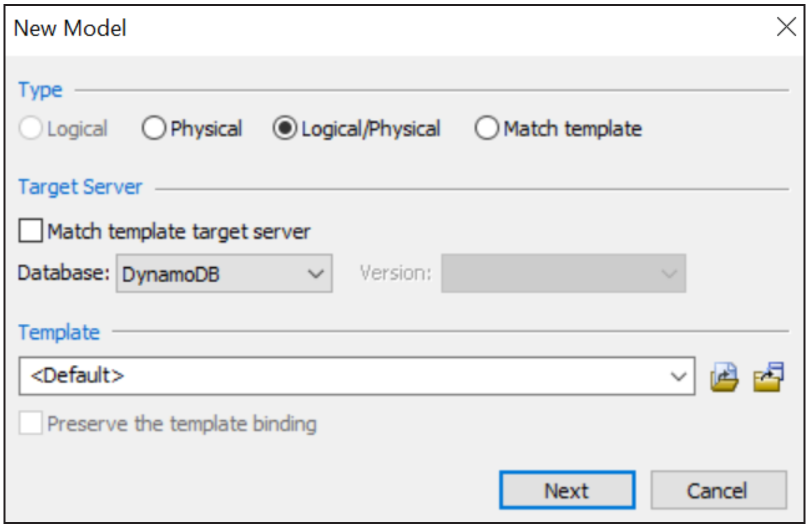
Reverse Engineering Models

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 information of each reverse engineering option, 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.



New Model

Type

○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

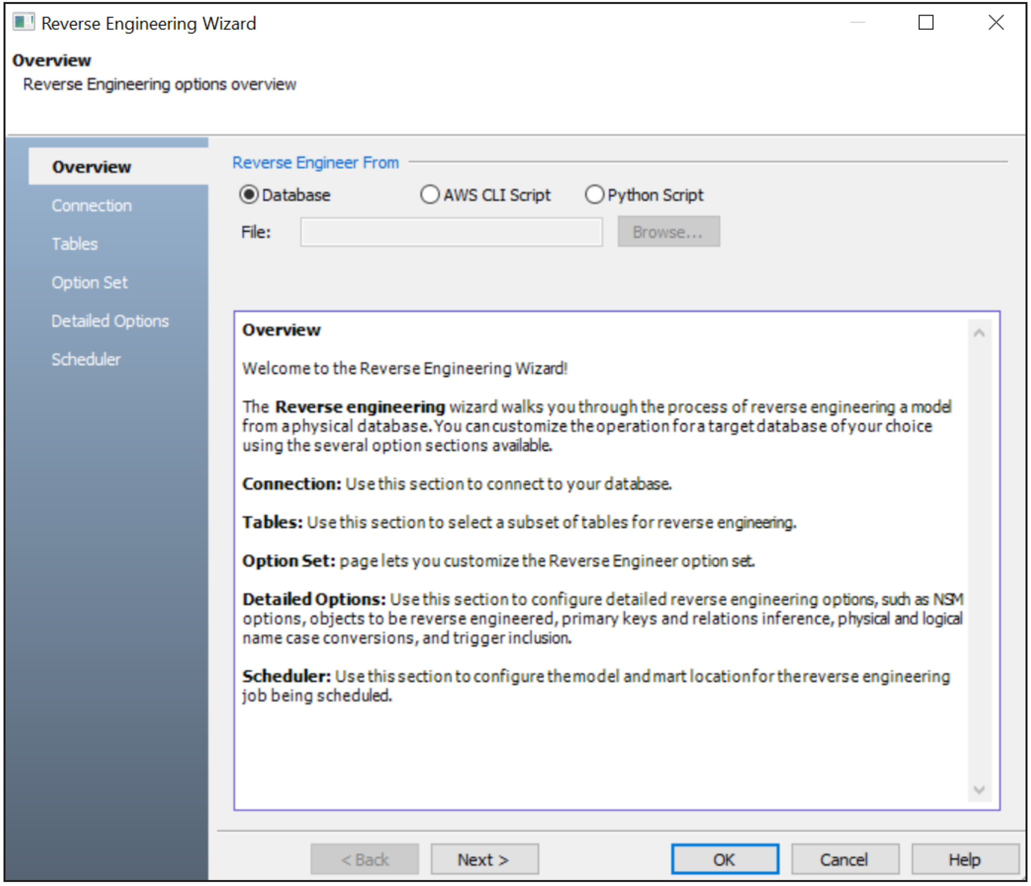
Database: DynamoDB Version:

Template

<Default>

□ Preserve the template binding

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



Overview
Reverse Engineering options overview

Reverse Engineer From

● Database ○ AWS CLI Script ○ Python Script

File:

Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Tables: Use this section to select a subset of tables for reverse engineering.

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

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

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

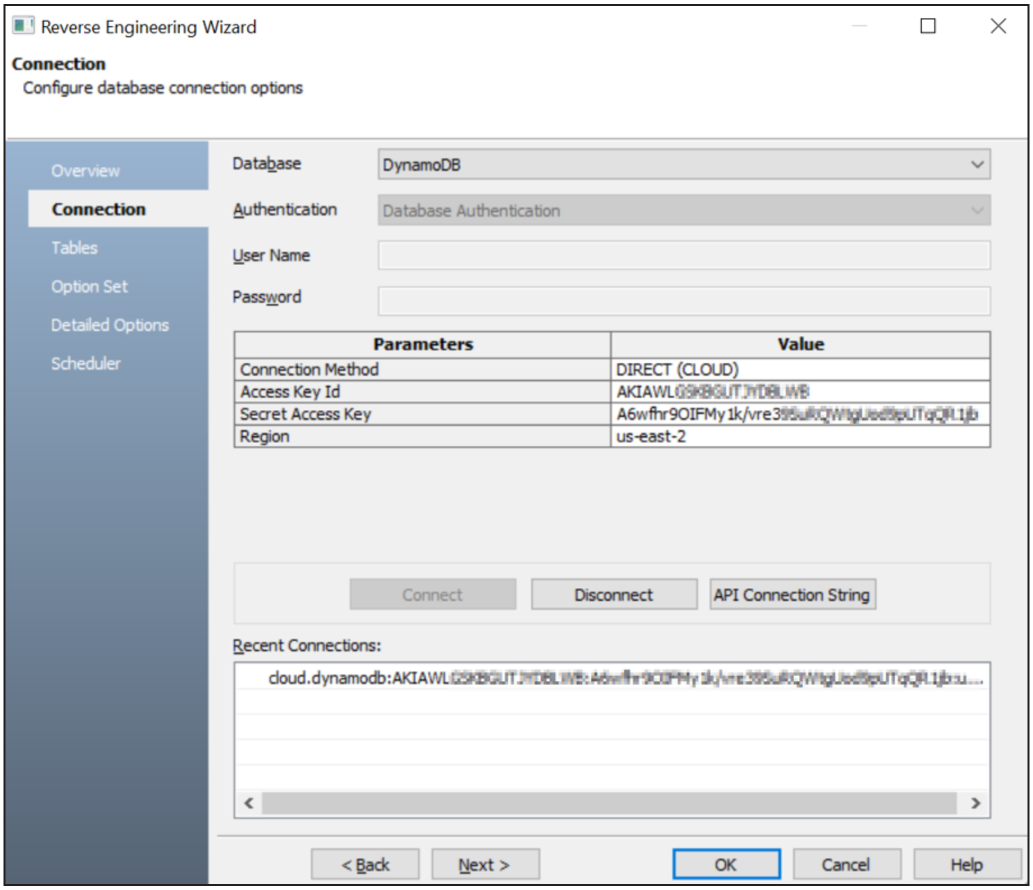
     > If you click **Database**, continue to step 5.

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

     > If you click **AWS CLI Script** or **Python Script**, go to step 11 to set detailed options for reverse engineering.

5. Click **Next**.

   The Connection section appears.



Connection

Configure database connection options

Database: DynamoDB

Authentication: Database Authentication

User Name:

Password:

Parameters	Value
Connection Method	DIRECT (CLOUD)
Access Key Id	AKIAWL[redacted]
Secret Access Key	A6wfhr9OIFMy1k/vre3[redacted]
Region	us-east-2

Recent Connections:

cloud.dynamodb:AKIAWL[redacted]

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

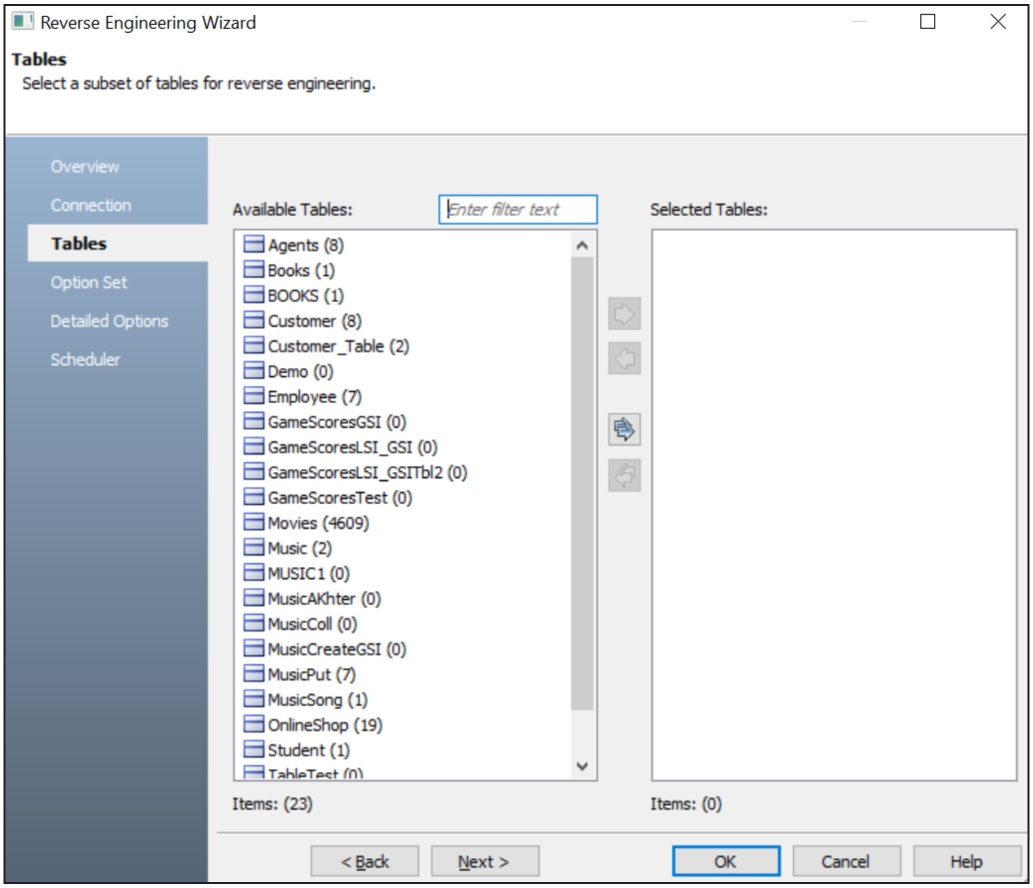
   The following table explains the connection parameters.

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 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 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 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	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 Connections.
8. Click **Next**.

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



Reverse Engineering Wizard

Tables

Select a subset of tables for reverse engineering.

Available Tables: Enter filter text

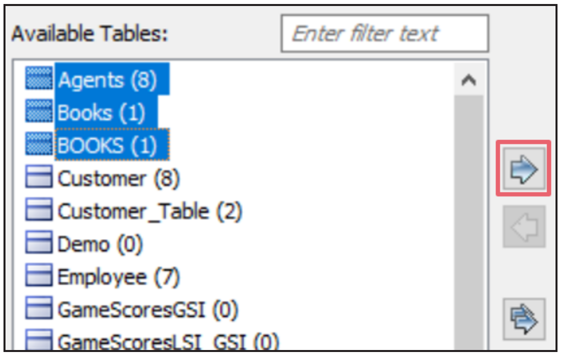
Agents (8)
Books (1)
BOOKS (1)
Customer (8)
Customer_Table (2)
Demo (0)
Employee (7)
GameScoresGSI (0)
GameScoresLSI_GSI (0)
GameScoresLSI_GSITbl2 (0)
GameScoresTest (0)
Movies (4609)
Music (2)
MUSIC1 (0)
MusicAKhter (0)
MusicColl (0)
MusicCreateGSI (0)
MusicPut (7)
MusicSong (1)
OnlineShop (19)
Student (1)
TableTest (0)

Items: (23)

Selected Tables:

Items: (0)

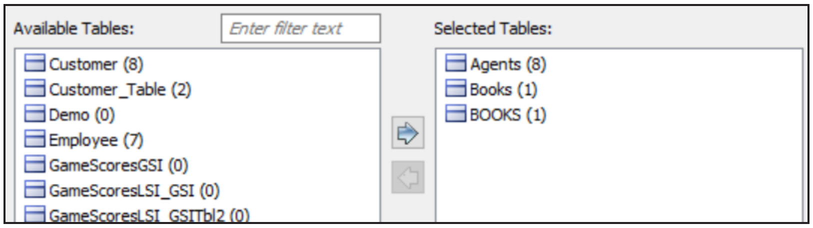
9. Under **Available Tables**, select the tables that you want to reverse engineer. Then, click ⇨.



Available Tables: Enter filter text

Agents (8)
Books (1)
BOOKS (1)
Customer (8)
Customer_Table (2)
Demo (0)
Employee (7)
GameScoresGSI (0)
GameScoresLSI_GSI (0)

This moves the selected tables under Selected Tables.



Available Tables:

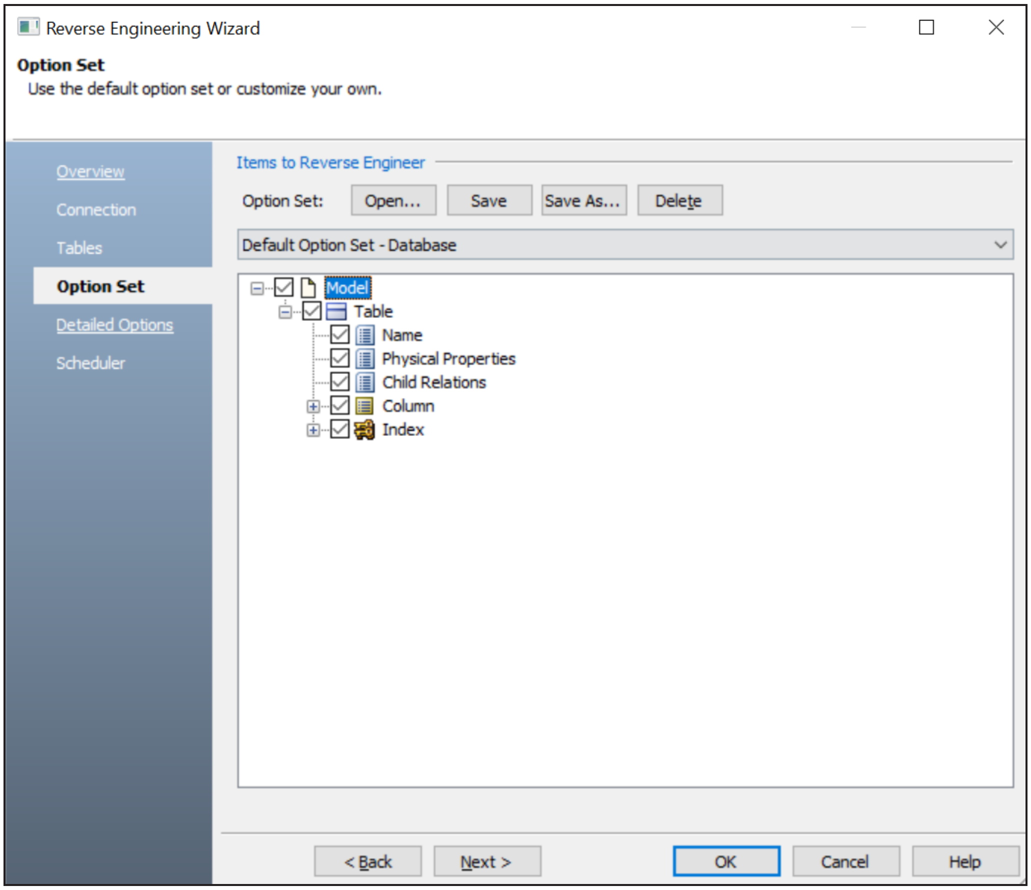
- Customer (8)
- Customer_Table (2)
- Demo (0)
- Employee (7)
- GameScoresGSI (0)
- GameScoresLSI_GSI (0)
- GameScoresLSI_GSITbl2 (0)

Selected Tables:

- Agents (8)
- Books (1)
- BOOKS (1)

10. Click **Next**.

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



Reverse Engineering Wizard

Option Set

Use the default option set or customize your own.

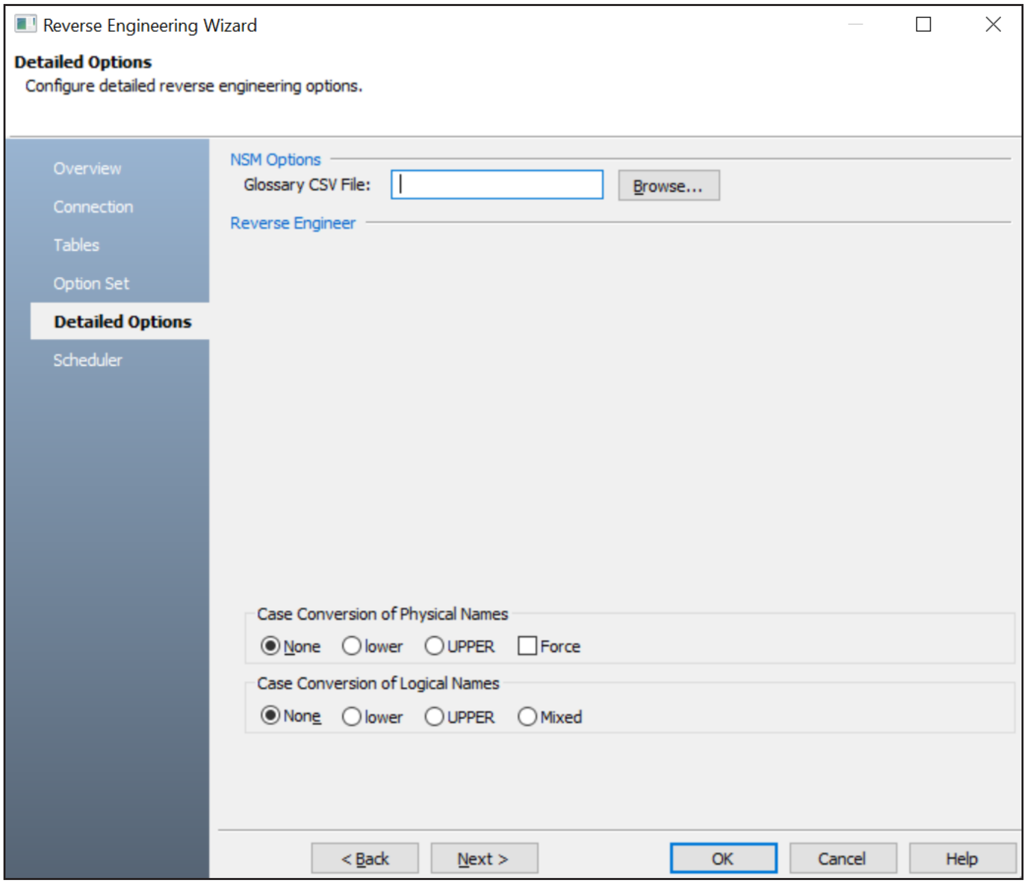
Items to Reverse Engineer

Option Set: Default Option Set - Database

- ☑ Model
  - ☑ Table
    - ☑ Name
    - ☑ Physical Properties
    - ☑ Child Relations
    - ☑ Column
    - ☑ Index

11. Click **Next**.

    The Detail Options section appears. Set up appropriate options based on your requirement.



Reverse Engineering Wizard

Detailed Options

Configure detailed reverse engineering options.

NSM Options

Glossary CSV File:

Reverse Engineer

Case Conversion of Physical Names

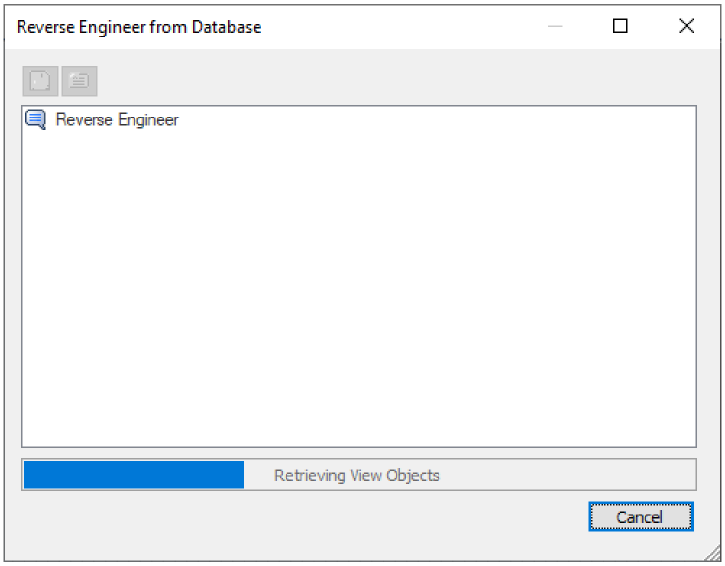
● None ○ lower ○ UPPER □ Force

Case Conversion of Logical Names

● None ○ lower ○ UPPER ○ Mixed

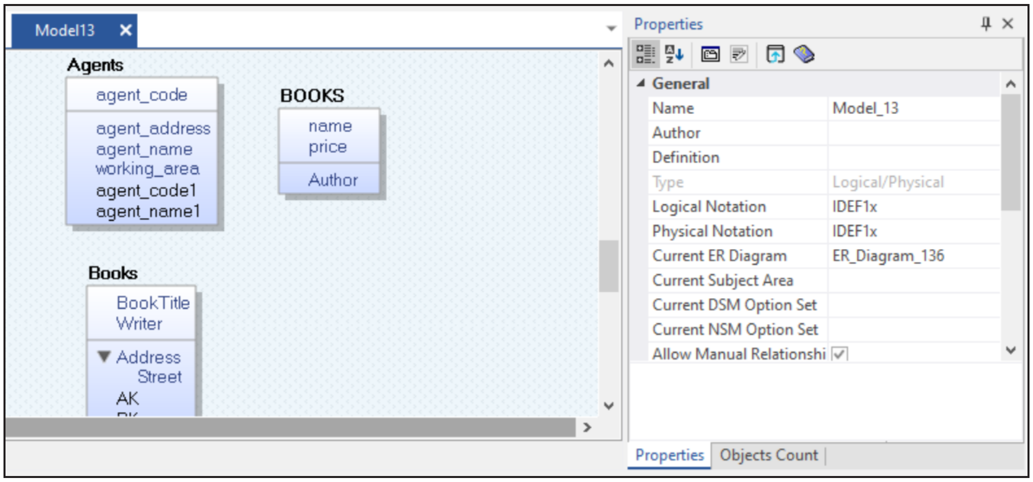
12. Click **OK**.

The reverse engineering process starts.



Retrieving View Objects

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



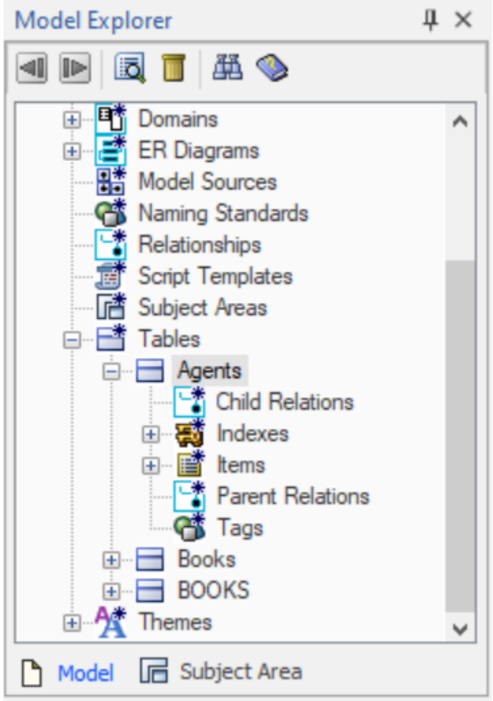
Properties

General

Name	Model_13
Author	
Definition	
Type	Logical/Physical
Logical Notation	IDEF1x
Physical Notation	IDEF1x
Current ER Diagram	ER_Diagram_136
Current Subject Area	
Current DSM Option Set	
Current NSM Option Set	
Allow Manual Relationshi	☑

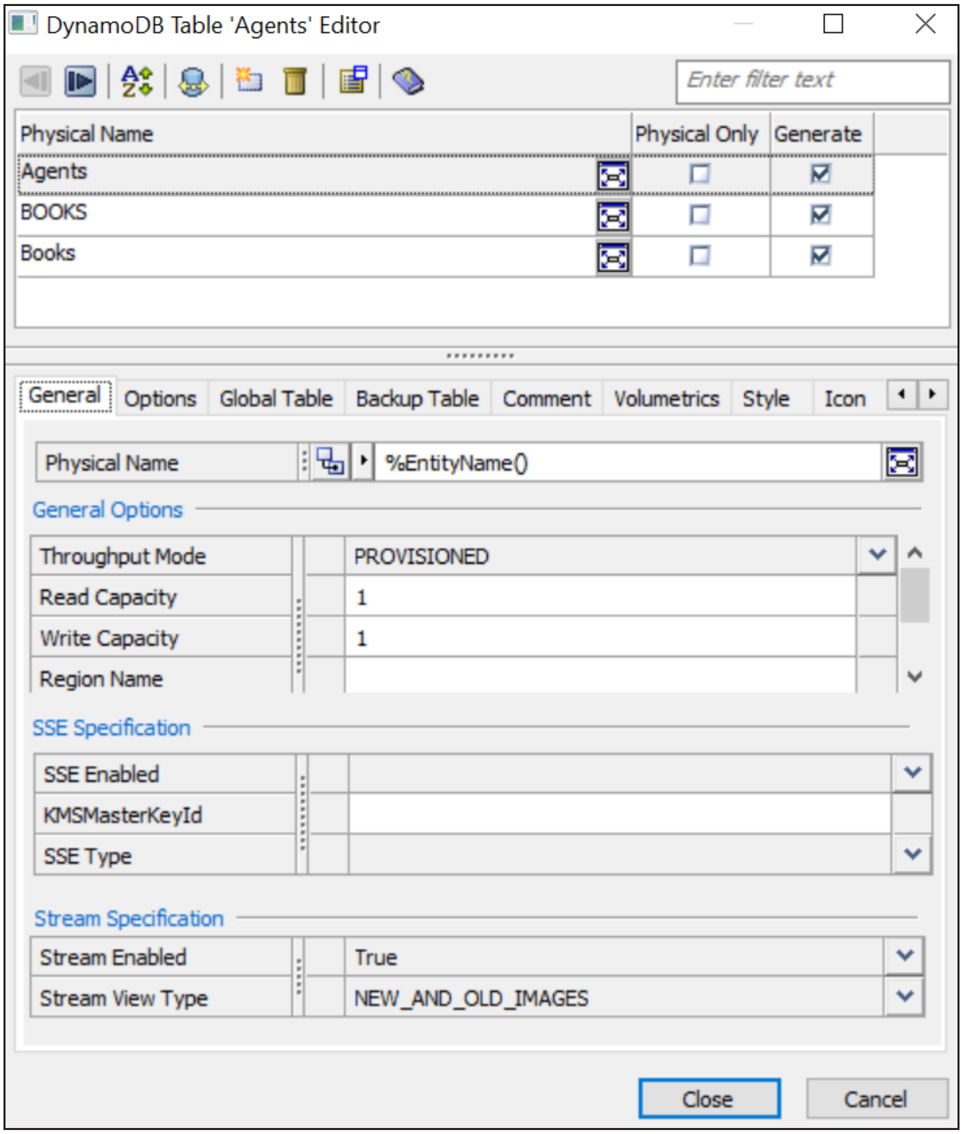
You can edit the shape of the nodes to look like the standard table-like structure. On the ribbon click **View** > **Field**. You can also change label color, size, and caption using the properties pane.

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



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's property on a model.



DynamoDB Table 'Agents' Editor

Physical Name	Physical Only	Generate
Agents	□	☑
BOOKS	□	☑
Books	□	☑

Physical Name: %EntityName()

General Options

Throughput Mode	PROVISIONED
Read Capacity	1
Write Capacity	1
Region Name	

SSE Specification

SSE Enabled	
KMSMasterKeyId	
SSE Type	

Stream Specification

Stream Enabled	True
Stream View Type	NEW_AND_OLD_IMAGES

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	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 where the reverse engineered model	When you schedule a job on a remote server, ensure the model path is same for remote and

	should be saved and its name	local server. For example: C:\Scheduler\<Model Name>.erwin
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.

Forward Engineering Models

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

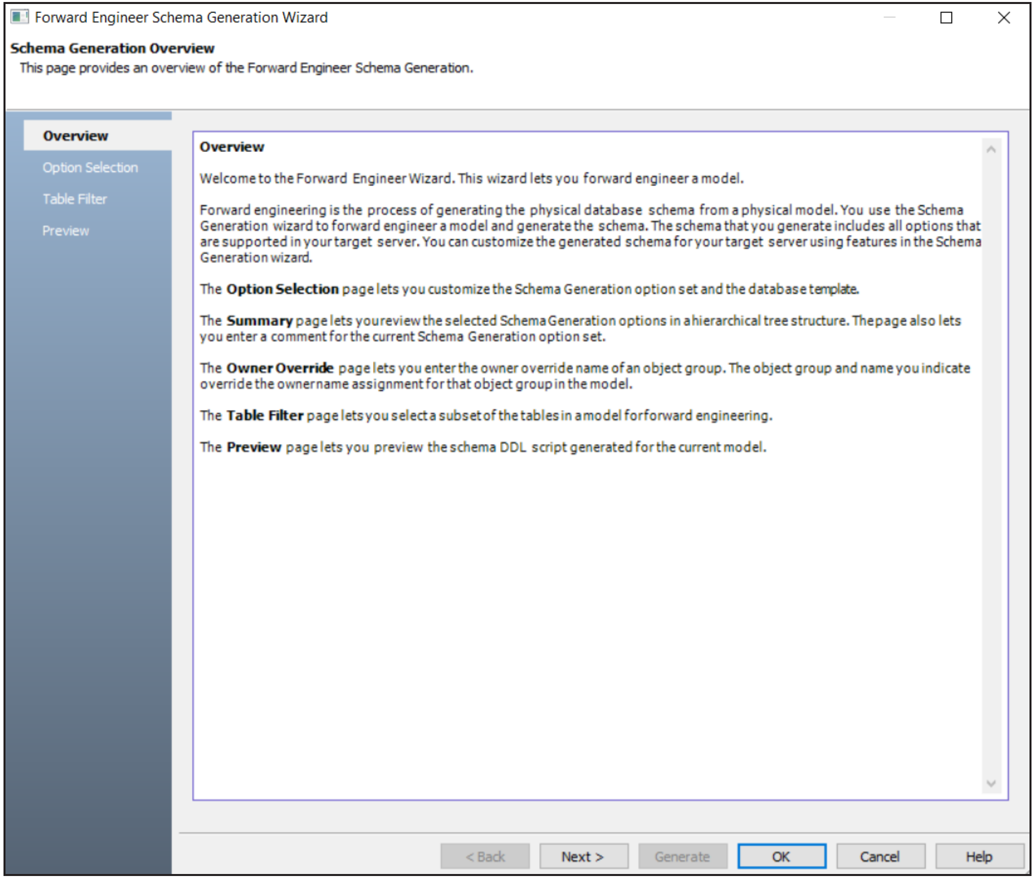
This topic walks you through the steps to forward engineer an DynamoDB model. For detailed information of each forward engineering option, refer to the Forward Engineering Options topic.

To forward engineer a DynamoDB model:

1. Open your DynamoDB 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.



Schema Generation Overview

This page provides an overview of the Forward Engineer Schema Generation.

Overview

Welcome to the Forward Engineer Wizard. This wizard lets you forward engineer a model.

Forward engineering is the process of generating the physical database schema from a physical model. You use the Schema Generation wizard to forward engineer a model and generate the schema. The schema that you generate includes all options that are supported in your target server. You can customize the generated schema for your target server using features in the Schema Generation wizard.

The **Option Selection** page lets you customize the Schema Generation option set and the database template.

The **Summary** page lets you review the selected Schema Generation options in a hierarchical tree structure. The page also lets you enter a comment for the current Schema Generation option set.

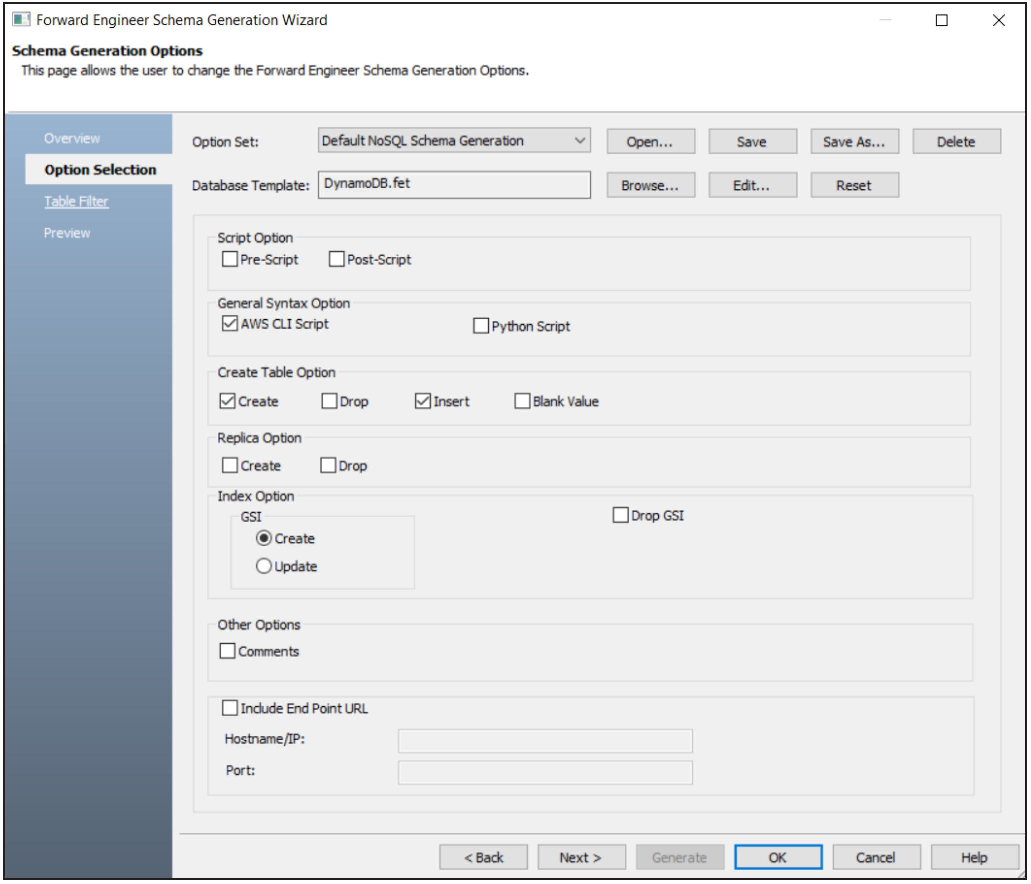
The **Owner Override** page lets you enter the owner override name of an object group. The object group and name you indicate override the owner name assignment for that object group in the model.

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

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

3. Click **Option Selection**.

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



Forward Engineer Schema Generation Wizard

Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default NoSQL Schema Generation

Database Template: DynamoDB.fet

Script Option
□ Pre-Script □ Post-Script

General Syntax Option
☑ AWS CLI Script □ Python Script

Create Table Option
☑ Create □ Drop ☑ Insert □ Blank Value

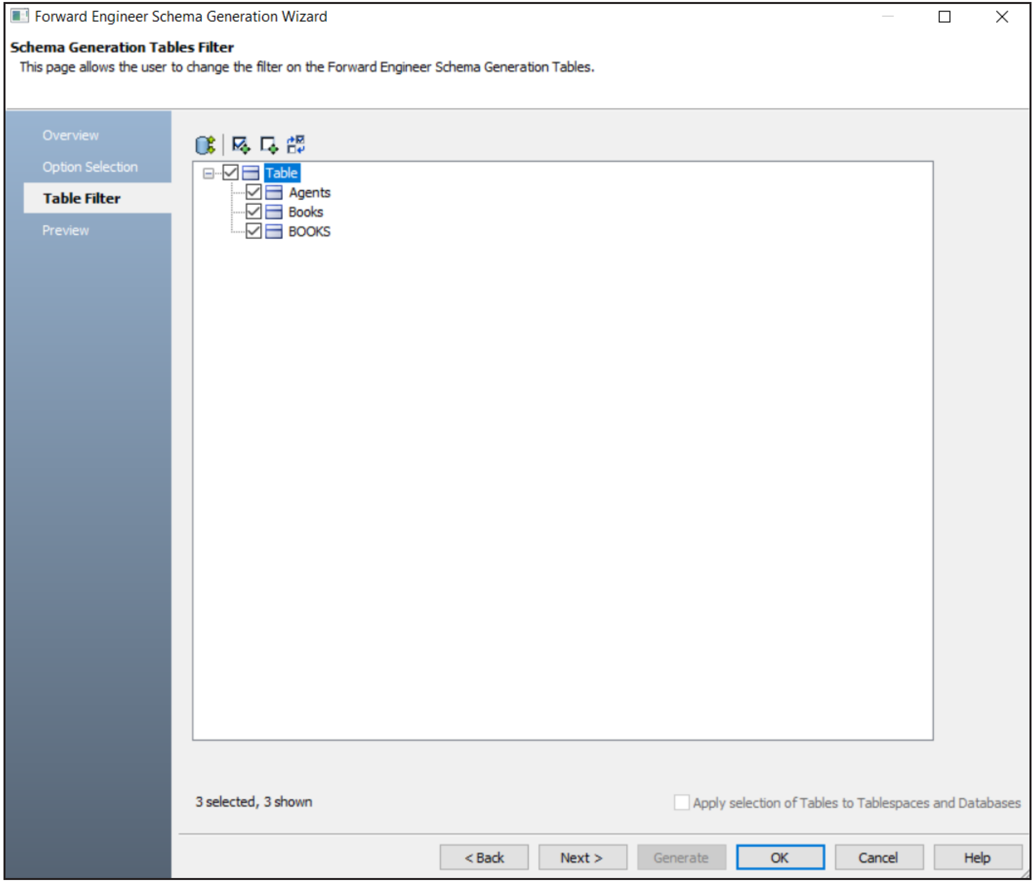
Replica Option
□ Create □ Drop

Index Option
GSI
● Create
○ Update
□ Drop GSI

Other Options
□ Comments

□ Include End Point URL
Hostname/IP:
Port:

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



Forward Engineer Schema Generation Wizard

Schema Generation Tables Filter

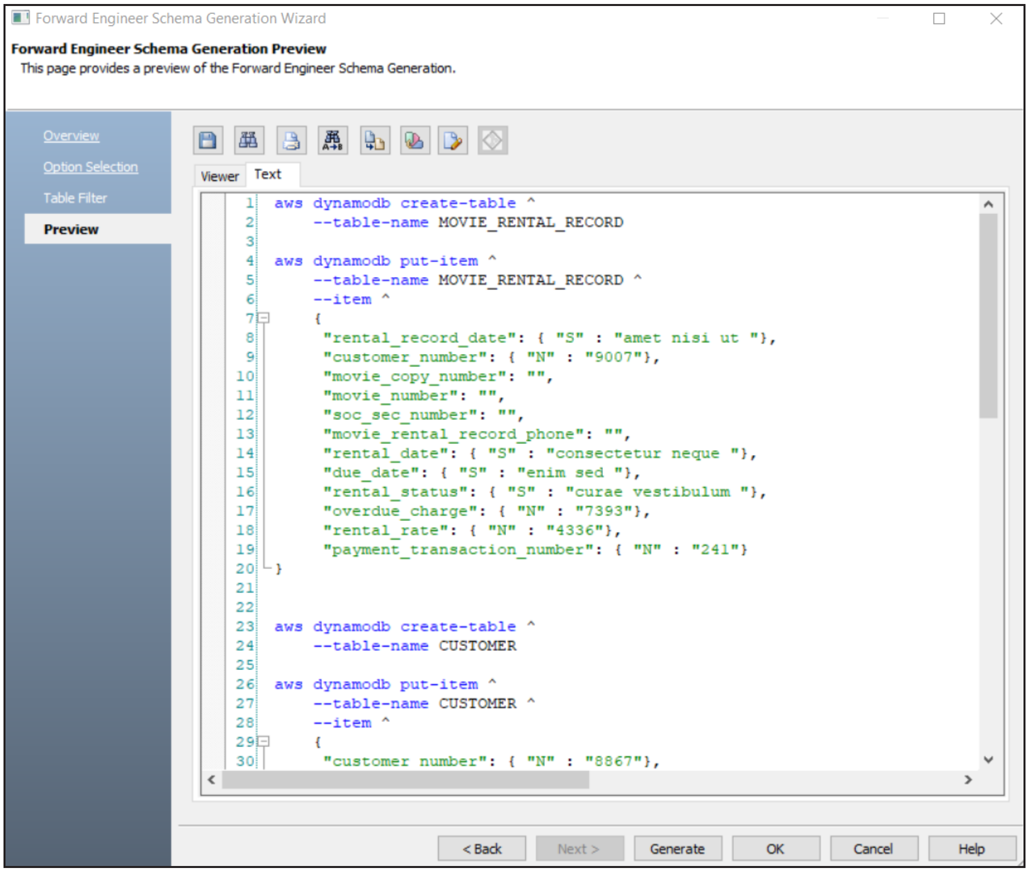
This page allows the user to change the filter on the Forward Engineer Schema Generation Tables.

3 selected, 3 shown

□ Apply selection of Tables to Tablespaces and Databases

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

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


Forward Engineer Schema Generation Wizard
Forward Engineer Schema Generation Preview
This page provides a preview of the Forward Engineer Schema Generation.
Overview
Option Selection
Table Filter
Preview
Viewer
Text
< Back
Next >
Generate
OK
Cancel
Help

```
aws dynamodb create-table ^
    --table-name MOVIE_RENTAL_RECORD

aws dynamodb put-item ^
    --table-name MOVIE_RENTAL_RECORD ^
    --item ^
    {
     "rental_record_date": { "S" : "amet nisi ut "},
     "customer_number": { "N" : "9007"},
     "movie_copy_number": "",
     "movie_number": "",
     "soc_sec_number": "",
     "movie_rental_record_phone": "",
     "rental_date": { "S" : "consectetur neque "},
     "due_date": { "S" : "enim sed "},
     "rental_status": { "S" : "curae vestibulum "},
     "overdue_charge": { "N" : "7393"},
     "rental_rate": { "N" : "4336"},
     "payment_transaction_number": { "N" : "241"}
}


aws dynamodb create-table ^
    --table-name CUSTOMER

aws dynamodb put-item ^
    --table-name CUSTOMER ^
    --item ^
    {
     "customer number": { "N" : "8867"},
```

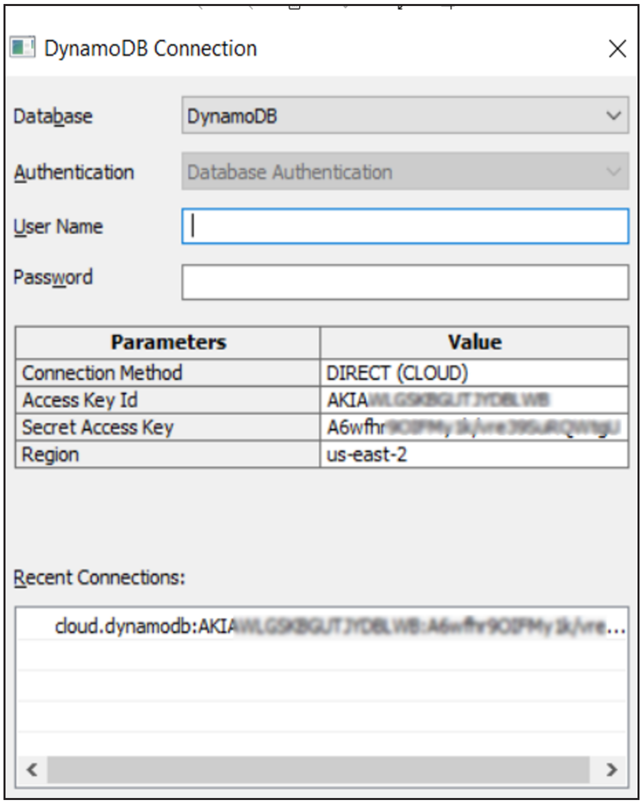
Use the following options:

- **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. 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 DynamoDB Connection editor appears.



DynamoDB Connection

Database: DynamoDB

Authentication: Database Authentication

User Name:

Password:

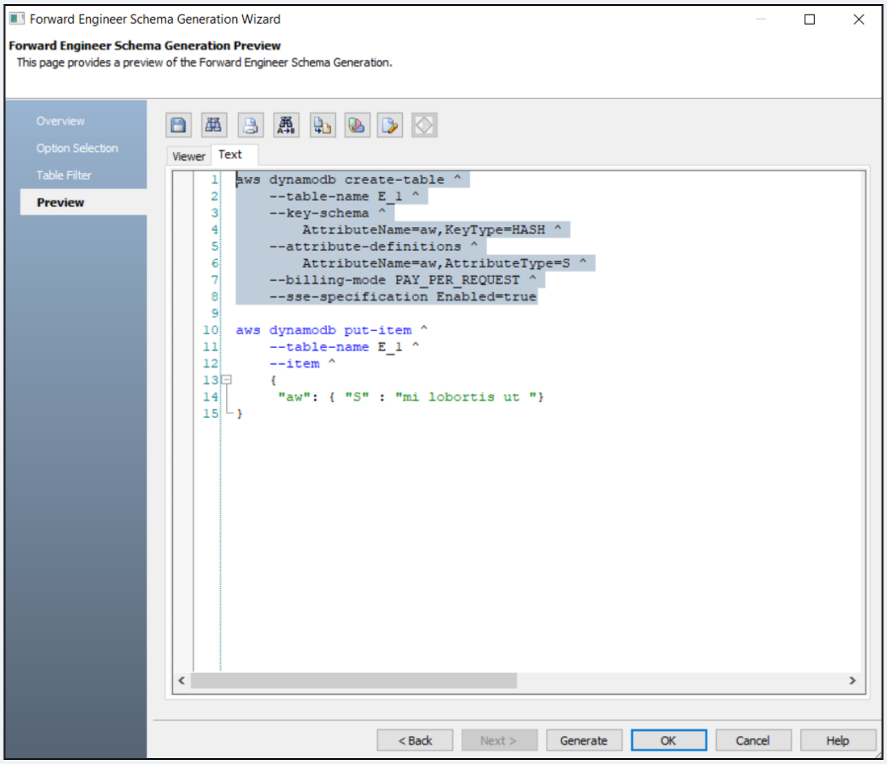
Parameters	Value
Connection Method	DIRECT (CLOUD)
Access Key Id	AKIA[redacted]
Secret Access Key	A6wfhr[redacted]
Region	us-east-2

Recent Connections:

cloud.dynamodb:AKIA[redacted]...

Generating forward engineering script for DynamoDB is not executed as expected and displays error, when you click Generate. Use one of the following method 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 is selected from two statements in the script.



Forward Engineer Schema Generation Preview

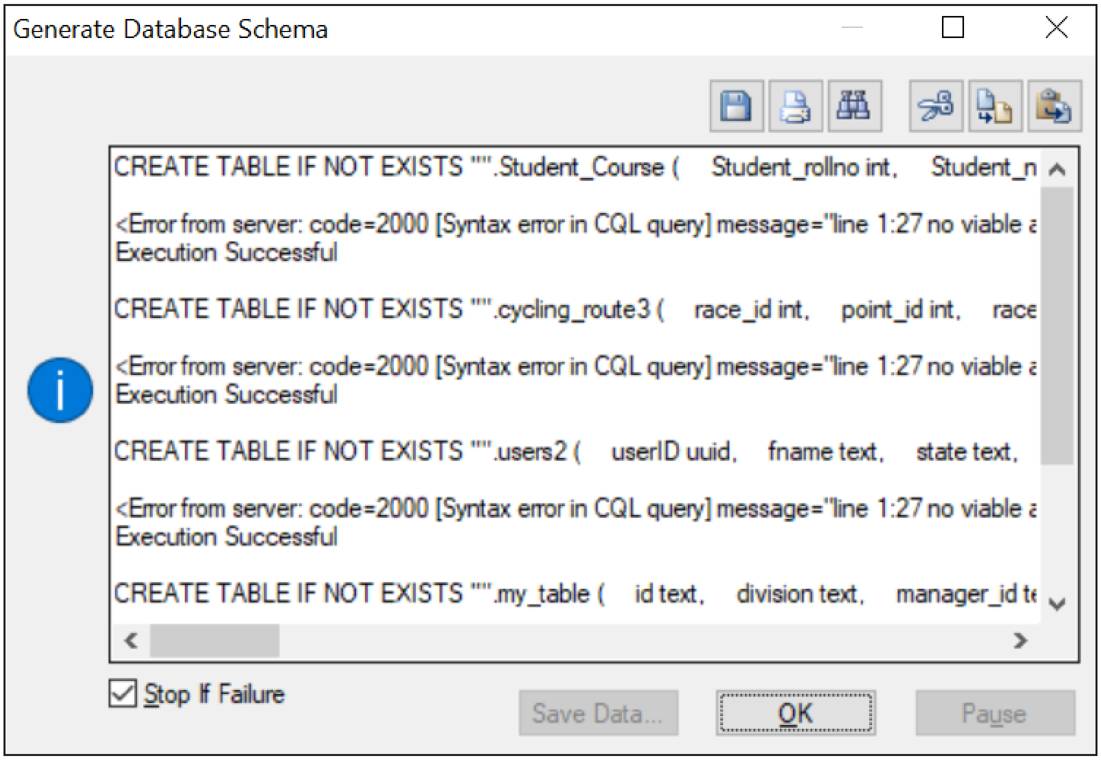
This page provides a preview of the Forward Engineer Schema Generation.

```
aws dynamodb create-table ^
    --table-name E_1 ^
    --key-schema ^
        AttributeName=aw,KeyType=HASH ^
    --attribute-definitions ^
        AttributeName=aw,AttributeType=S ^
    --billing-mode PAY_PER_REQUEST ^
    --sse-specification Enabled=true

aws dynamodb put-item ^
    --table-name E_1 ^
    --item ^
    {
     "aw": { "S" : "mi lobortis ut "}
}
```

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.



```
CREATE TABLE IF NOT EXISTS "".Student_Course (    Student_rollno int,    Student_n

<Error from server: code=2000 [Syntax error in CQL query] message="line 1:27 no viable a
Execution Successful

CREATE TABLE IF NOT EXISTS "".cycling_route3 (    race_id int,    point_id int,    race

<Error from server: code=2000 [Syntax error in CQL query] message="line 1:27 no viable a
Execution Successful

CREATE TABLE IF NOT EXISTS "".users2 (    userID uuid,    fname text,    state text,

<Error from server: code=2000 [Syntax error in CQL query] message="line 1:27 no viable a
Execution Successful

CREATE TABLE IF NOT EXISTS "".my_table (    id text,    division text,    manager_id te
```

☑ Stop If Failure

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 prescripts 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.

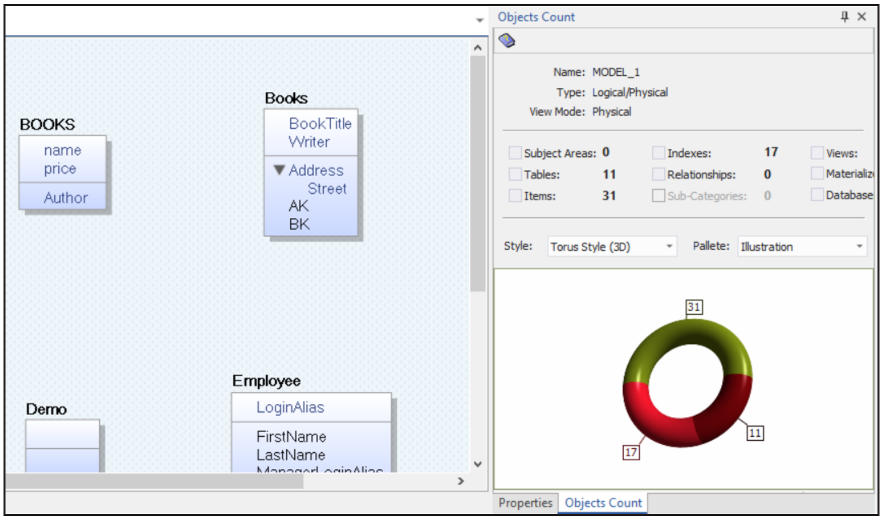
This topic walks you through the steps to compare a DynamoDB model with database.

To compare models with database:

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

   > Ensure that you are in Physical mode.

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

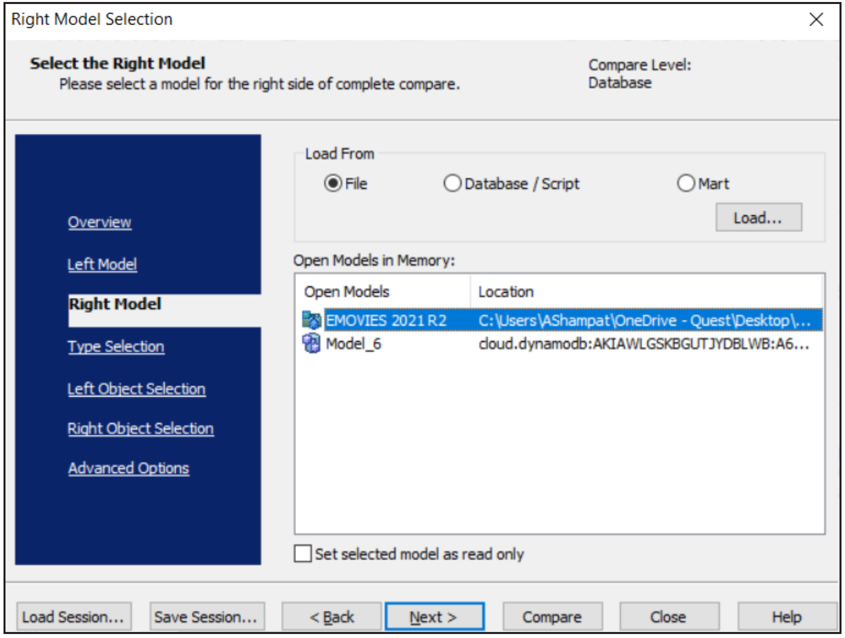


Name: MODEL_1
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 0 □ Indexes: 17 □ Views:
□ Tables: 11 □ Relationships: 0 □ Materializ
□ Items: 31 □ Sub-Categories: 0 □ Database

Style: Torus Style (3D) Pallete: Illustration

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.



Right Model Selection

Select the Right Model
Please select a model for the right side of complete compare.

Compare Level:
Database

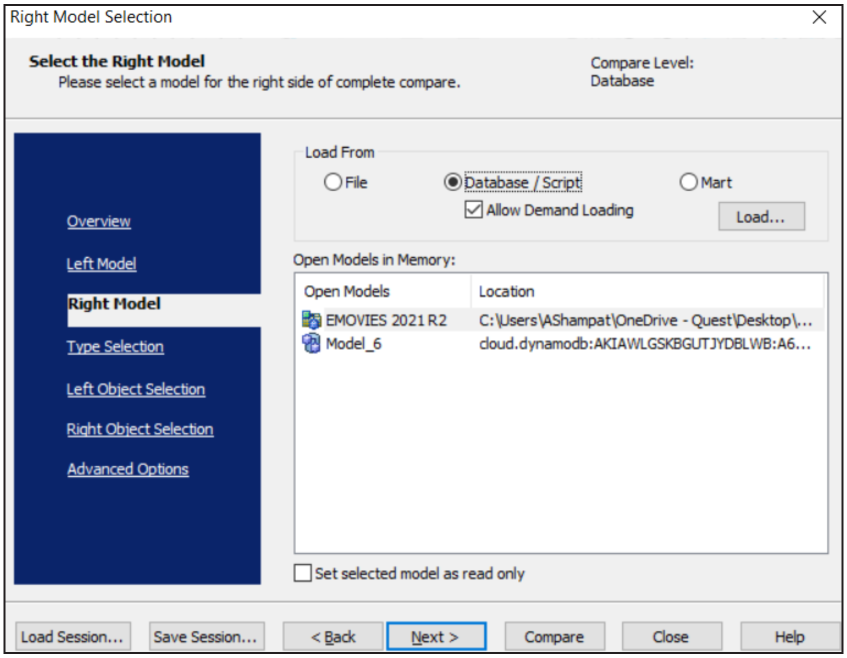
Load From: ● File ○ Database / Script ○ Mart

Open Models in Memory:

Open Models	Location
EMOVIES 2021 R2	C:\Users\AShampat\OneDrive - Quest\Desktop\...
Model_6	cloud.dynamodb:AKIAWLGSKBGUTJYDBLWB:A6...

□ Set selected model as read only

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



Right Model Selection

Select the Right Model
Please select a model for the right side of complete compare.

Compare Level:
Database

Load From: ○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

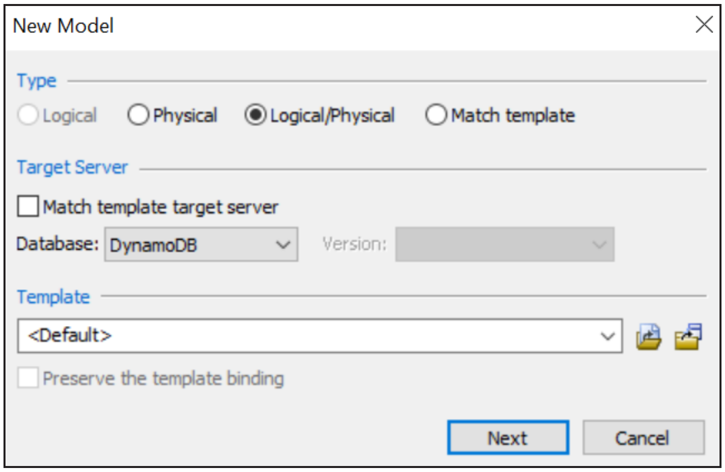
Open Models in Memory:

Open Models	Location
EMOVIES 2021 R2	C:\Users\AShampat\OneDrive - Quest\Desktop\...
Model_6	cloud.dynamodb:AKIAWLGSKBGUTJYDBLWB:A6...

□ Set selected model as read only

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

a model from the database to compare.



New Model

Type

○ Logical ○ Physical ● Logical/Physical ○ Match template

Target Server

□ Match template target server

Database: DynamoDB

Version:

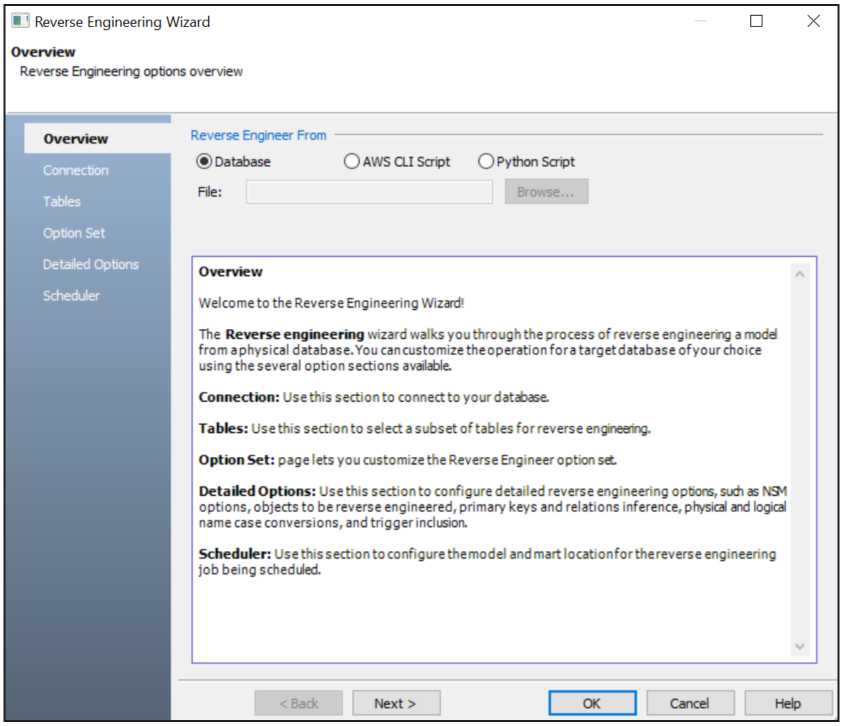
Template

<Default>

□ Preserve the template binding

5. Ensure that the Database is set to the correct one. In this case, DynamoDB. Then, click **Next**.

   The Reverse Engineer Wizard appears.



Reverse Engineering Wizard

Overview

Reverse Engineering options overview

Reverse Engineer From

● Database ○ AWS CLI Script ○ Python Script

File:

Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Tables: Use this section to select a subset of tables for reverse engineering.

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

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

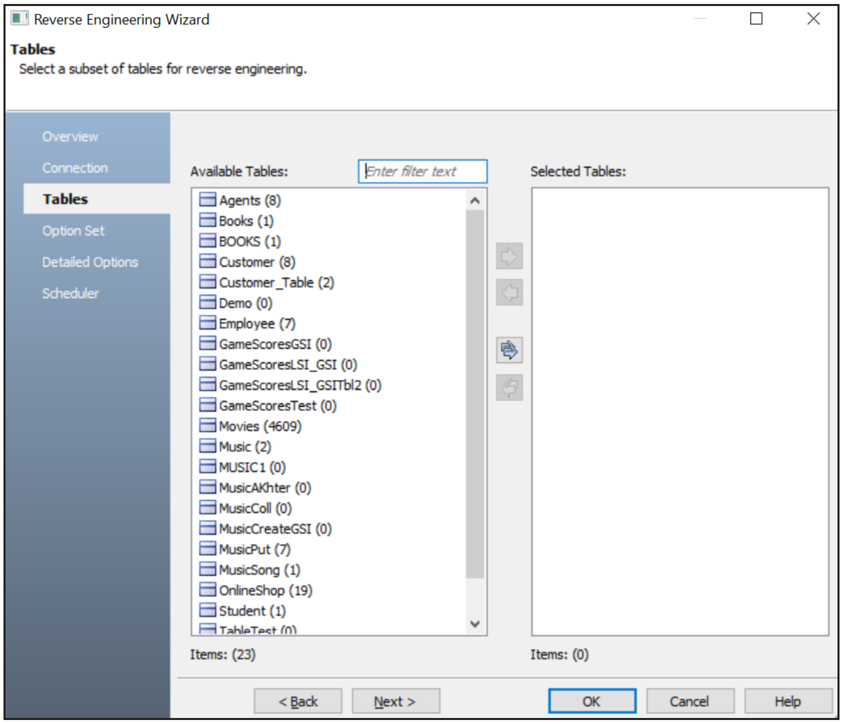
Scheduler: Use this section to configure the model and mart location for the reverse engineering job being scheduled.

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.



Tables

Select a subset of tables for reverse engineering.

Available Tables:

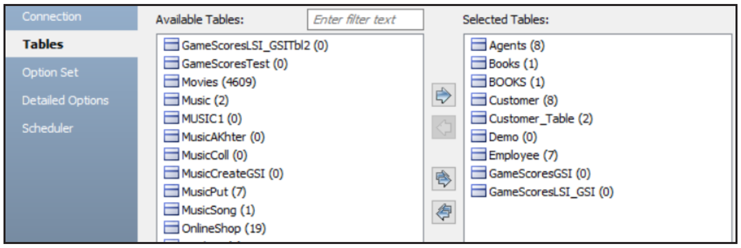
Agents (8)
Books (1)
BOOKS (1)
Customer (8)
Customer_Table (2)
Demo (0)
Employee (7)
GameScoresGSI (0)
GameScoresLSI_GSI (0)
GameScoresLSI_GSITbl2 (0)
GameScoresTest (0)
Movies (4609)
Music (2)
MUSIC1 (0)
MusicAKhter (0)
MusicColl (0)
MusicCreateGSI (0)
MusicPut (7)
MusicSong (1)
OnlineShop (19)
Student (1)
TableTest (0)

Items: (23)

Items: (0)

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

   This moves the tables under Selected Tables.



Available Tables:

GameScoresLSI_GSITbl2 (0)
GameScoresTest (0)
Movies (4609)
Music (2)
MUSIC1 (0)
MusicAKhter (0)
MusicColl (0)
MusicCreateGSI (0)
MusicPut (7)
MusicSong (1)
OnlineShop (19)

Selected Tables:

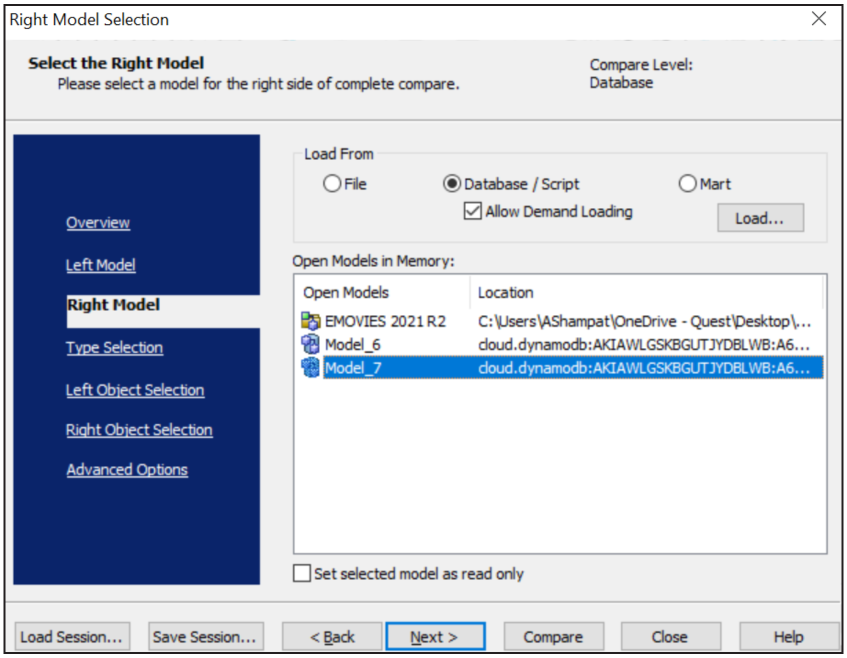
Agents (8)
Books (1)
BOOKS (1)
Customer (8)
Customer_Table (2)
Demo (0)
Employee (7)
GameScoresGSI (0)
GameScoresLSI_GSI (0)

9. Click **Next** and in the Option Set section, keep the default configuration.

10. Click **Next** and in the Detail 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.



Right Model Selection

Select the Right Model

Please select a model for the right side of complete compare.

Compare Level: Database

Load From

○ File ● Database / Script ○ Mart

☑ Allow Demand Loading

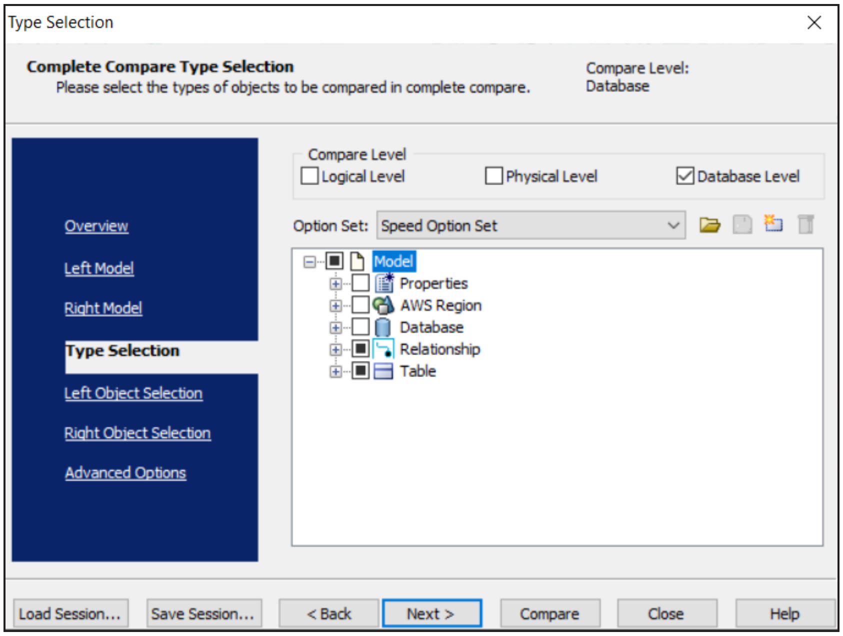
Open Models in Memory:

Open Models	Location
EMOVIES 2021 R2	C:\Users\AShampat\OneDrive - Quest\Desktop\...
Model_6	cloud.dynamodb:AKIAWLGSKBGUTJYDBLWB:A6...
Model_7	cloud.dynamodb:AKIAWLGSKBGUTJYDBLWB:A6...

☐ Set selected model as read only

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

    For example, the following image shows the default options.



Type Selection

Complete Compare Type Selection

Please select the types of objects to be compared in complete compare.

Compare Level: Database

Compare Level

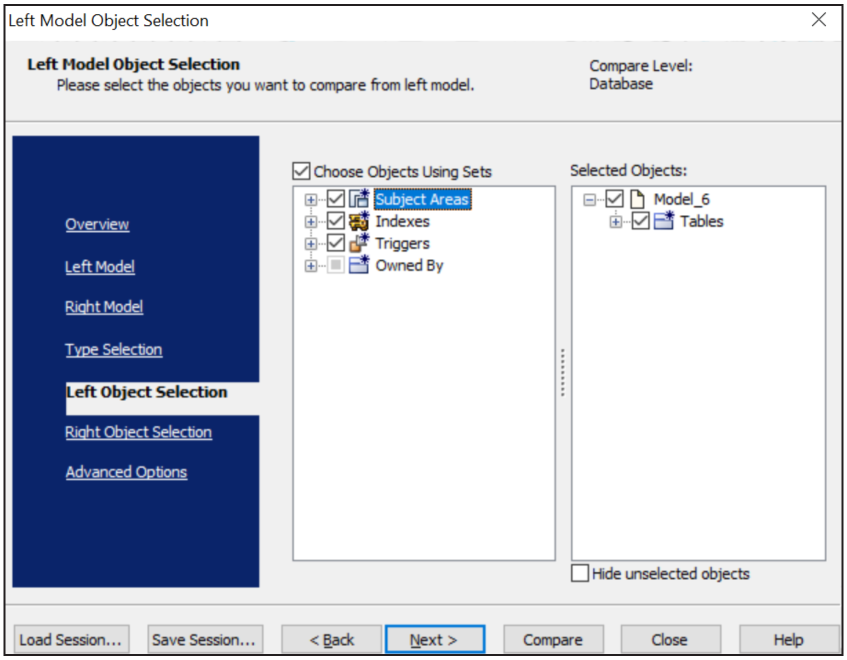
☐ Logical Level ☐ Physical Level ☑ Database Level

Option Set: Speed Option Set

- ■ Model
  - ☐ Properties
  - ☐ AWS Region
  - ☐ Database
  - ■ Relationship
  - ■ Table

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

    For example, the following image shows the default options.



Left Model Object Selection

Left Model Object Selection
Please select the objects you want to compare from left model.

Compare Level:
Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

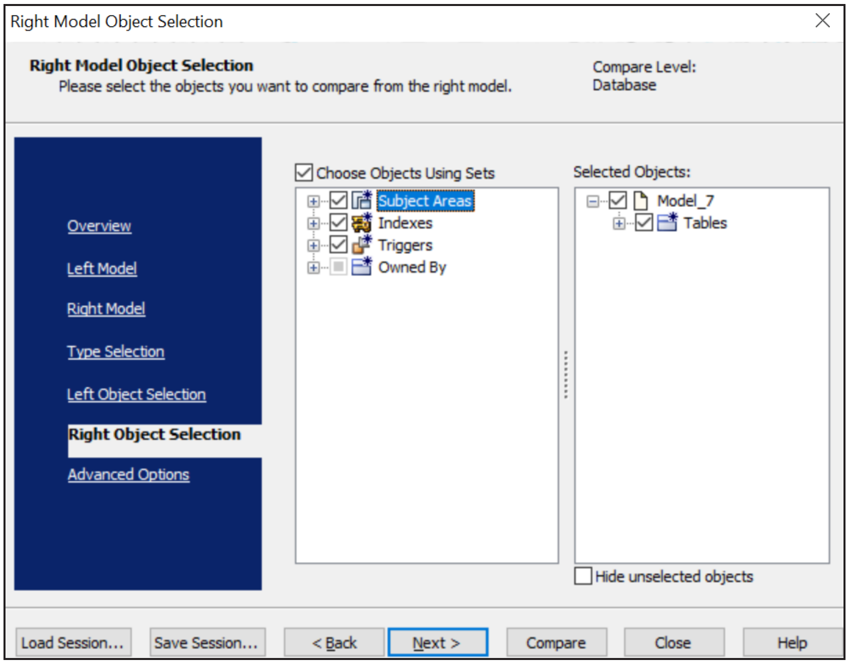
Selected Objects:

- ☑ Model_6
  - ☑ Tables

□ Hide unselected objects

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

For example, the following image shows the default options.



Right Model Object Selection

Right Model Object Selection
Please select the objects you want to compare from the right model.

Compare Level:
Database

☑ Choose Objects Using Sets

- ☑ Subject Areas
- ☑ Indexes
- ☑ Triggers
- □ Owned By

Selected Objects:

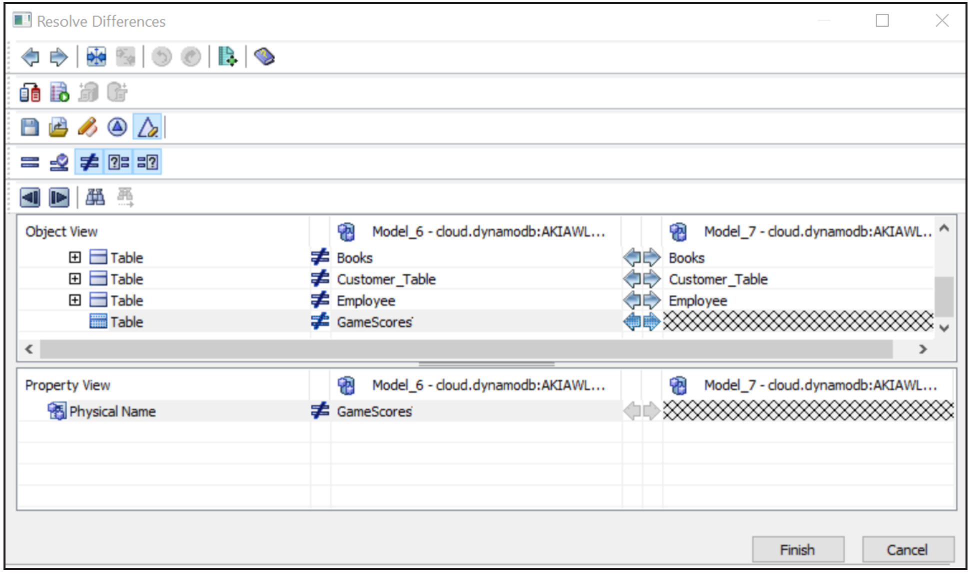
- ☑ Model_7
  - ☑ Tables

□ Hide unselected objects

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.



Resolve Differences

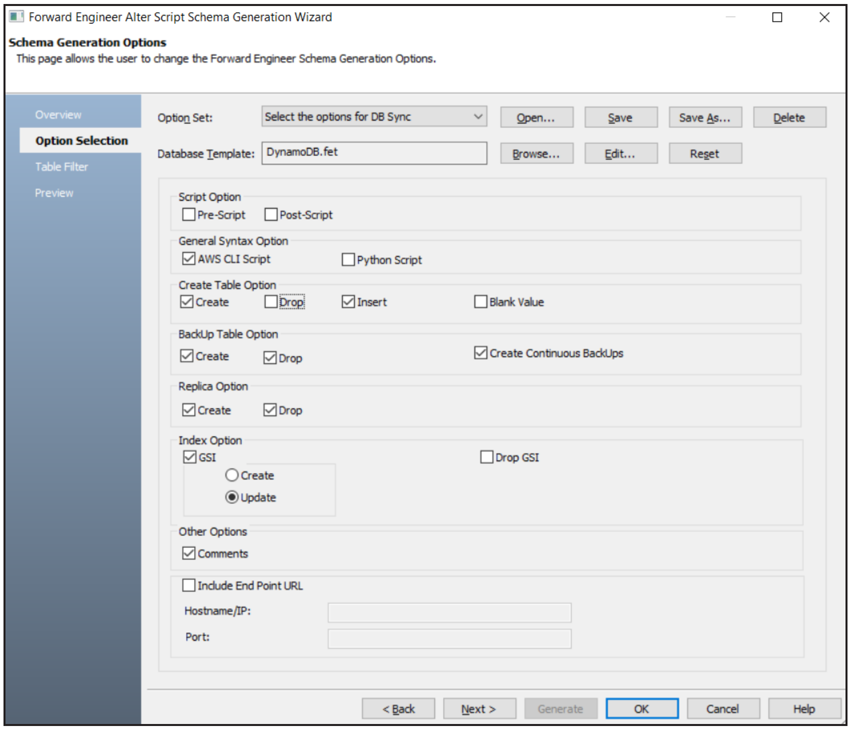
Object View	Model_6 - cloud.dynamodb:AKIAWL...		Model_7 - cloud.dynamodb:AKIAWL...
⊞ Table	≠ Books	⇦⇨	Books
⊞ Table	≠ Customer_Table	⇦⇨	Customer_Table
⊞ Table	≠ Employee	⇦⇨	Employee
Table	≠ GameScores	⇦⇨	▨

Property View	Model_6 - cloud.dynamodb:AKIAWL...		Model_7 - cloud.dynamodb:AKIAWL...
Physical Name	≠ GameScores	⇦⇨	▨

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.



Forward Engineer Alter Script Schema Generation Wizard

Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Select the options for DB Sync

Database Template: DynamoDB.fet

Script Option
☐ Pre-Script ☐ Post-Script

General Syntax Option
☑ AWS CLI Script ☐ Python Script

Create Table Option
☑ Create ☐ Drop ☑ Insert ☐ Blank Value

BackUp Table Option
☑ Create ☑ Drop ☑ Create Continuous BackUps

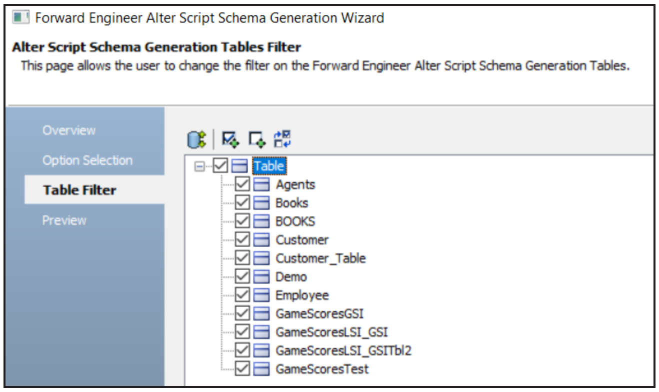
Replica Option
☑ Create ☑ Drop

Index Option
☑ GSI ☐ Drop GSI
○ Create
● Update

Other Options
☑ Comments

☐ Include End Point URL
Hostname/IP:
Port:

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



Forward Engineer Alter Script Schema Generation Wizard

Alter Script Schema Generation Tables Filter

This page allows the user to change the filter on the Forward Engineer Alter Script Schema Generation Tables.

☑ Table
- ☑ Agents
- ☑ Books
- ☑ BOOKS
- ☑ Customer
- ☑ Customer_Table
- ☑ Demo
- ☑ Employee
- ☑ GameScoresGSI
- ☑ GameScoresLSI_GSI
- ☑ GameScoresLSI_GSITbl2
- ☑ GameScoresTest

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 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.

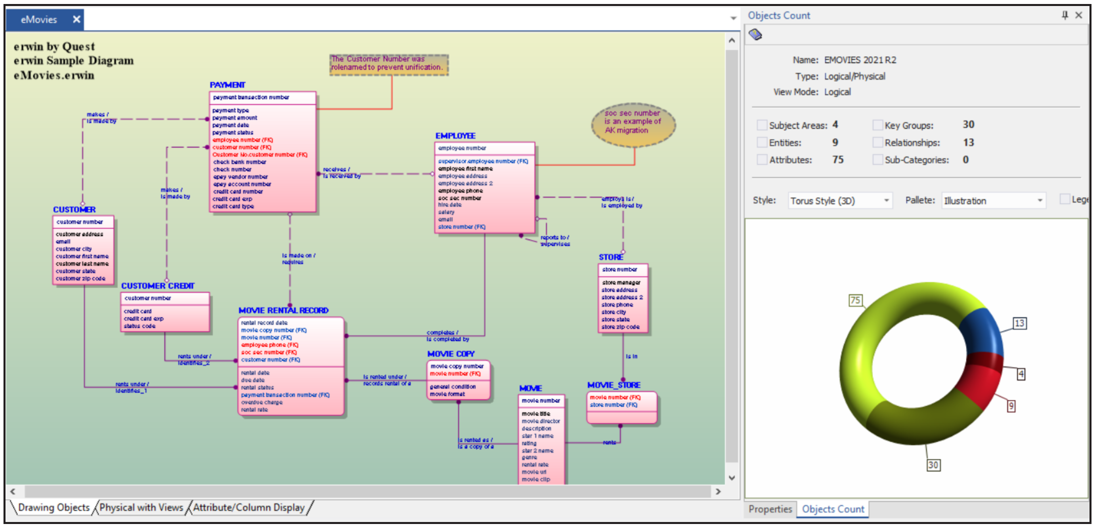
Migration by Changing the Target Database

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

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

> 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.

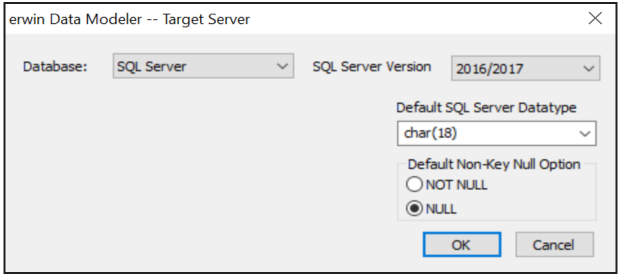


Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Logical

Subject Areas: 4
Entities: 9
Attributes: 75
Key Groups: 30
Relationships: 13
Sub-Categories: 0

Style: Torus Style (3D)
Pallete: Illustration

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.


erwin Data Modeler -- Target Server

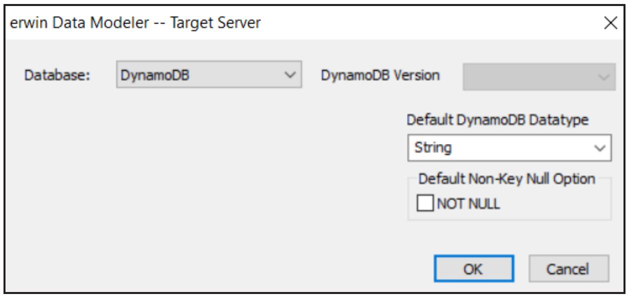
Database: SQL Server

SQL Server Version: 2016/2017

Default SQL Server Datatype: char(18)

Default Non-Key Null Option
○ NOT NULL
● NULL

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


erwin Data Modeler -- Target Server

Database: DynamoDB

DynamoDB Version

Default DynamoDB Datatype: String

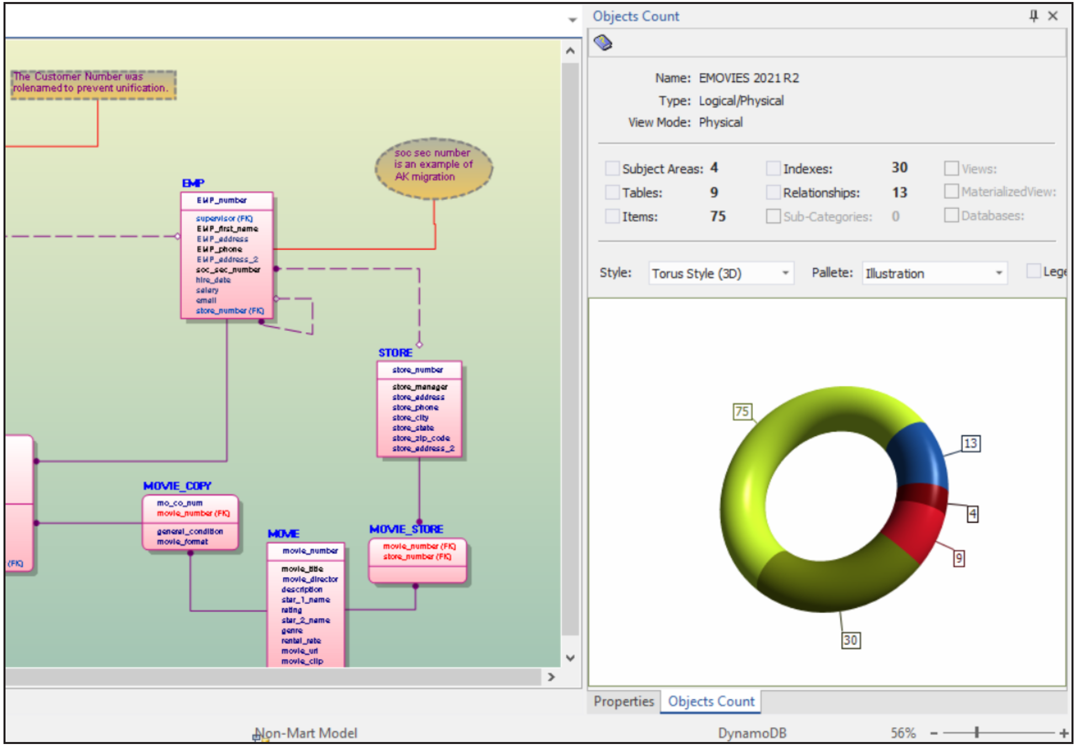
Default Non-Key Null Option
□ NOT NULL

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 DynamoDB model.



Objects Count

Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 4
□ Tables: 9
□ Items: 75
□ Indexes: 30
□ Relationships: 13
□ Sub-Categories: 0
□ Views:
□ MaterializedView:
□ Databases:

Style: Torus Style (3D)
Pallete: Illustration
□ Lege

In the **Objects Count** pane, you can view the details of converted database. The migration process converts and merges multiple tables, columns, and relationships to the DynamoDB format.

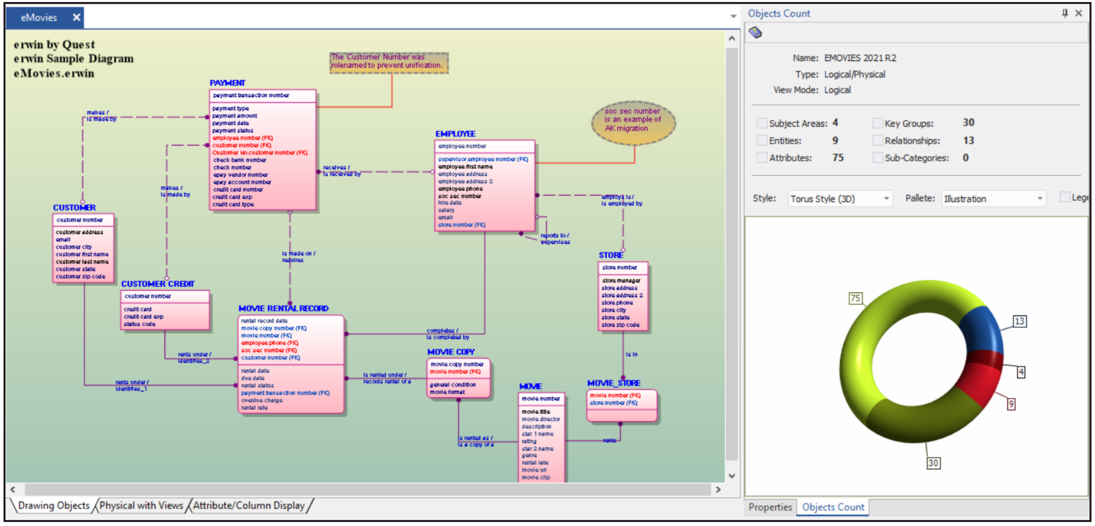
Migration by Deriving a Model

To migrate by deriving a model, follow these steps:

1. Open your relational model in erwin DM.

   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.

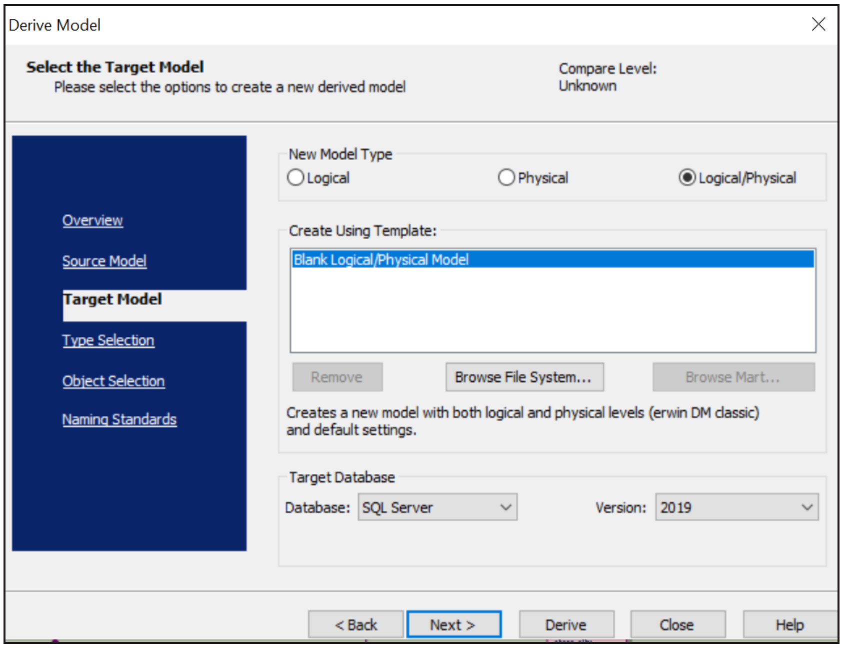


Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Logical

Subject Areas: 4
Entities: 9
Attributes: 75
Key Groups: 30
Relationships: 13
Sub-Categories: 0

Style: Torus Style (3D)
Pallete: Illustration

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: 2019

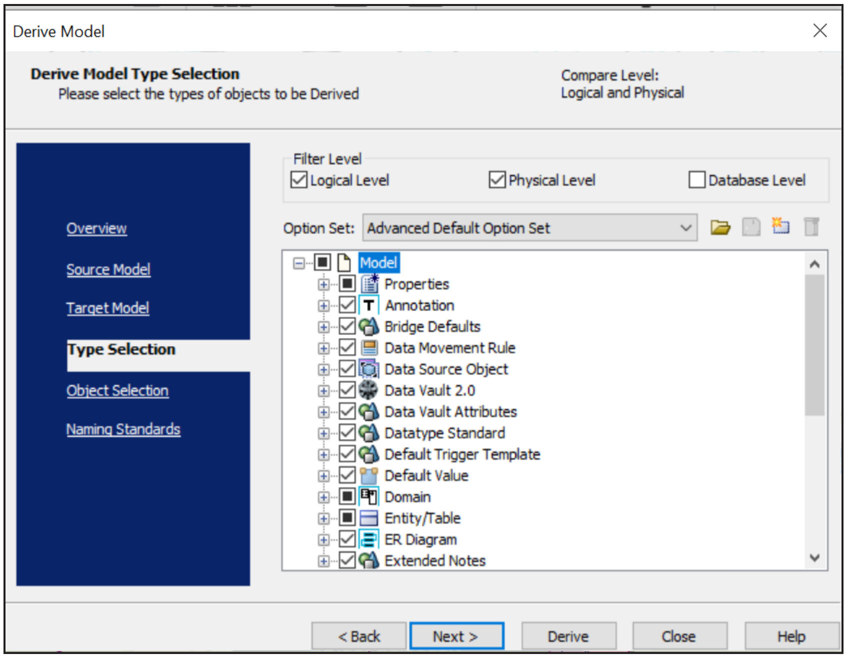
< Back | Next > | Derive | Close | Help

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.



Derive Model

Derive Model Type Selection
Please select the types of objects to be Derived

Compare Level:
Logical and Physical

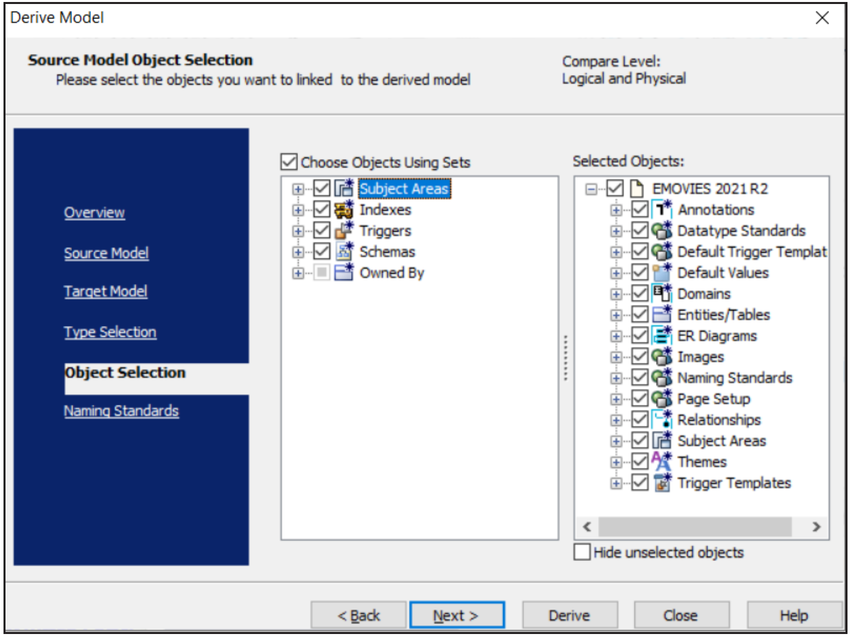
Filter Level
☑ Logical Level ☑ Physical Level □ Database Level

Option Set: Advanced Default Option Set

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.



Derive Model

Source Model Object Selection
Please select the objects you want to linked to the derived model

Compare Level:
Logical and Physical

☑ Choose Objects Using Sets

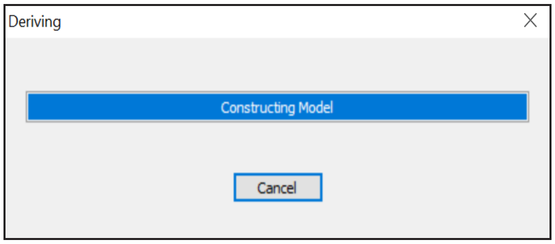
Selected Objects:

□ Hide unselected objects

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

8. Click **Derive**.

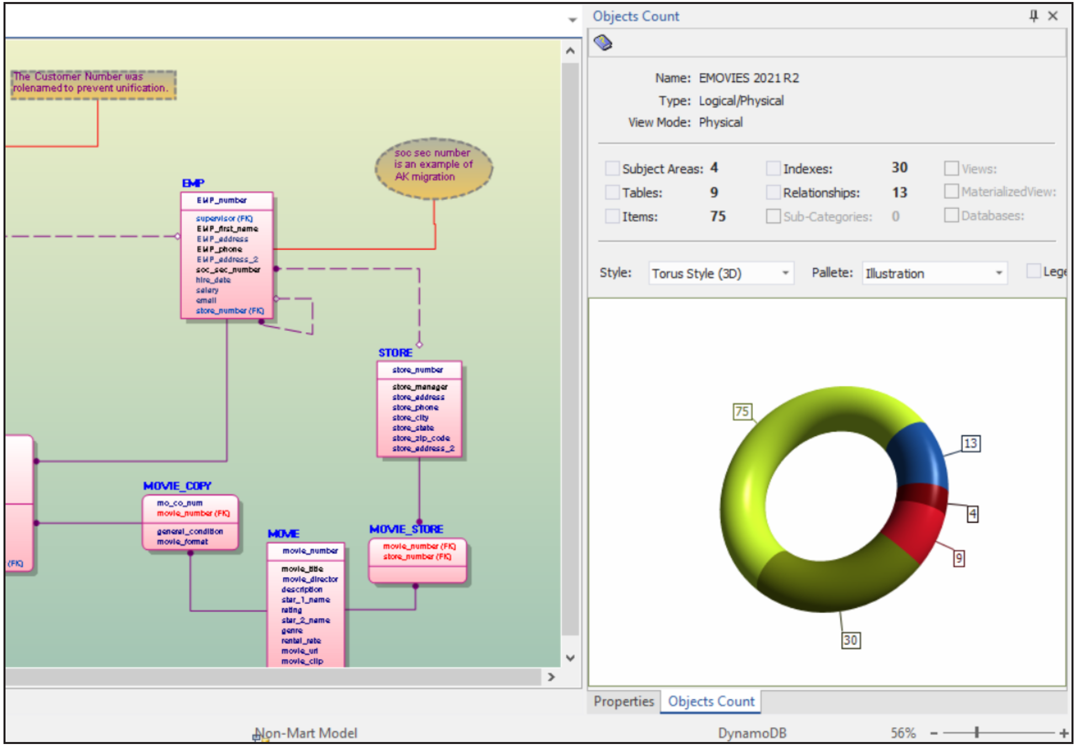
   The model derivation process starts.



Deriving

Constructing Model

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



Objects Count

Name: EMOVIES 2021 R2
Type: Logical/Physical
View Mode: Physical

□ Subject Areas: 4	□ Indexes: 30	□ Views:
□ Tables: 9	□ Relationships: 13	□ MaterializedView:
□ Items: 75	□ Sub-Categories: 0	□ Databases:

Style: Torus Style (3D) Pallete: Illustration □ Lege

In the **Objects Count** pane, you can view the details of converted database. The migration process converts and merges multiple tables, columns, and relationships to the DynamoDB format.

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

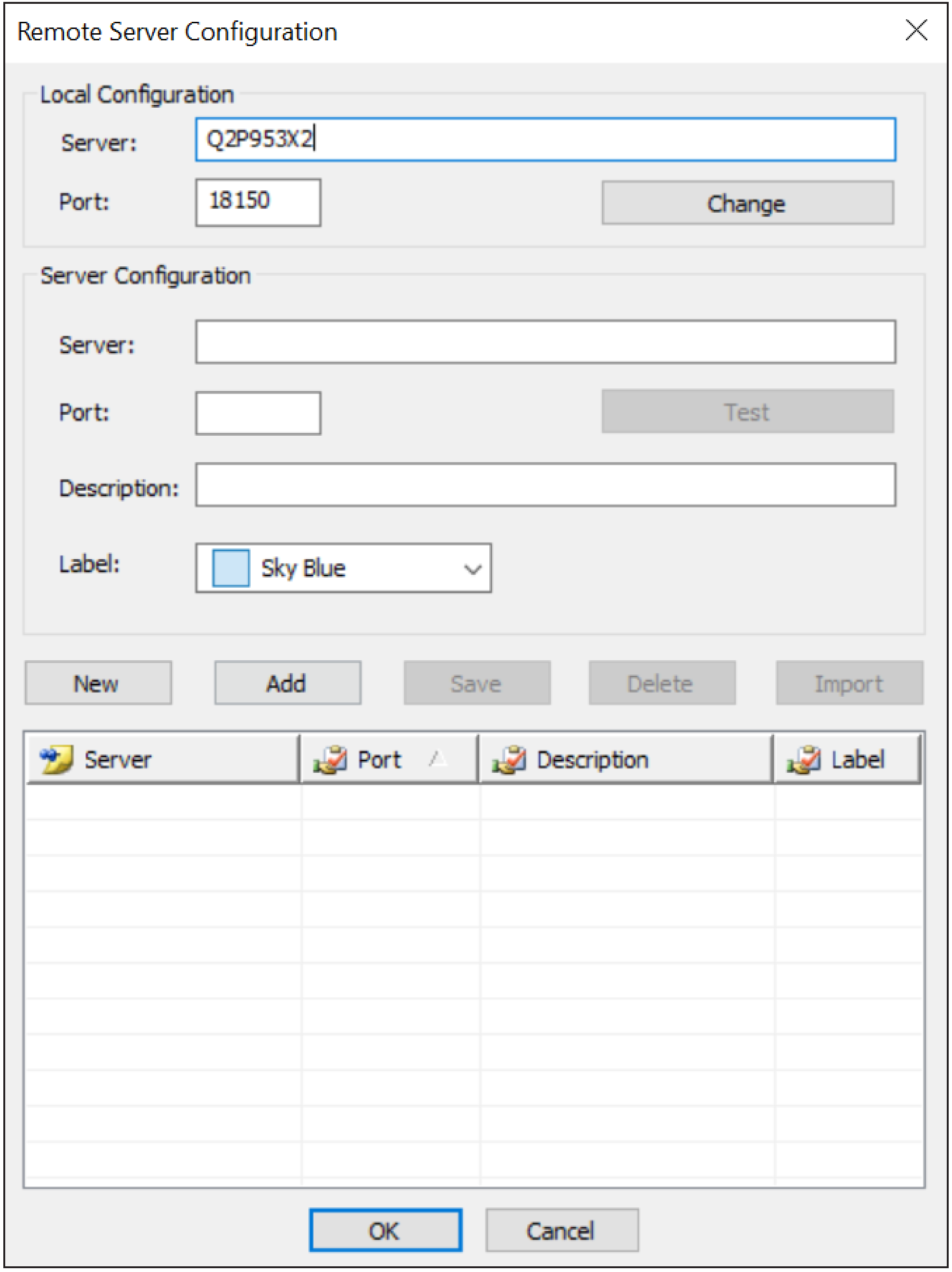
You can set up remote server configurations to schedule a job on the remote server. Configure multiple remote servers, label them, and manage them using the Remote Server Configuration pane.

> To schedule a job successfully on a remote server, ensure the following prerequisites are met:
>
> - Ensure the remote server is running.
> - On the local server, configure the remote server information in the Server Configuration section.
> - Similarly, on the remote server, configure the local server information in the Server Configuration section.

Also, save the server configuration as a predefined configuration for scheduling a job. Access these predefined configurations on the erwin DM Scheduler Event Details page. For more information on scheduling a job using the predefined remote server configurations, refer to Scheduling Jobs topic.

To create a remote server configuration, follow these steps:

1. On the ribbon, in the Remote group, click **Remote Server Configuration**.
   The Remote Server Configuration pane appears.



Remote Server Configuration

Local Configuration

Server: Q2P953X2

Port: 18150

Server Configuration

Server:

Port:

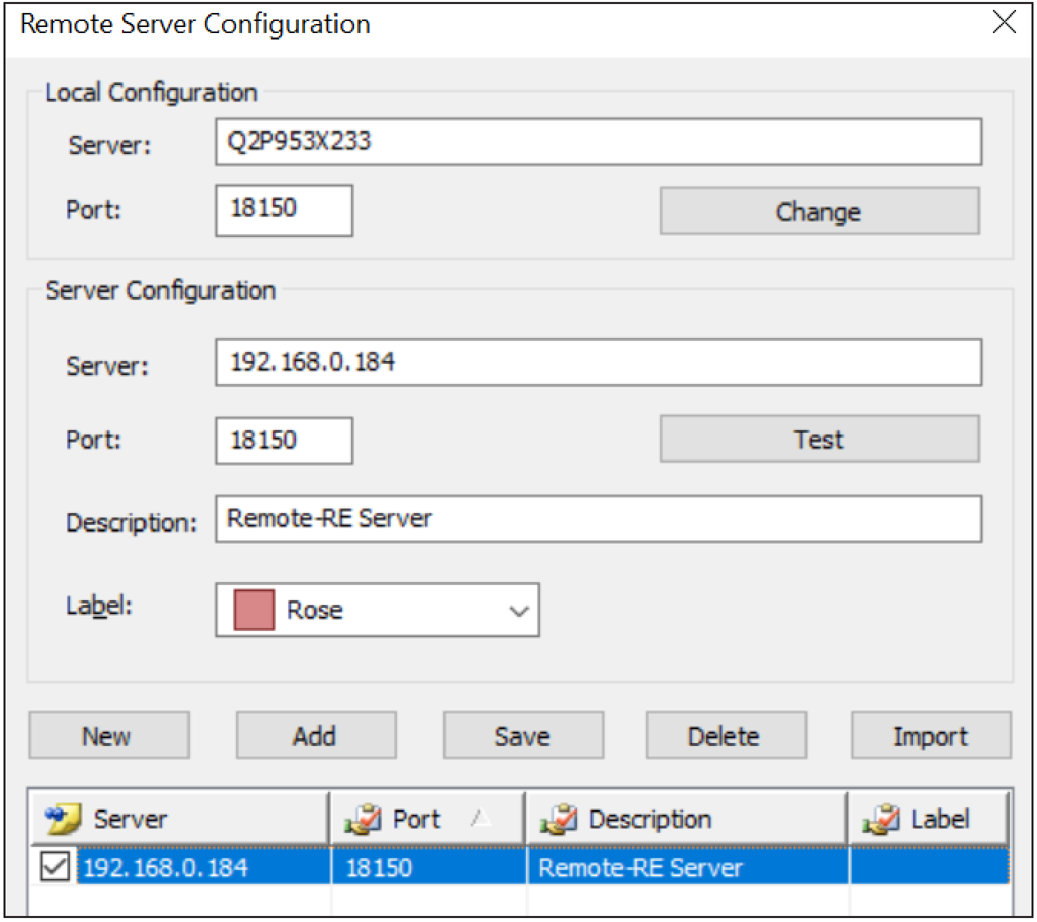
Description:

Label: Sky Blue

2. On the Remote Server Configuration pane, use the following options in the below table to set up server configurations.

Section	Option	Description
Local Configuration	Server	Enter the IP address of the local host. Displays local host name by default. It is recommended to use the IP address instead of local host name.
	Port	Enter the service port number for the remote scheduler. This field displays the default port number. Click **Change** to update the port number.
Server Configuration	Server	Enter the IP address of the remote server.
	Port	Enter the port number for remote server. Ensure the remote server is functional before testing the connection.
	Description	Enter a description for the remote server.
	Label	Select a label color to categorize the server configurations.

3. Once you have added remote server configuration, click **Test**.
   The erwin DM Scheduler appears on a successful connection.
4. Click **Add**.
   The remote server configuration is added to the list on the Remote Server Configuration pane.



Remote Server Configuration

Local Configuration

Server: Q2P953X233

Port: 18150

Server Configuration

Server: 192.168.0.184

Port: 18150

Description: Remote-RE Server

Label: Rose

Server	Port	Description	Label
☑ 192.168.0.184	18150	Remote-RE Server	

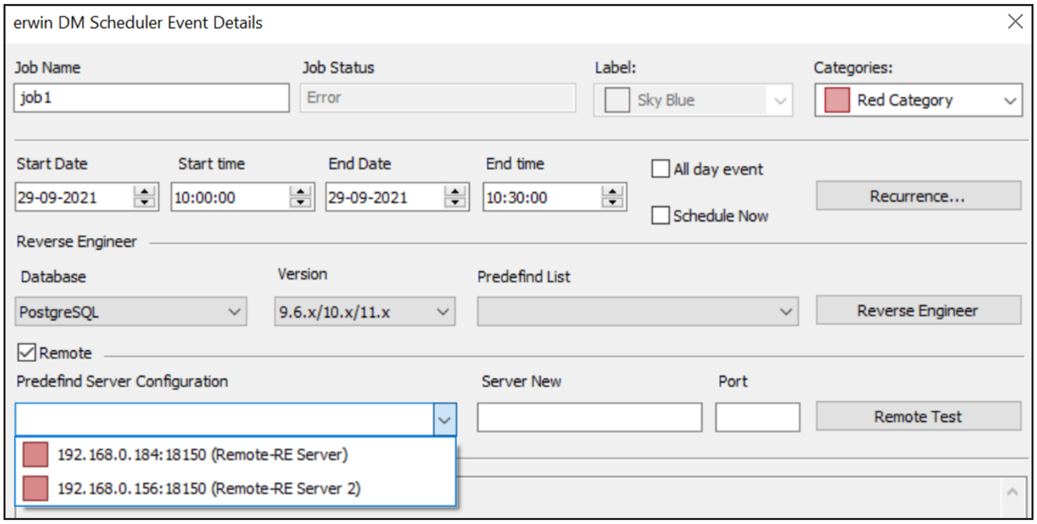
5. Once you have created a remote server configuration, on the Remote Server Configuration pane, use one of the following options:

   - **New**: Use this option to create a new reverse engineering configuration. Selecting this option resets the Server Configuration section.
   - **Add**: Use this option to add the new configuration. The added configurations are displayed on the configurations list.
   - **Save**: Use this option to save the changes to selected server configuration on the list.
   - **Delete**: Use this option to delete the selected configurations on the list.

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

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

6. Click **OK**.
   The remote server 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.



erwin DM Scheduler Event Details

Job Name: job1
Job Status: Error
Label: Sky Blue
Categories: 🟥 Red Category

Start Date: 29-09-2021
Start time: 10:00:00
End Date: 29-09-2021
End time: 10:30:00
□ All day event
□ Schedule Now

Reverse Engineer

Database: PostgreSQL
Version: 9.6.x/10.x/11.x
Predefind List:

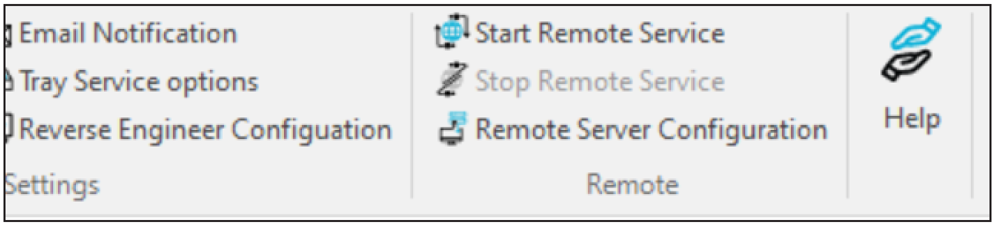
☑ Remote

Predefind Server Configuration:
Server New:
Port:

🟥 192.168.0.184:18150 (Remote-RE Server)
🟥 192.168.0.156:18150 (Remote-RE Server 2)

Once you have configured the remote server, you can start or stop the remote services. On the ribbon, in the Remote group, click either of the following options:

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



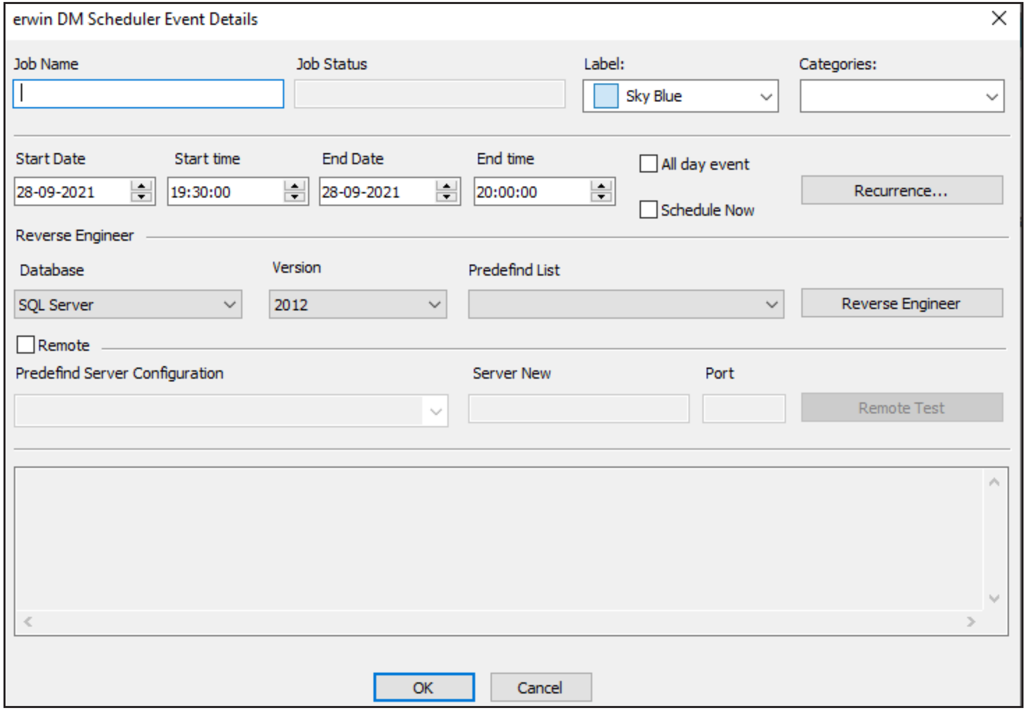
Scheduling Remote Jobs

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.



erwin DM Scheduler Event Details

Job Name:

Job Status:

Label: Sky Blue

Categories:

Start Date: 28-09-2021

Start time: 19:30:00

End Date: 28-09-2021

End time: 20:00:00

□ All day event

□ Schedule Now

Reverse Engineer

Database: SQL Server

Version: 2012

Predefind List:

□ Remote

Predefind Server Configuration:

Server New:

Port:

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	
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 sched-

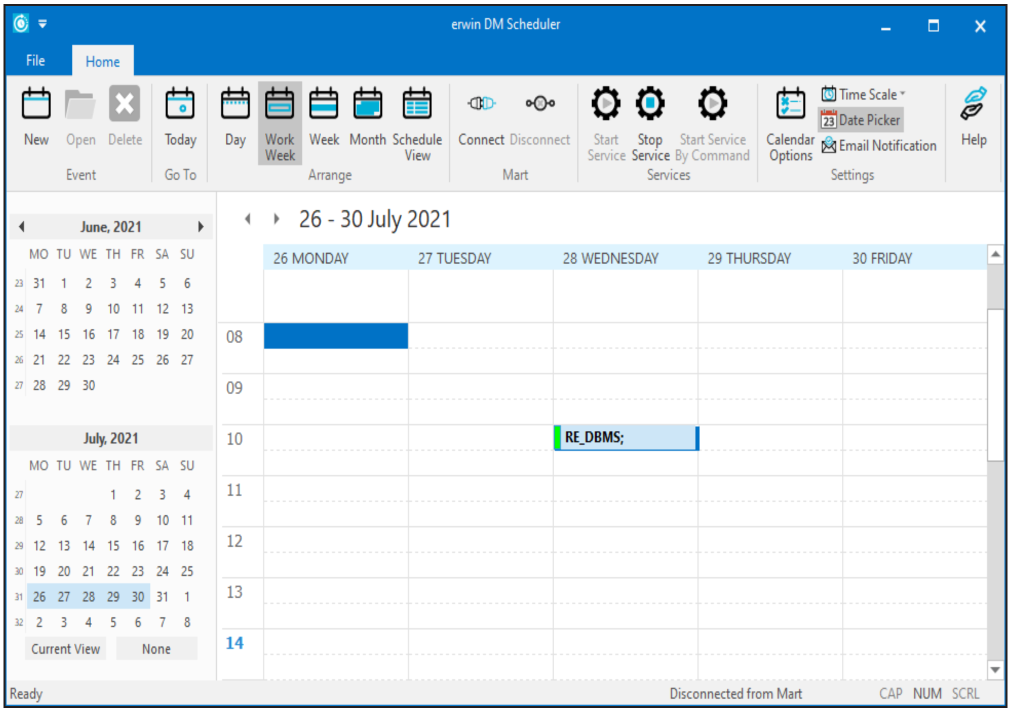
Option	Description	Additional Information
		uled at the same time with the Schedule Now option, it randomly selects a job to run. Therefore, it is recommended that you do not schedule multiple jobs to run at the same time.
Start Time	Specifies the start time for a job	
End Date	Specifies the end date 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.
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

Option	Description	Additional Information
		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 engineering	
Predefined Server Configuration	Displays the lists of predefined remote servers for reverse engineering	
Server New		
Port	Specifies the port number	

Option	Description	Additional Information
	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.

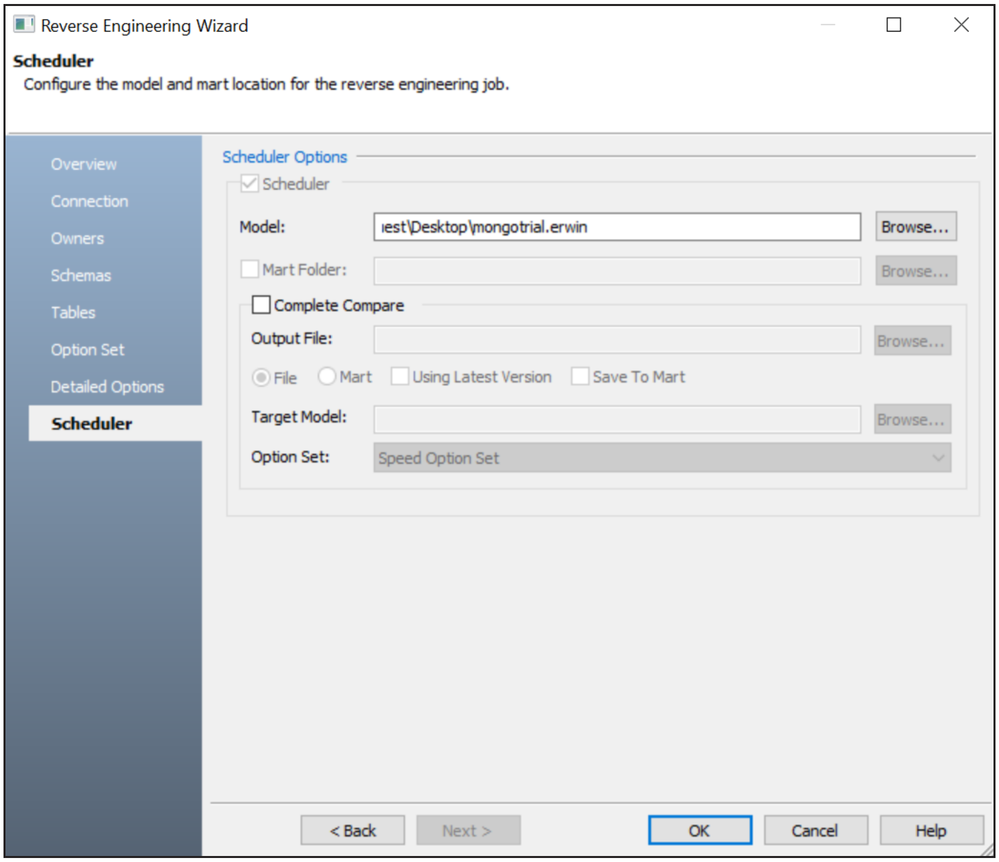


Depending on the settings you make and the job duration that you set, the job tile displays the following information about the job:

- 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.



Reverse Engineering Wizard

Scheduler

Configure the model and mart location for the reverse engineering job.

Scheduler Options

☑ Scheduler

Model: ıest\Desktop\mongotrial.erwin

□ Mart Folder:

□ Complete Compare

Output File:

● File ○ Mart □ Using Latest Version □ Save To Mart

Target Model:

Option Set: Speed Option Set

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

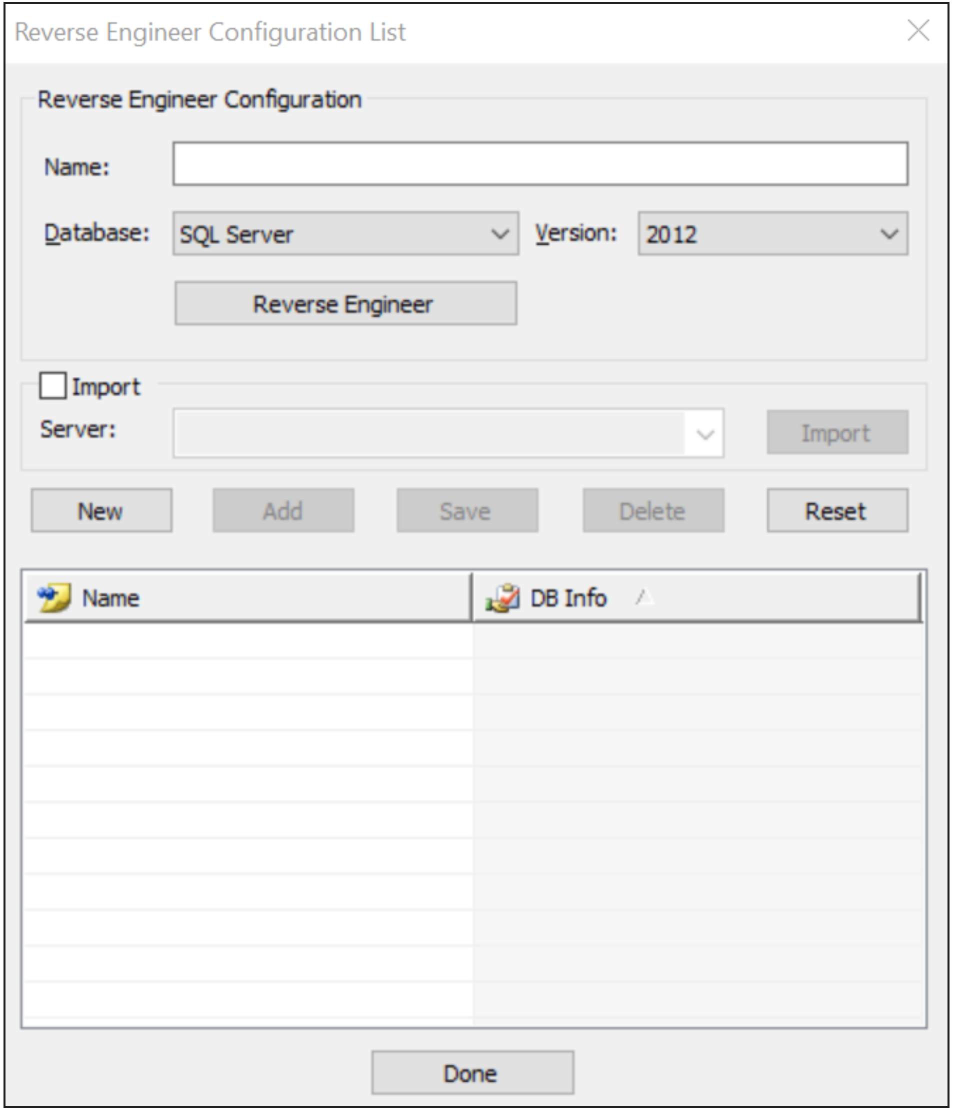
ewin DM Scheduler can now run multiple jobs in parallel on a remote server at the same time.

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.



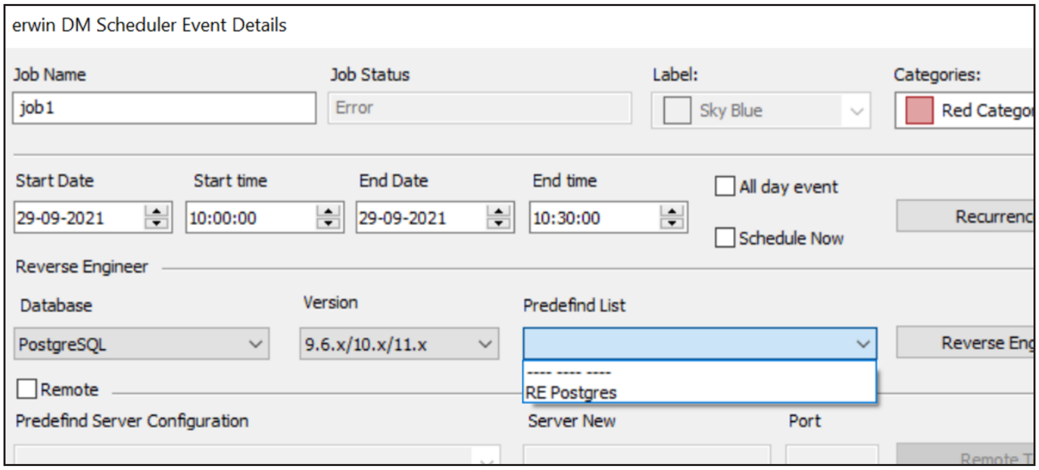
2. On the Reverse Engineer Configuration List, use the following options in the below table to create or import configurations.

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.



erwin DM Scheduler Event Details

Job Name: job1

Job Status: Error

Label: Sky Blue

Categories: Red Categor

Start Date: 29-09-2021

Start time: 10:00:00

End Date: 29-09-2021

End time: 10:30:00

□ All day event

□ Schedule Now

Reverse Engineer

Database: PostgreSQL

Version: 9.6.x/10.x/11.x

Predefind List:

---- ---- ----
RE Postgres

□ Remote

Predefind Server Configuration

Server New

Port

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.

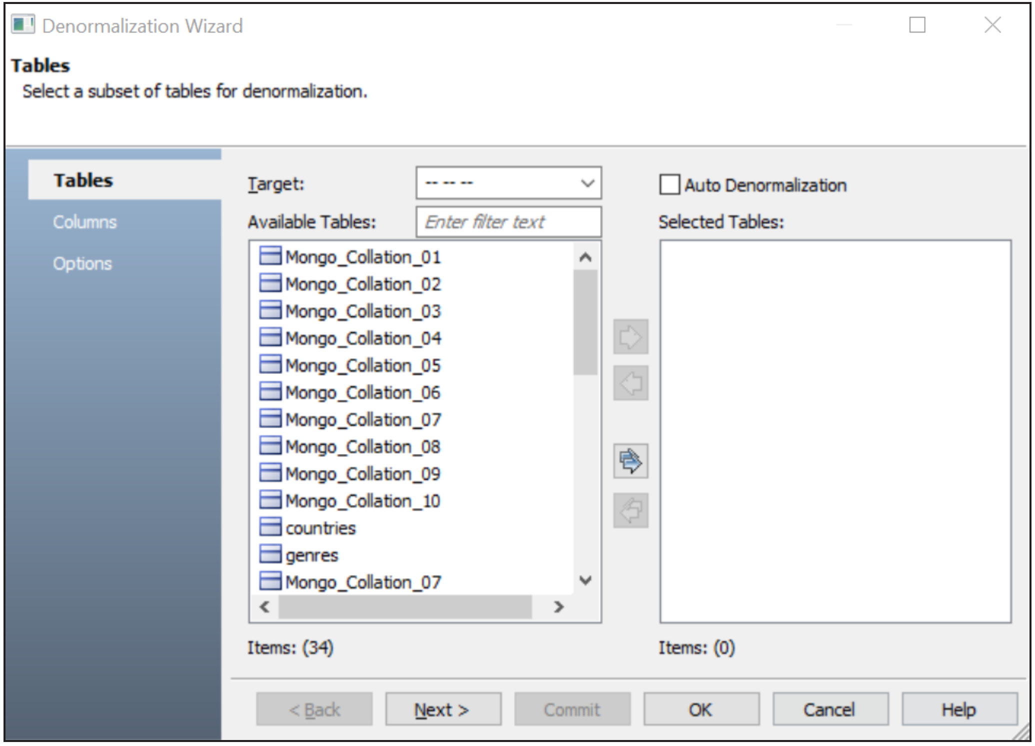
Advanced Denormalization

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.



Tables

Select a subset of tables for denormalization.

Target: -- -- --

Available Tables: Enter filter text

□ Auto Denormalization

Selected Tables:

- Mongo_Collation_01
- Mongo_Collation_02
- Mongo_Collation_03
- Mongo_Collation_04
- Mongo_Collation_05
- Mongo_Collation_06
- Mongo_Collation_07
- Mongo_Collation_08
- Mongo_Collation_09
- Mongo_Collation_10
- countries
- genres
- Mongo_Collation_07

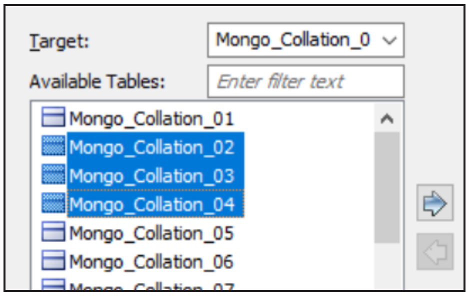
Items: (34)

Items: (0)

> 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.

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

2.

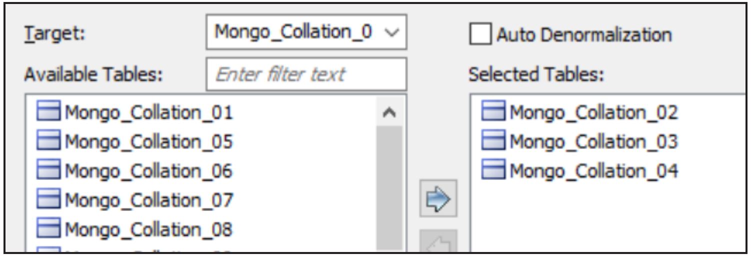


Target: Mongo_Collation_0

Available Tables: Enter filter text

- Mongo_Collation_01
- Mongo_Collation_02
- Mongo_Collation_03
- Mongo_Collation_04
- Mongo_Collation_05
- Mongo_Collation_06
- Mongo_Collation_07

This moves the selected tables under Selected Tables.



Target: Mongo_Collation_0

□ Auto Denormalization

Available Tables:

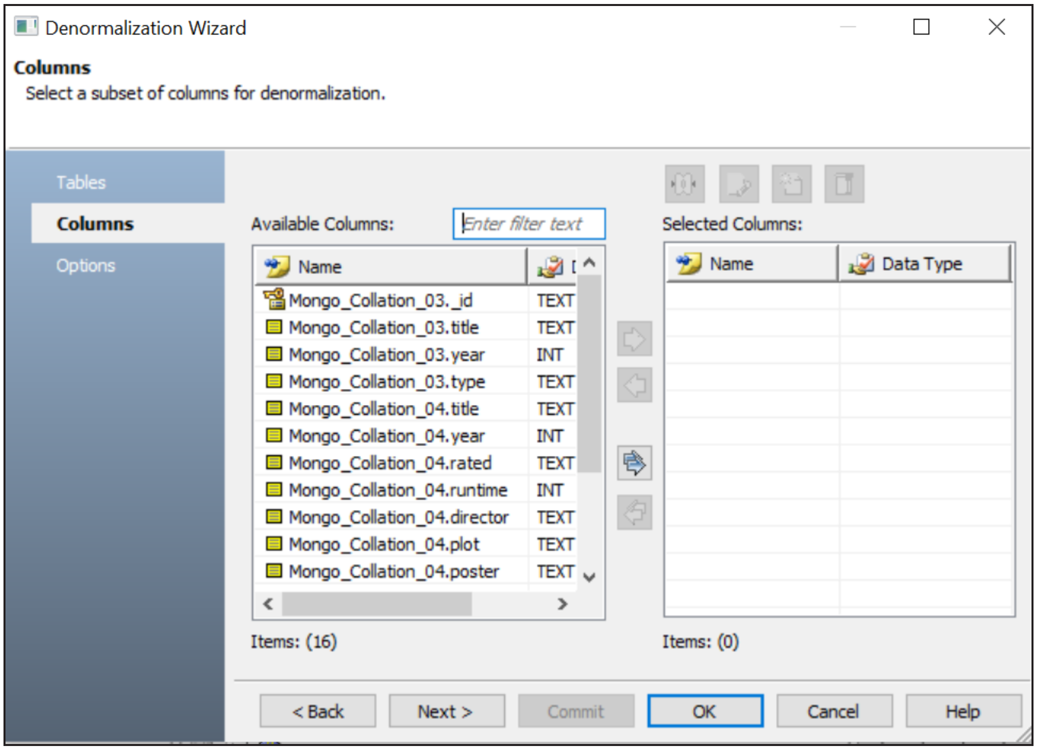
- Mongo_Collation_01
- Mongo_Collation_05
- Mongo_Collation_06
- Mongo_Collation_07
- Mongo_Collation_08

Selected Tables:

- Mongo_Collation_02
- Mongo_Collation_03
- Mongo_Collation_04

3. Click **Next**.

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



Denormalization Wizard

Columns

Select a subset of columns for denormalization.

Available Columns:

Name	[
Mongo_Collation_03._id	TEXT
Mongo_Collation_03.title	TEXT
Mongo_Collation_03.year	INT
Mongo_Collation_03.type	TEXT
Mongo_Collation_04.title	TEXT
Mongo_Collation_04.year	INT
Mongo_Collation_04.rated	TEXT
Mongo_Collation_04.runtime	INT
Mongo_Collation_04.director	TEXT
Mongo_Collation_04.plot	TEXT
Mongo_Collation_04.poster	TEXT

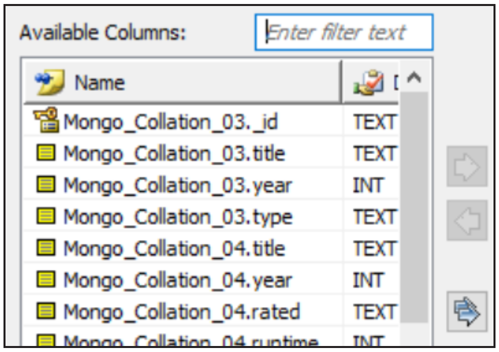
Items: (16)

Selected Columns:

Name	Data Type

Items: (0)

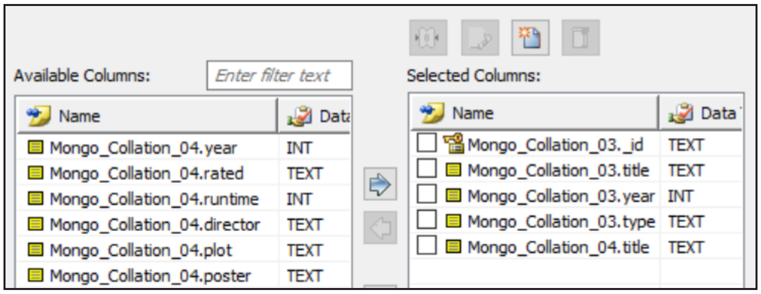
4. Under **Available Columns**, select the that you want to merge. Then, click ⇨.



Available Columns: Enter filter text

Name	D
Mongo_Collation_03._id	TEXT
Mongo_Collation_03.title	TEXT
Mongo_Collation_03.year	INT
Mongo_Collation_03.type	TEXT
Mongo_Collation_04.title	TEXT
Mongo_Collation_04.year	INT
Mongo_Collation_04.rated	TEXT
Mongo_Collation_04.runtime	INT

This moves the selected databases under Selected Columns.



Available Columns: Enter filter text

Name	Data
Mongo_Collation_04.year	INT
Mongo_Collation_04.rated	TEXT
Mongo_Collation_04.runtime	INT
Mongo_Collation_04.director	TEXT
Mongo_Collation_04.plot	TEXT
Mongo_Collation_04.poster	TEXT

Selected Columns:

Name	Data
□ Mongo_Collation_03._id	TEXT
□ Mongo_Collation_03.title	TEXT
□ Mongo_Collation_03.year	INT
□ Mongo_Collation_03.type	TEXT
□ Mongo_Collation_04.title	TEXT

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.

Git Integration

Starting erwin Data Modeler (DM) 12.0, you can connect erwin DM to Git repositories. This allows you to push Forward Engineering (FE) scripts for an Online Mart Model to GitLab or GitHub. Working with these repositories help you in:

- DevOps adoption
- collaborating amongst team members
- version control
- workflow management
- data integrity

To connect erwin DM to Git, ensure that

- erwin DM is connected to erwin Mart Server. To know how to connect erwin DM to erwin Mart Server, refer to the Connect to Mart topic.
- 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.

For information on connecting to Git, refer to the Connecting to Git Repositories topic.

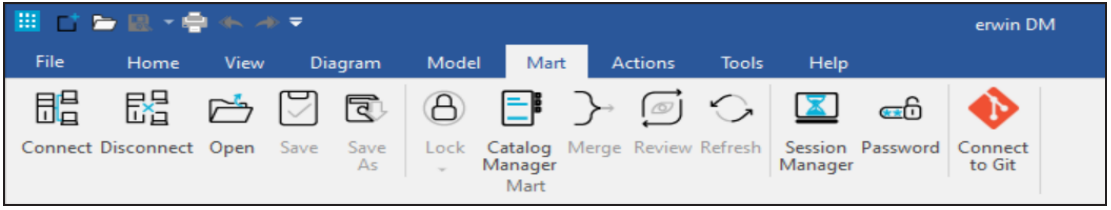
Connecting to Git Repositories

Once, you are connected to erwin Mart Server, connect erwin DM to a Git repository. The Git repository may be hosted on GitLab or GitHub.

To connect Git repositories to erwin DM, follow these steps:

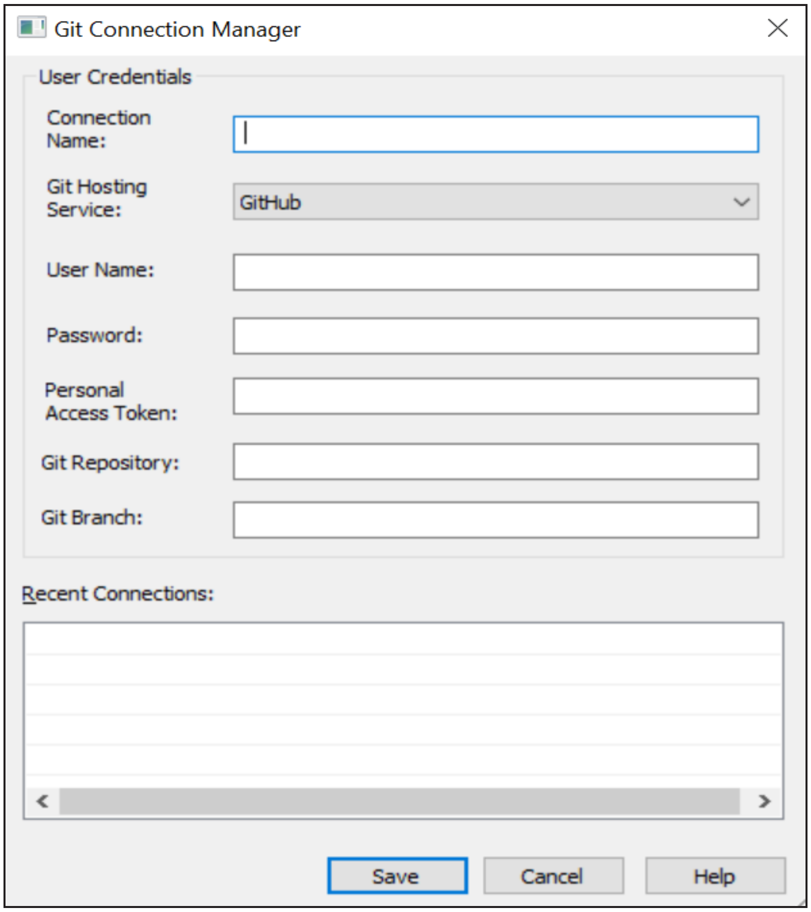
1. On the ribbon, click **Mart**.

   The available options appear.



2. Click **Connect to Git**.

   The Connect to Git page appears.



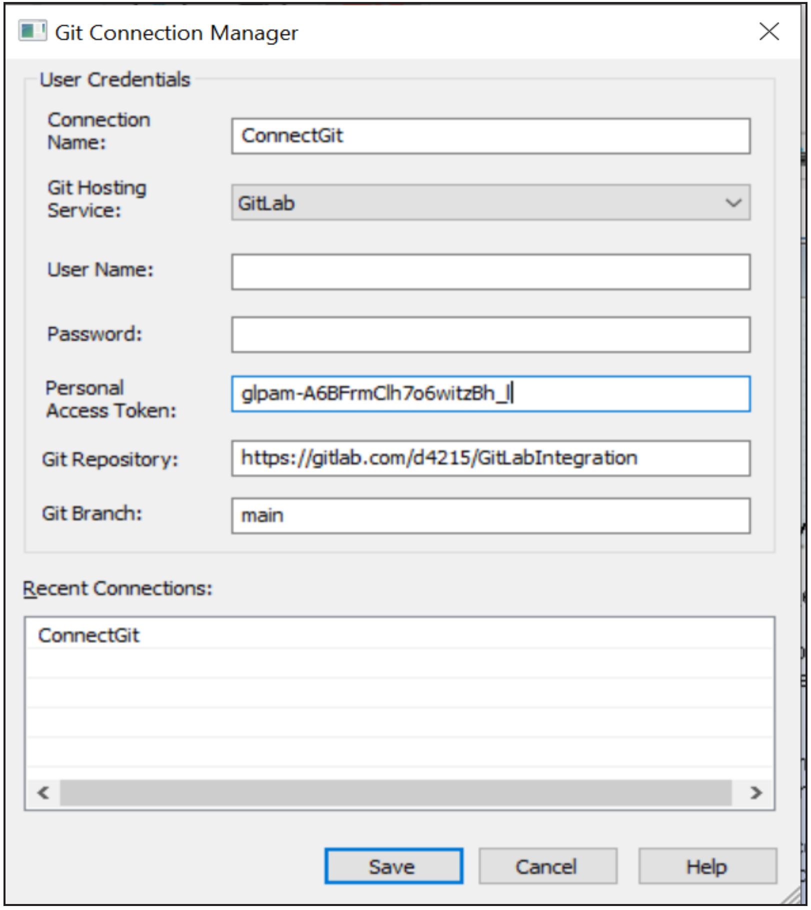
3. Enter appropriate values in the fields. Refer to the following table for field descriptions.

Field Name	Description	Additional Information
Connection	Specifies the user defined	For example, ConnectGit.

Field Name	Description	Additional Information
Name	connection name	You can create multiple connections each owing to a 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
User Name	Specifies the user name to log on to the git hosting service (GitLab or GitHub)	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 the git hosting service	For example, glpam-A6BFrmClh7o6witzBh_l
GIT Repository	Specifies the URL path of a git repository where you want to push the Forward engineering script	For example, https://-gitlab.com/d4215/GitLabIntegration
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.



User Credentials

Connection Name: ConnectGit

Git Hosting Service: GitLab

User Name:

Password:

Personal Access Token: glpam-A6BFrmClh7o6witzBh_l

Git Repository: https://gitlab.com/d4215/GitLabIntegration

Git Branch: main

Recent Connections:

ConnectGit

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

Committing Forward Engineering Scripts

You can commit Forward Engineering (FE) scripts to a Git repository using:

- **Scenario 1: Committing FE scripts for the first time**:

  Use **Forward Engineer Schema Generation Wizard** wizard to commit a physical database schema or FE script from an Online Mart Model. Ensure that your file names are unique. For more information, refer to the Scenario 1: Committing FE Scripts for the First Time topic.

- **Scenario 2: Committing alter scripts after making changes to a model**:

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

- **Commit and append an alter script to an existing file in Git repository**: To append the alter script to an existing file, ensure that Auto Append check box is selected and correct File Name and Git Path is set on the Commit to Git page.
- **Commit and create a new alter script file in the Git repository**: 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. This creates and commits a new file with the following naming convention: <File Name>_YYYY-MM-DD_HH-MM-SS.

For more information, refer to the Scenario 2: Committing Alter Scripts After Making Changes to a Model topic.

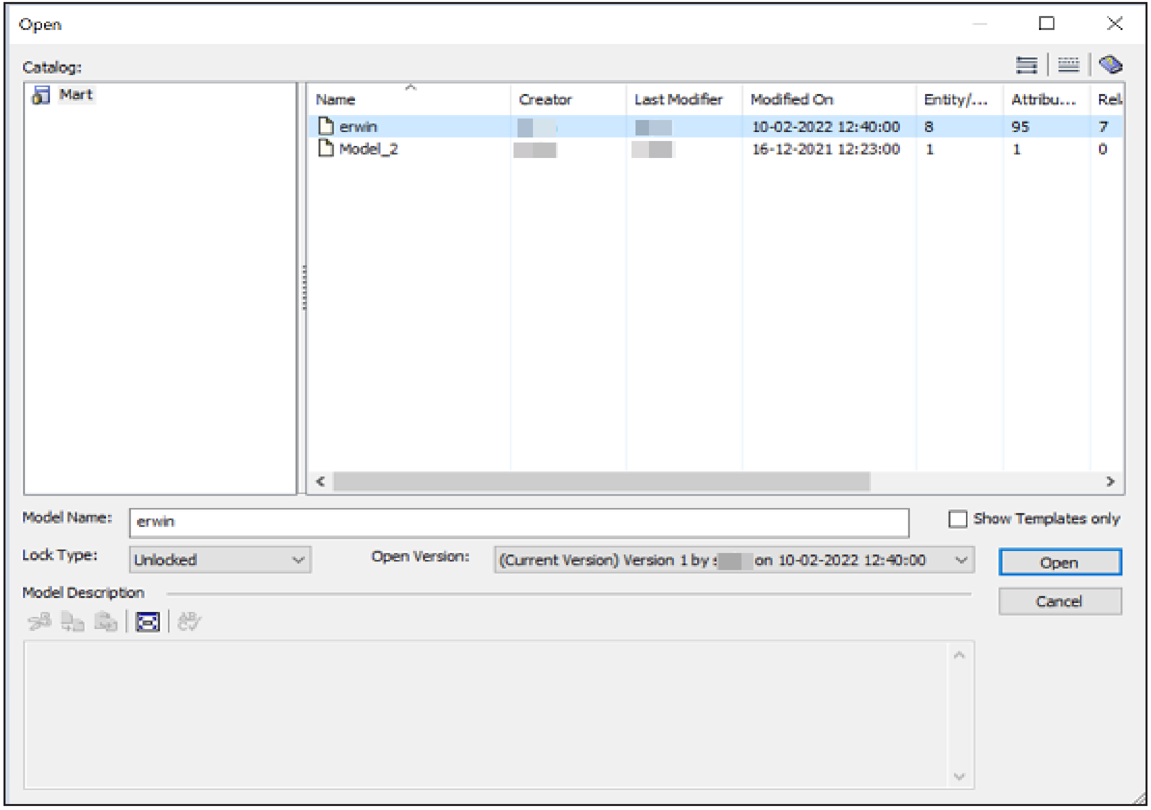
Scenario 1: Committing FE Scripts for the First Time

With Forward Engineer (FE) Schema Generation Wizard, you can generate a physical database schema or FE script. You can push this FE script for an Online Mart Model to a Git repository. The Git repository may be hosted on GitLab or GitHub. This enables your team to utilize all the advantages of storing a code on a Git repository like efficient collaboration with team members and version control.

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

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

   The Open page appears.



Name	Creator	Last Modifier	Modified On	Entity/...	Attribu...	Rel.
erwin	[redacted]	[redacted]	10-02-2022 12:40:00	8	95	7
Model_2	[redacted]	[redacted]	16-12-2021 12:23:00	1	1	0

Model Name: erwin

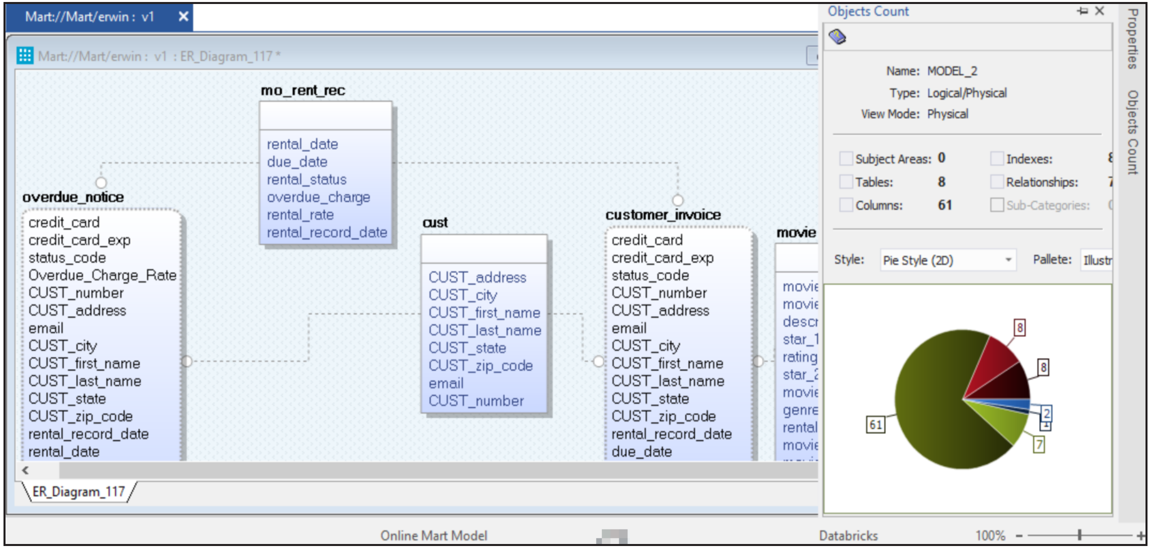
□ Show Templates only

Lock Type: Unlocked

Open Version: (Current Version) Version 1 by [redacted] on 10-02-2022 12:40:00

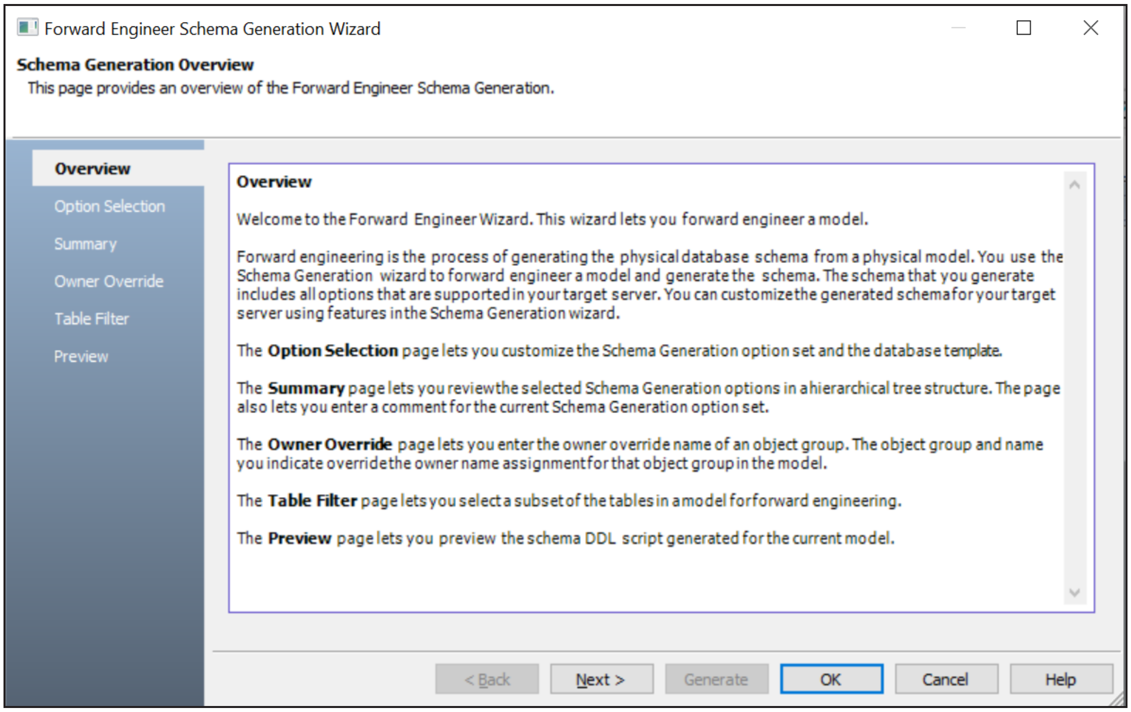
2. Click the required model, and then click **Open**.

   The Online Mart Model opens.



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

   The **Forward Engineer Schema Generation Wizard** appears.



Forward Engineer Schema Generation Wizard

Schema Generation Overview

This page provides an overview of the Forward Engineer Schema Generation.

Overview

Welcome to the Forward Engineer Wizard. This wizard lets you forward engineer a model.

Forward engineering is the process of generating the physical database schema from a physical model. You use the Schema Generation wizard to forward engineer a model and generate the schema. The schema that you generate includes all options that are supported in your target server. You can customize the generated schema for your target server using features in the Schema Generation wizard.

The **Option Selection** page lets you customize the Schema Generation option set and the database template.

The **Summary** page lets you review the selected Schema Generation options in a hierarchical tree structure. The page also lets you enter a comment for the current Schema Generation option set.

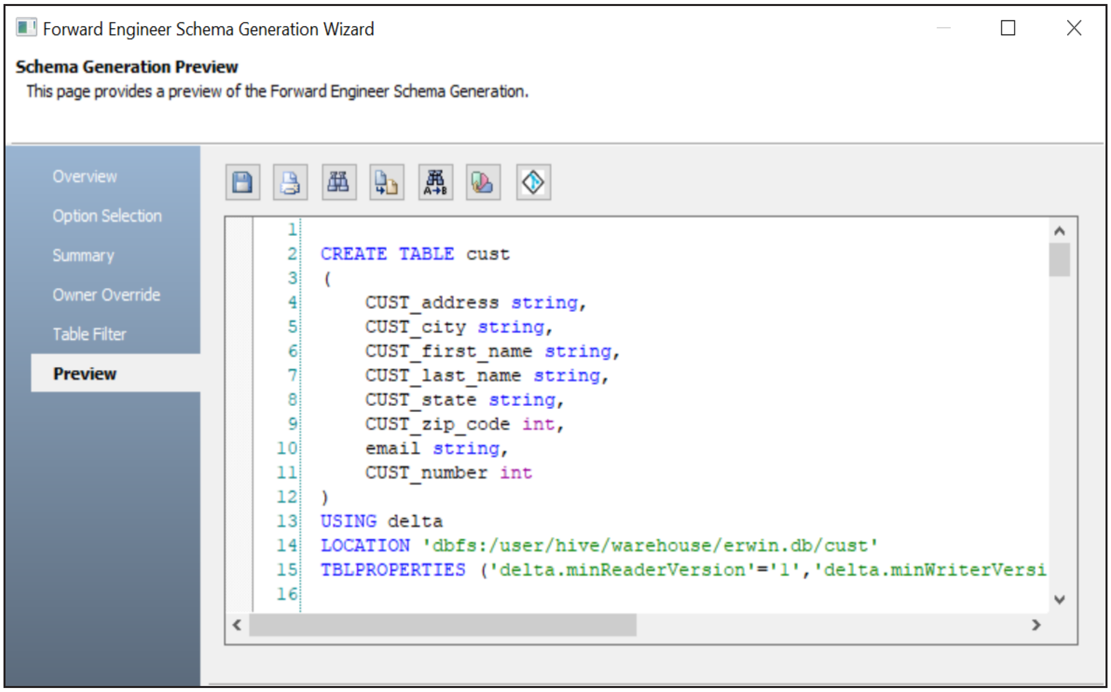
The **Owner Override** page lets you enter the owner override name of an object group. The object group and name you indicate override the owner name assignment for that object group in the model.

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

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

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

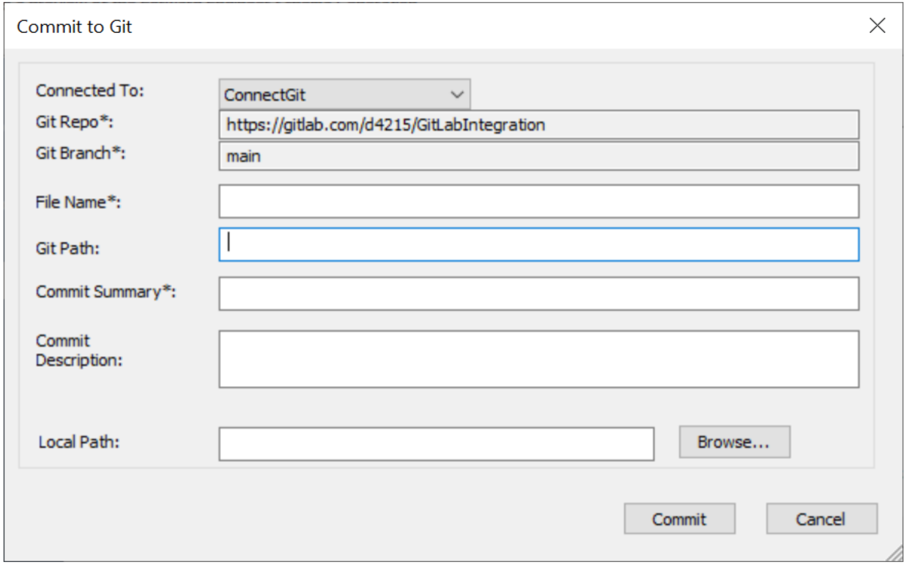
The FE script appears. For example, in the following image the Preview section displays FE script of a Databricks database. For more information on generating FE scripts, refer to the Forward Engineering/Schema Generation for Databases topic.


Forward Engineer Schema Generation Wizard
Schema Generation Preview
This page provides a preview of the Forward Engineer Schema Generation.
Overview
Option Selection
Summary
Owner Override
Table Filter
Preview

```

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
)
USING delta
LOCATION 'dbfs:/user/hive/warehouse/erwin.db/cust'
TBLPROPERTIES ('delta.minReaderVersion'='1','delta.minWriterVersi

```

5. Click ◈.

   The Commit to Git page appears.



Commit to Git

Connected To: ConnectGit

Git Repo*: https://gitlab.com/d4215/GitLabIntegration

Git Branch*: main

File Name*:

Git Path:

Commit Summary*:

Commit Description:

Local Path:

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

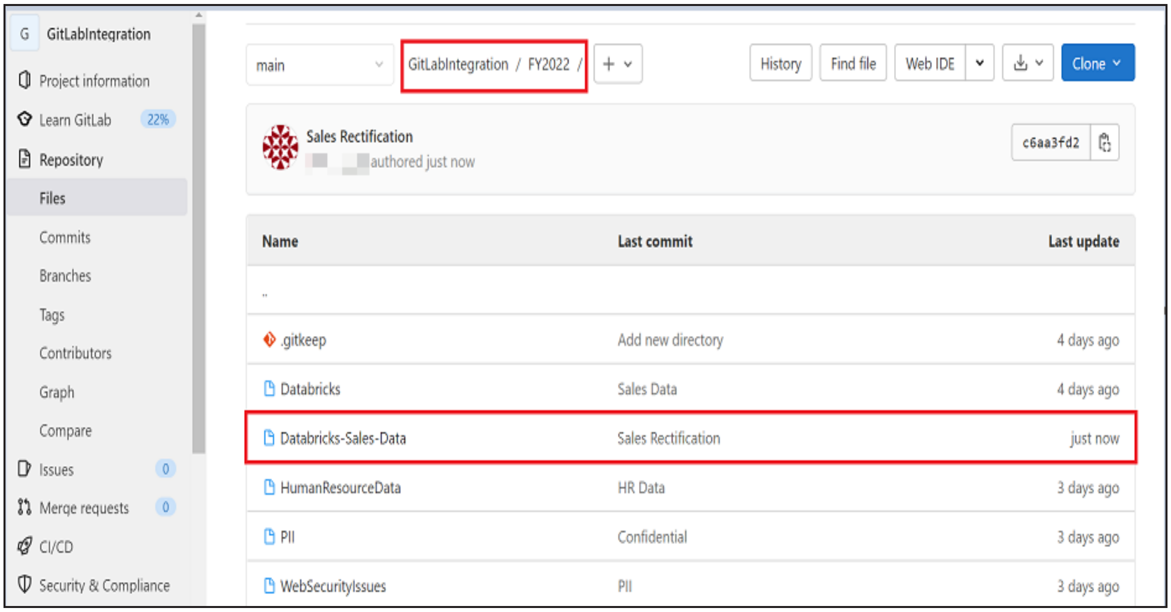
Field Name	Description	Additional Information
Connected To	Specifies the connection that connects erwin DM to a Git repository	For example, ConnectGit.
Git Repo	Specifies the Git repository that was set for the connection in the Git Connection Manager	For example, https://gitlab.com/d4215/GitLabIntegration is set for the ConnectGit connection. This field autopopulates based on the value set in the Connected To field.
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 value set in the Connected To field.
File Name	Specifies the user-defined name of the FE script file being committed to a Git repository	For example, Databricks-Sales-Data.sql Ensure that you use unique file names. If a file name matches with an existing file in the same location

Field Name	Description	Additional Information
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.
Commit Description	Specifies the description of the push commit	For example, Forward schema script includes December sales data.
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.

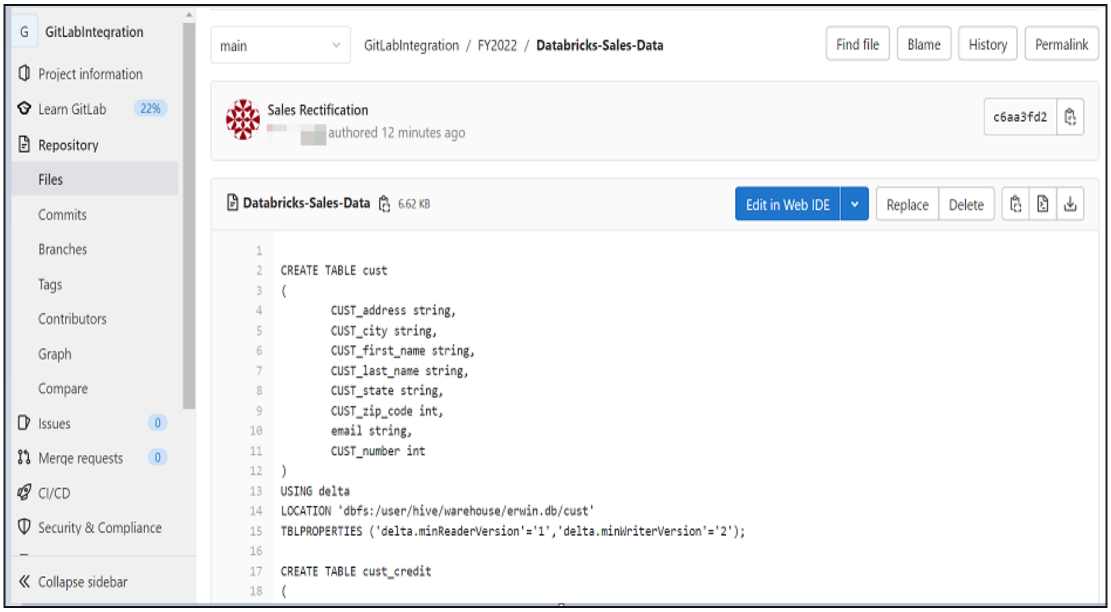
   For example, in the following image FE script is committed to a GitLab repository in a file, Databricks-Sales-Data with a commit summary, Sales Rectification is committed using the main branch.



Sales Rectification
[redacted] authored just now
c6aa3fd2

Name	Last commit	Last update
..		
.gitkeep	Add new directory	4 days ago
Databricks	Sales Data	4 days ago
Databricks-Sales-Data	Sales Rectification	just now
HumanResourceData	HR Data	3 days ago
PII	Confidential	3 days ago
WebSecurityIssues	PII	3 days ago

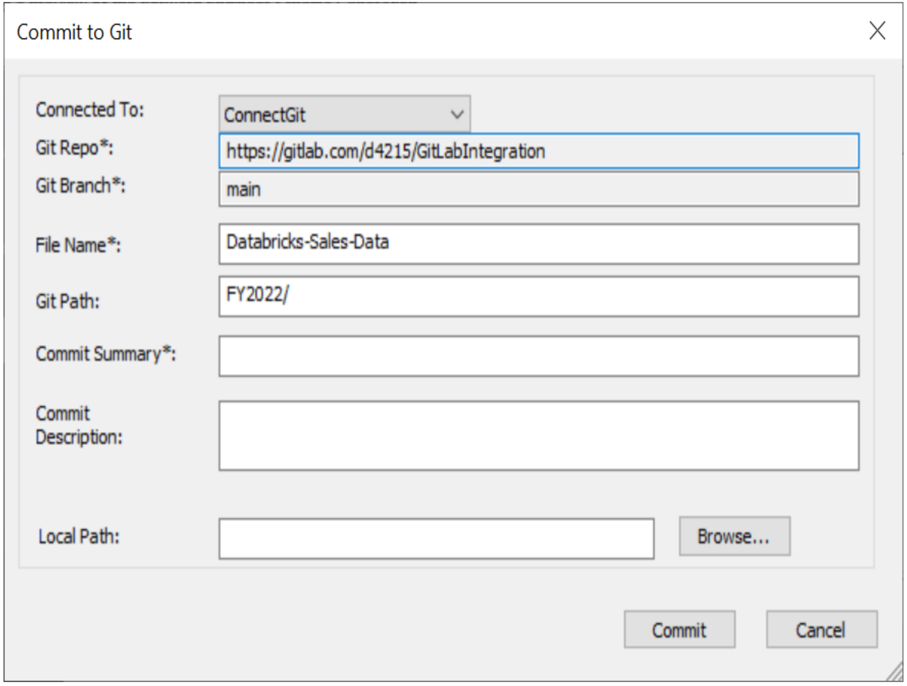
You can click the file to review its content. For example, in the following image, Databricks-Sales-Data's content is visible in a GitLab repository.



```

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
)
USING delta
LOCATION 'dbfs:/user/hive/warehouse/erwin.db/cust'
TBLPROPERTIES ('delta.minReaderVersion'='1','delta.minWriterVersion'='2');

CREATE TABLE cust_credit
(
```

You can use FE Schema Generation Wizard to commit FE script using the same connection again. This time the Commit to Git page 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/.



Commit to Git

Connected To: ConnectGit

Git Repo*: https://gitlab.com/d4215/GitLabIntegration

Git Branch*: main

File Name*: Databricks-Sales-Data

Git Path: FY2022/

Commit Summary*:

Commit Description:

Local Path:

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

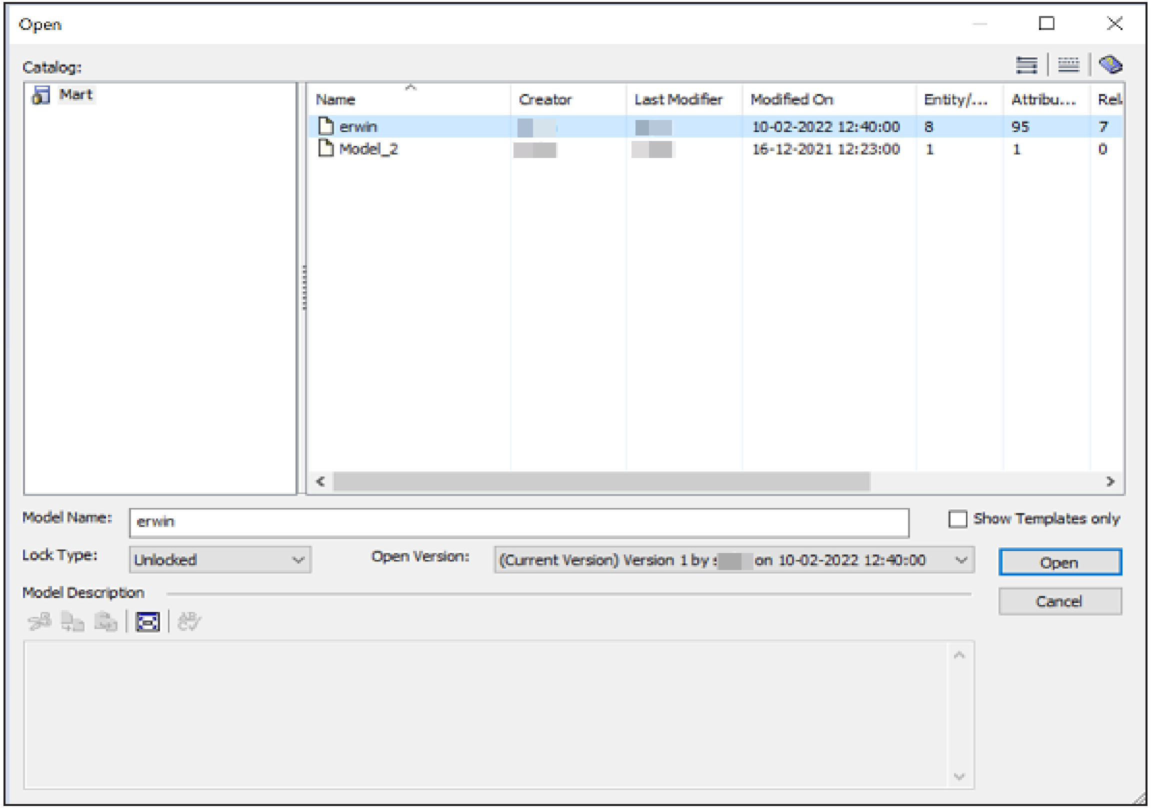
Scenario 2: Committing Alter Scripts After Making Changes to a Model

With Forward Engineer (FE) Alter Schema Generation Wizard, you can generate an alter script for a database after you make changes to a model. You can push this alter script to a Git repository and append it to an existing FE script in the Git repository.

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

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

   The Open page appears.



Name	Creator	Last Modifier	Modified On	Entity/...	Attribu...	Rel.
erwin	[redacted]	[redacted]	10-02-2022 12:40:00	8	95	7
Model_2	[redacted]	[redacted]	16-12-2021 12:23:00	1	1	0

Model Name: erwin

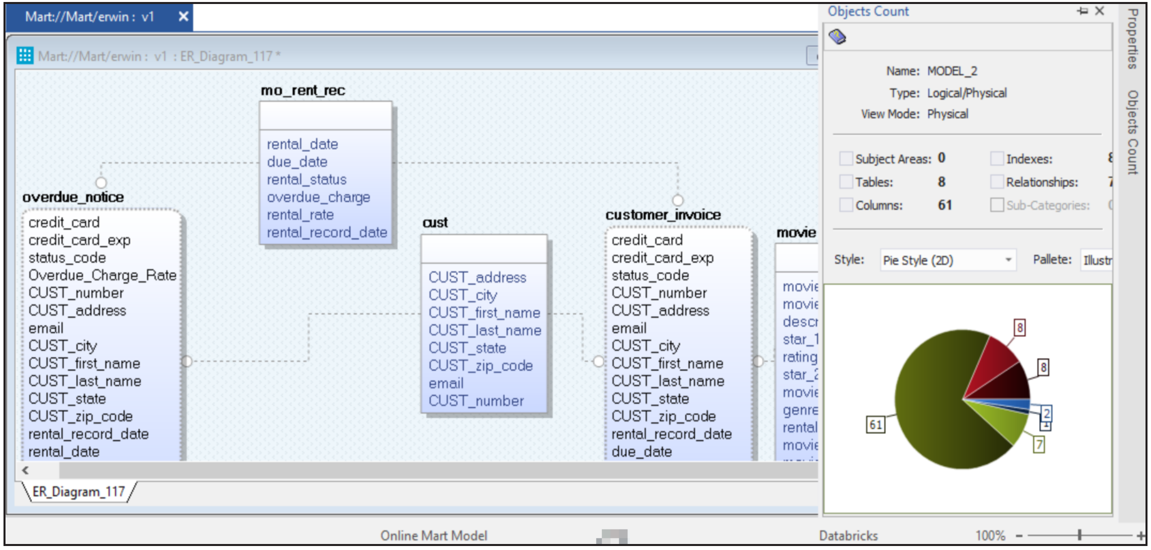
☐ Show Templates only

Lock Type: Unlocked

Open Version: (Current Version) Version 1 by s[redacted] on 10-02-2022 12:40:00

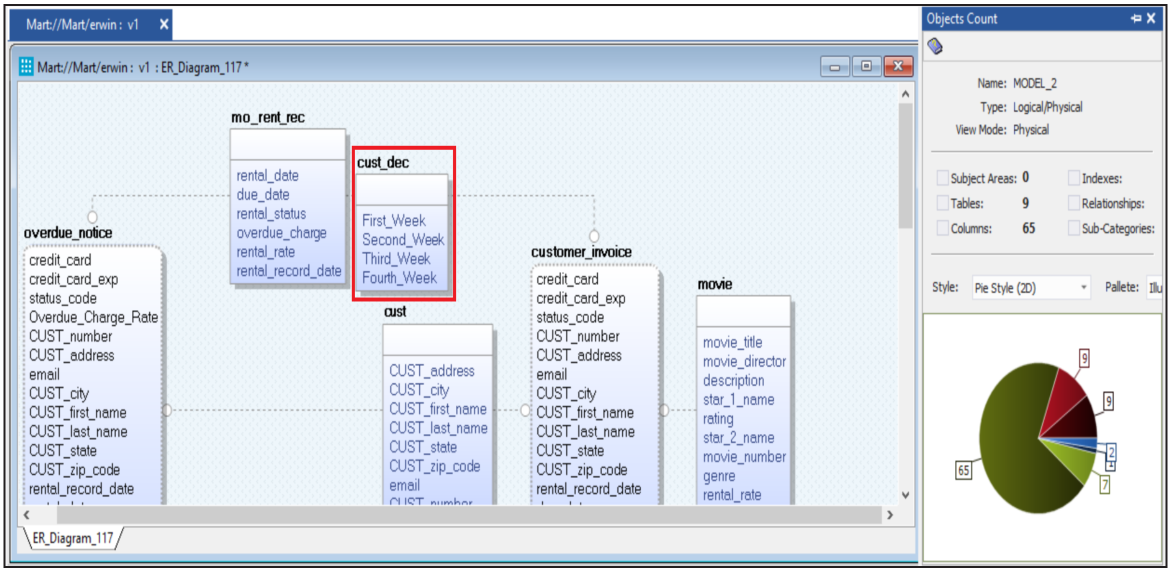
2. Click the required model, and then click **Open**.

   The Online Mart Model opens.



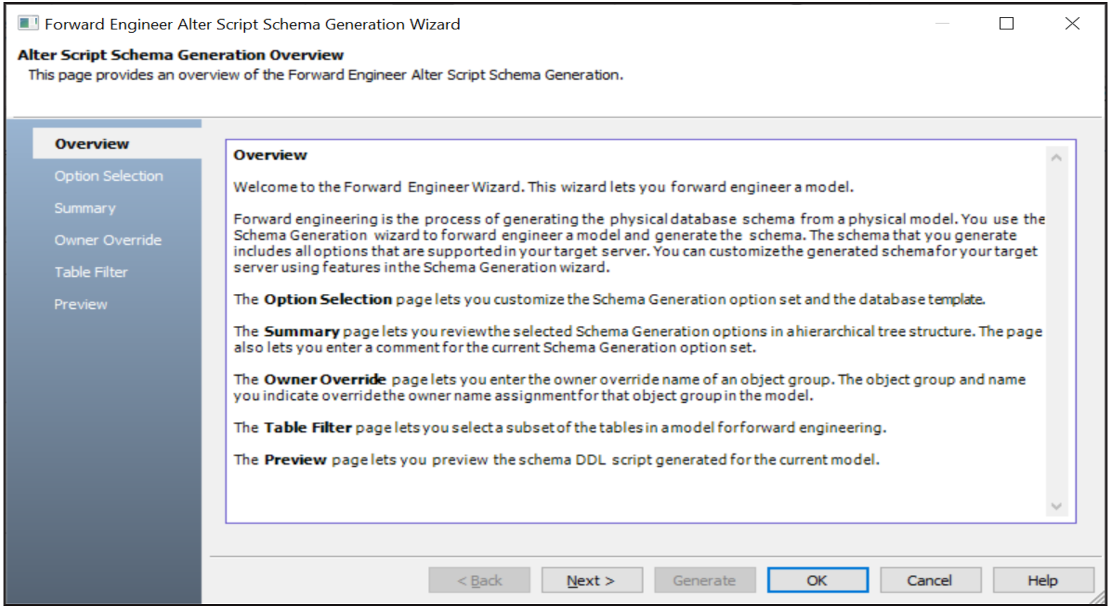
3. Make the required changes in the model.

   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.



Alter Script Schema Generation Overview

This page provides an overview of the Forward Engineer Alter Script Schema Generation.

Overview

Welcome to the Forward Engineer Wizard. This wizard lets you forward engineer a model.

Forward engineering is the process of generating the physical database schema from a physical model. You use the Schema Generation wizard to forward engineer a model and generate the schema. The schema that you generate includes all options that are supported in your target server. You can customize the generated schema for your target server using features in the Schema Generation wizard.

The **Option Selection** page lets you customize the Schema Generation option set and the database template.

The **Summary** page lets you review the selected Schema Generation options in a hierarchical tree structure. The page also lets you enter a comment for the current Schema Generation option set.

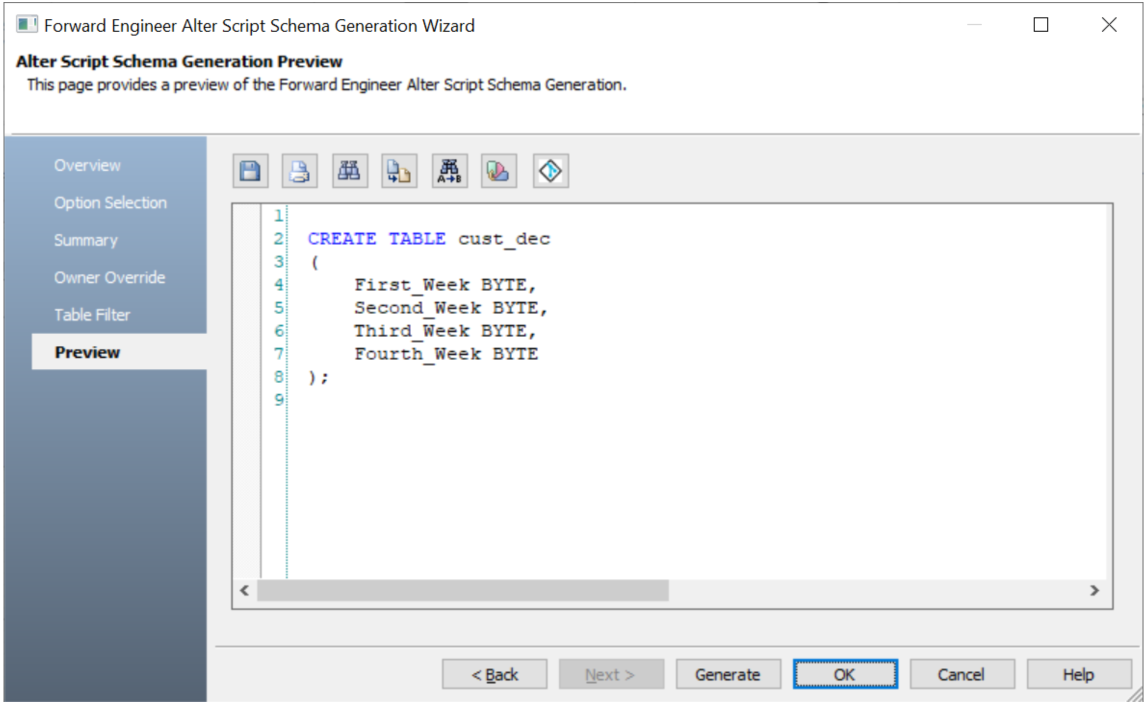
The **Owner Override** page lets you enter the owner override name of an object group. The object group and name you indicate override the owner name assignment for that object group in the model.

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

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

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

   The alter script appears. For example, in the following image the Preview section displays an alter script of a Databricks database. For more information on generating alter scripts, refer to the Generating Alter Script for Databases topic.



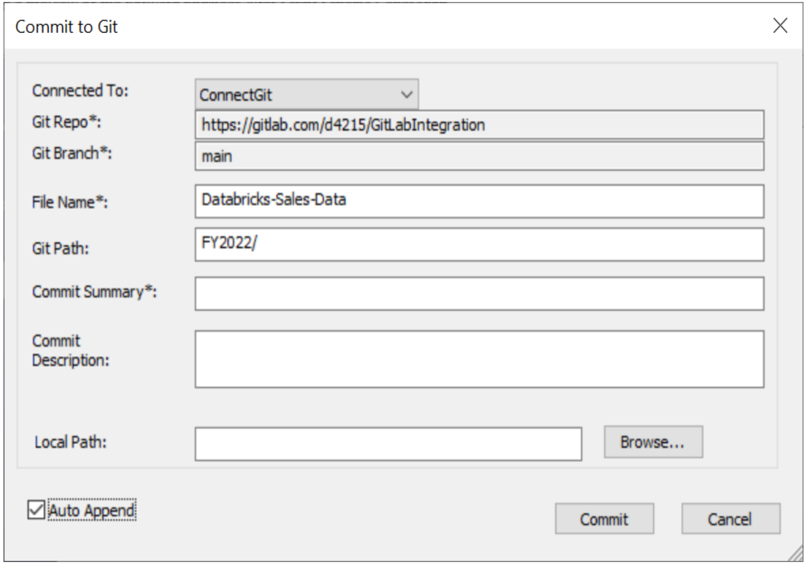
Alter Script Schema Generation Preview

This page provides a preview of the Forward Engineer Alter Script Schema Generation.

```

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

```

6. Click ◈.

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


Commit to Git

Connected To:	ConnectGit
Git Repo*:	https://gitlab.com/d4215/GitLabIntegration
Git Branch*:	main
File Name*:	Databricks-Sales-Data
Git Path:	FY2022/
Commit Summary*:	
Commit Description:	
Local Path:	

☑ Auto Append

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

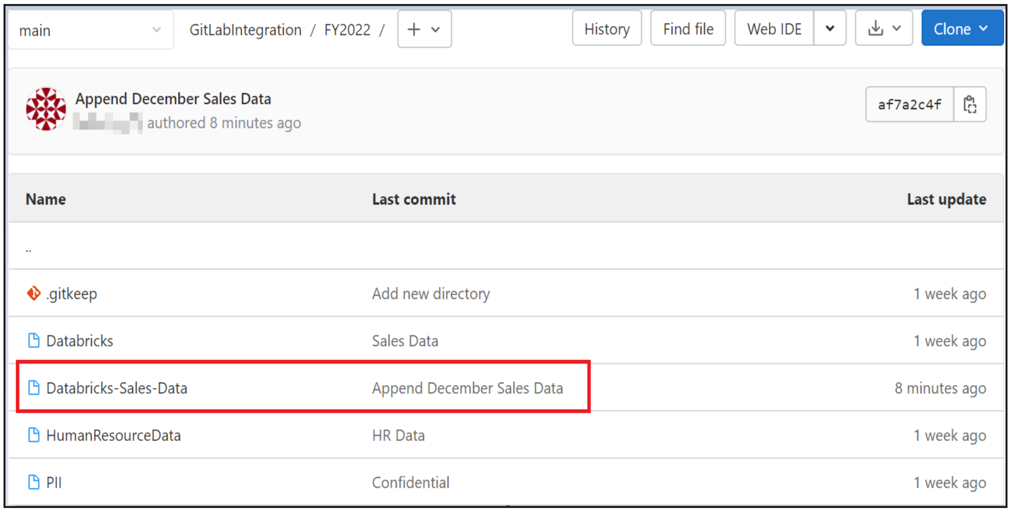
Field Name	Description	Additional Information
Connected To	Specifies the connection that connects erwin DM to a Git repository	For example, ConnectGit.
Git Repo	Specifies the Git repository that was set for the connection in the Git Connection Manager	For example, https://-gitlab.com/d4215/GitLabIntegration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

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.
Commit Description	Specifies the description of the push commit	For example, Forward schema script includes December sales data.
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.

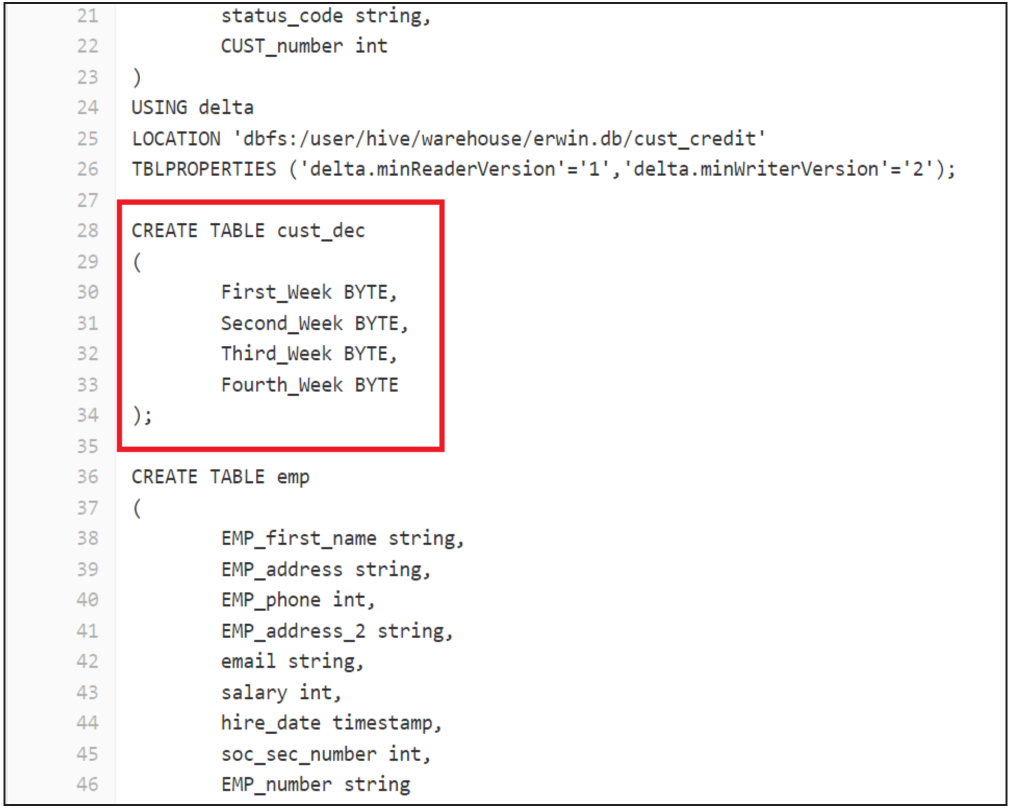


Append December Sales Data
[redacted] authored 8 minutes ago

af7a2c4f

Name	Last commit	Last update
..		
.gitkeep	Add new directory	1 week ago
Databricks	Sales Data	1 week ago
Databricks-Sales-Data	Append December Sales Data	8 minutes ago
HumanResourceData	HR Data	1 week ago
PII	Confidential	1 week ago

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

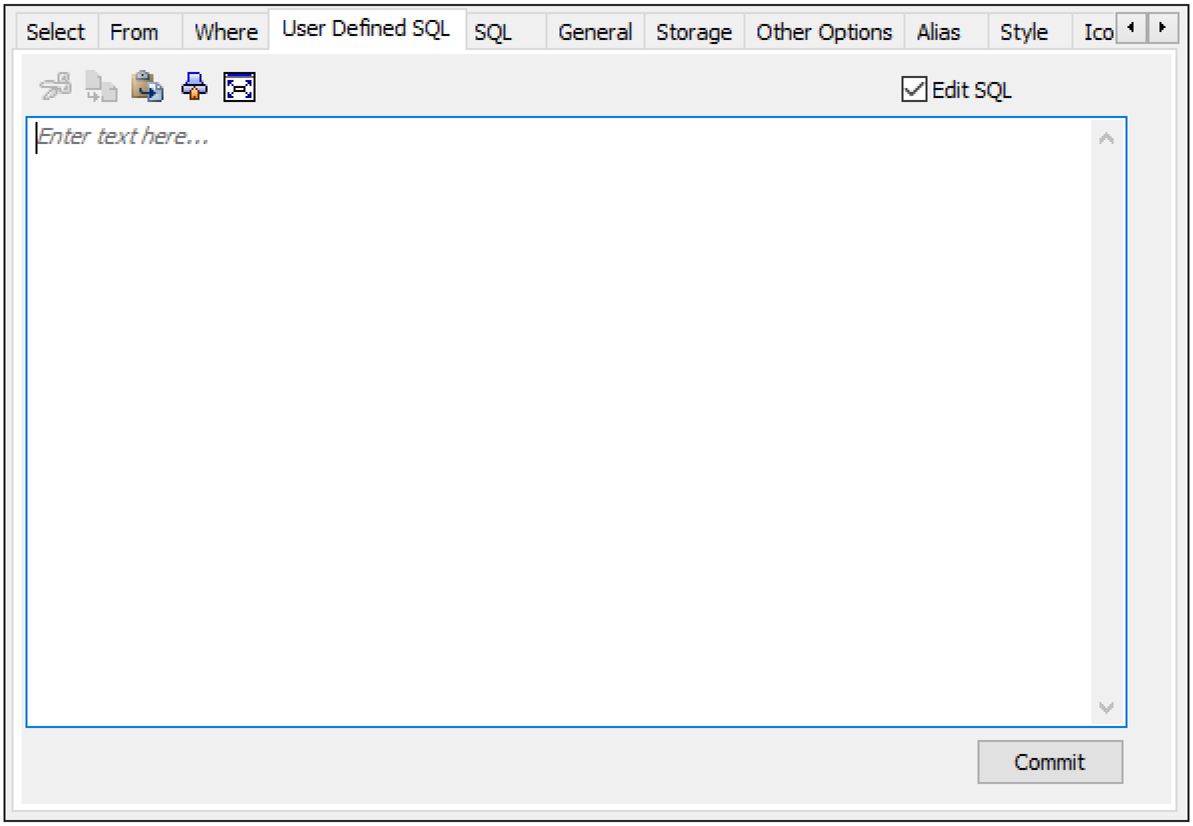


```
        status_code string,
        CUST_number int
)
USING delta
LOCATION 'dbfs:/user/hive/warehouse/erwin.db/cust_credit'
TBLPROPERTIES ('delta.minReaderVersion'='1','delta.minWriterVersion'='2');

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
```

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).

Snowflake Enhancements

erwin Data Modeler (DM) 12.0 brings the following enhancements to the Snowflake database:

- Object Tagging
- Reverse Engineering Enhancements
- Key-Pair Authentication

Object Tagging

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

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

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

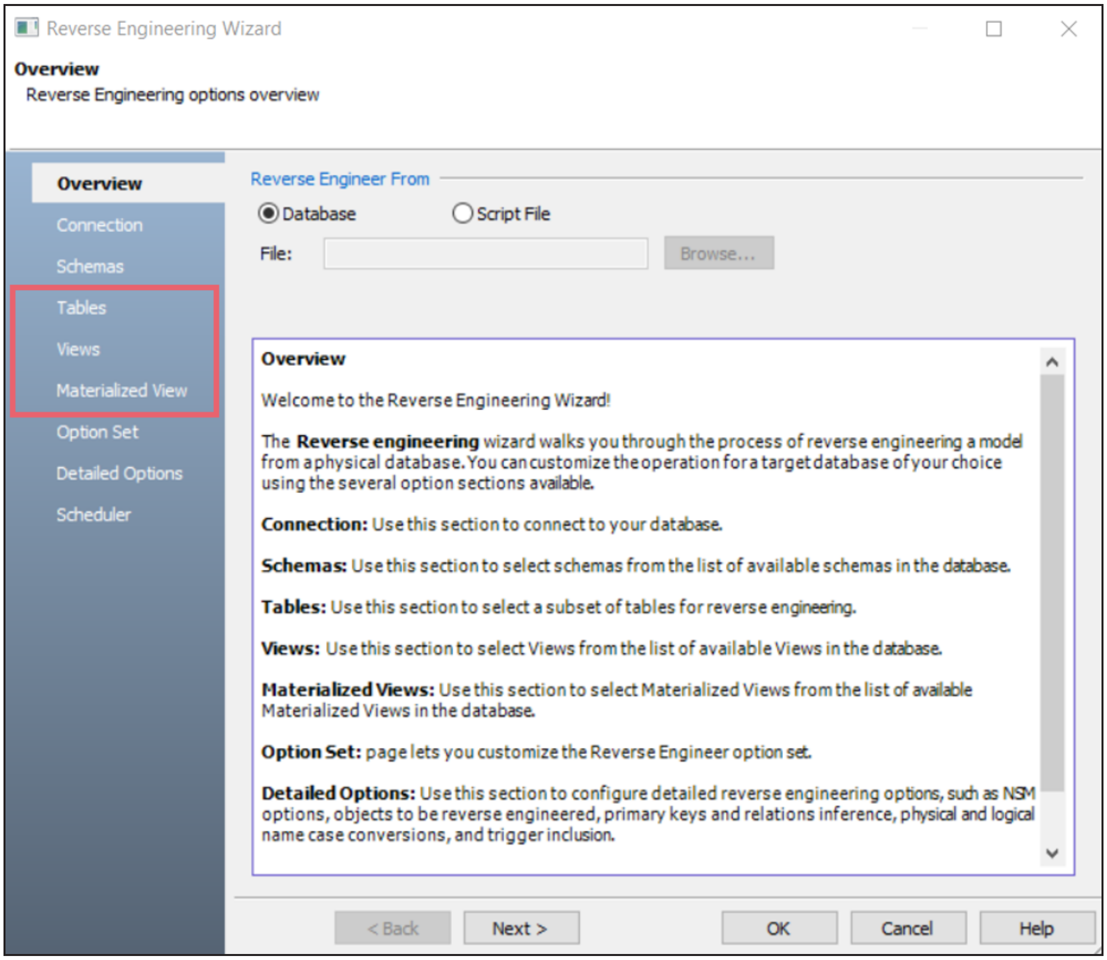
Reverse Engineering Enhancements

Reverse Engineering Wizard now adds filters for objects such as Tables, Views, and Materialized Views in different sections. These sections display all the objects in a schema, without selecting a schema during a JDBC connection.

For more information about reverse engineering a database, refer to the Selecting the Reverse Engineering Options topic.

To view filters for Tables, Views, and Materialized Views, do the following:

1. On the application ribbon, click **Action** > **Reverse Engineer**.
2. In the New Model screen, select **Snowflake** as the target server, click **Next**.
   The Reverse Engineering Wizard appears. The wizard now adds filter sections for Tables, Views, and Materialized Views objects. For more information on reverse engineering options, refer to the Reverse Engineering Options for Snowflake topic.



Overview

Welcome to the Reverse Engineering Wizard!

The **Reverse engineering** wizard walks you through the process of reverse engineering a model from a physical database. You can customize the operation for a target database of your choice using the several option sections available.

Connection: Use this section to connect to your database.

Schemas: Use this section to select schemas from the list of available schemas in the database.

Tables: Use this section to select a subset of tables for reverse engineering.

Views: Use this section to select Views from the list of available Views in the database.

Materialized Views: Use this section to select Materialized Views from the list of available Materialized Views in the database.

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

Detailed Options: Use this section to configure detailed reverse engineering options, such as NSM options, objects to be reverse engineered, primary keys and relations inference, physical and logical name case conversions, and trigger inclusion.

Key-Pair Authentication

Snowflake database supports user based key-pair authentication for reverse engineering and forward engineering models. You need a private key, public key, and password to successfully connect to a database using the key pair authentication method.

You can use either of the following methods to authenticate the database connection:

- Authenticate using unencrypted key
- Authenticate using an encrypted key

> To generate a private and public key, ensure that you installed OpenSSL 64 bit version and the set environment variables path in your machine.

Authentication using Unencrypted Key

To set up a database connection using an unencrypted key, you need to generate a private key and a public key. Then, run the script in the Snowflake console to establish a successful connection.

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

1. Open Command Prompt, and execute the following command to generate a private key:

   ```
   openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -out
   <private key>.p8 -nocrypt
   ```

   In the above statement replace the <private key> with a relevant file name of the key as shown in the below example:
   openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -out ***rsa_key.p8*** *-nocrypt*

   This generates a private key file (.p8 file format) in the specific folder location.

   > Ensure that the .p8 file is stored in a folder and specify the file path in the **Private Key File (For Key-Pair)** field while you establish connection during reverse engineering or forward engineering. Specify the file path along with the file name as shown in the example below.
   > *C:\Users\<User>\Documents\Keys\rsa_key.p8*

2. Execute the following command to generate a public key using private key:

   ```
   openssl rsa -in <private key>.p8 -pubout -out <public
   key>.pub
   ```

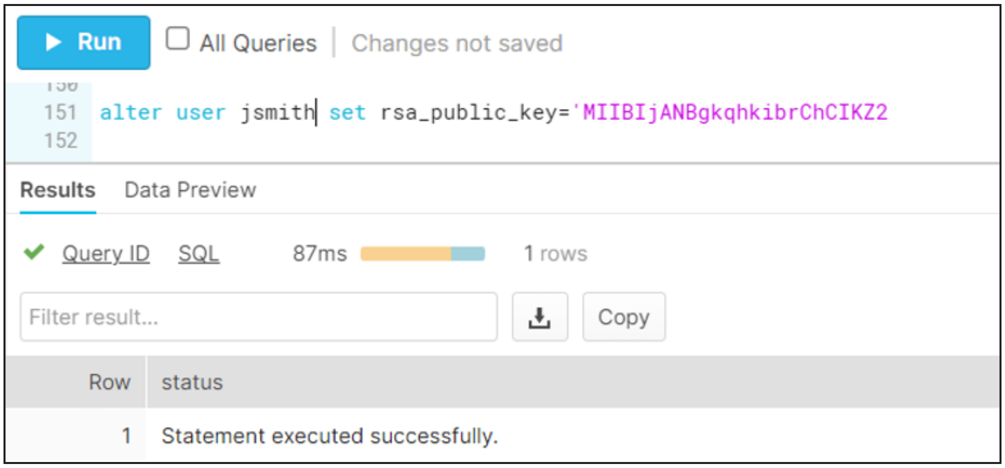
In the above statement replace the <private key> and <public key> with a relevant file name as shown in the below example:
*openssl rsa -in **rsa_key.p8** -pubout -out **rsa_key.pub***
This generates the public key file (.pub file format) in the specific folder location.

3. Open the public key file in a notepad and copy the key.
4. Go to Snowflake database console, enter the following command with a user name and a public key (in the following format):

   ```
   alter user <user name> set rsa_public_key='<public key>'
   ```

   In the above statement replace the <server name> and <public key> with relevant parameters as shown in the below example:
   *alter user **jsmith** set ras_public_key='MIIBIjANBgkqhk....'*

5. Click **Run**.
   The statement is executed and the status is displayed.



```
alter user jsmith set rsa_public_key='MIIBIjANBgkqhkibrChCIKZ2
```

Row	status
1	Statement executed successfully.

Once the keys are generated, you can enter other database connection parameters to establish a successful connection in erwin DM. For more information about database connection parameters, refer to the Database Connection Parameters topic.

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. 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. Open Command Prompt, and execute the following command to generate a private key:

   ```
   openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -out
   <private key>.p8
   ```

   In the above statement replace the <private key> with a relevant file name of the key as shown in the below example:
   openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -out ***rsa_key.p8***

   This generates a private key file (.p8 file format) in the specific folder location.

2. Enter a encryption password and verify it.

3. Execute the following command to generate a second private key using the private key in step 1:

   ```
   openssl pkcs8 -topk8 -inform PEM -v1 PBE-MD5-DES -in <private
   key>.p8 -out <private key 2>.p8
   ```

   In the above statement replace the <private key> and <private key 2> with relevant file name of the key as shown in the below example:
   openssl pkcs8 -topk8 -inform PEM -v1 PBE-MD5-DES -in ***rsa_key.p8*** *-out <private key* ***rsa_key2.p8***
   This generates a public key file (.pub file format) in the specific folder location.

   > Ensure that, this .p8 file is stored in a folder and specify the file path in the **Private Key File (For Key-Pair)** field while you establish connection during reverse engineering or forward engineering. Specify the file path along with the file name as shown in the example below.
   > *C:\Users\<User>\Documents\Keys\rsa_key.p8*

4. Enter a pass phrase for the private key.

5. Enter the encryption key and verify it.

6. Execute the following command to generate a public key using the private key in step 3:

   ```
   openssl rsa -in <private key 2>.p8 -pubout -out <public
   key>.pub
   ```

   In the above statement replace the <private key 2> and <public key 2> with a relevant file name of the key as shown in the below example:
   *openssl rsa -in **rsa_key2**.p8 -pubout -out **rsa_key2**.pub*
   This generates the public key file (.pub file format) in the specific folder location.

7. Enter the pass phrase of the second private key in step 4.

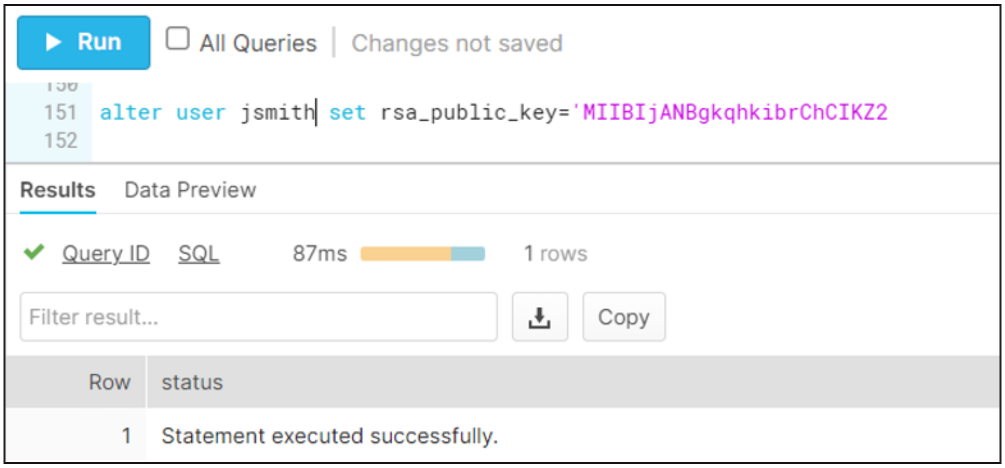
8. Open the second public key file in a notepad and copy the key.

9. Go to Snowflake database console, enter the following command with a user name and a public key (in the following format):

   ```
   alter user <user name> set rsa_public_key='<public key>'
   ```

   In the above statement replace the <server name> and <public key> with relevant parameters as shown in the below example:
   *alter user **jsmith** set ras_public_key='MIIBIjANBgkqhk....'*

10. Click **Run**.
    The statement is executed and the status is displayed.



```
alter user jsmith set rsa_public_key='MIIBIjANBgkqhkibrChCIKZ2
```

Row	status
1	Statement executed successfully.

Once the keys are generated, you can enter other database connection parameters to establish a successful connection. For more information about database connection parameters, refer to the Database Connection Parameters topic.

MongoDB: Schema Validation

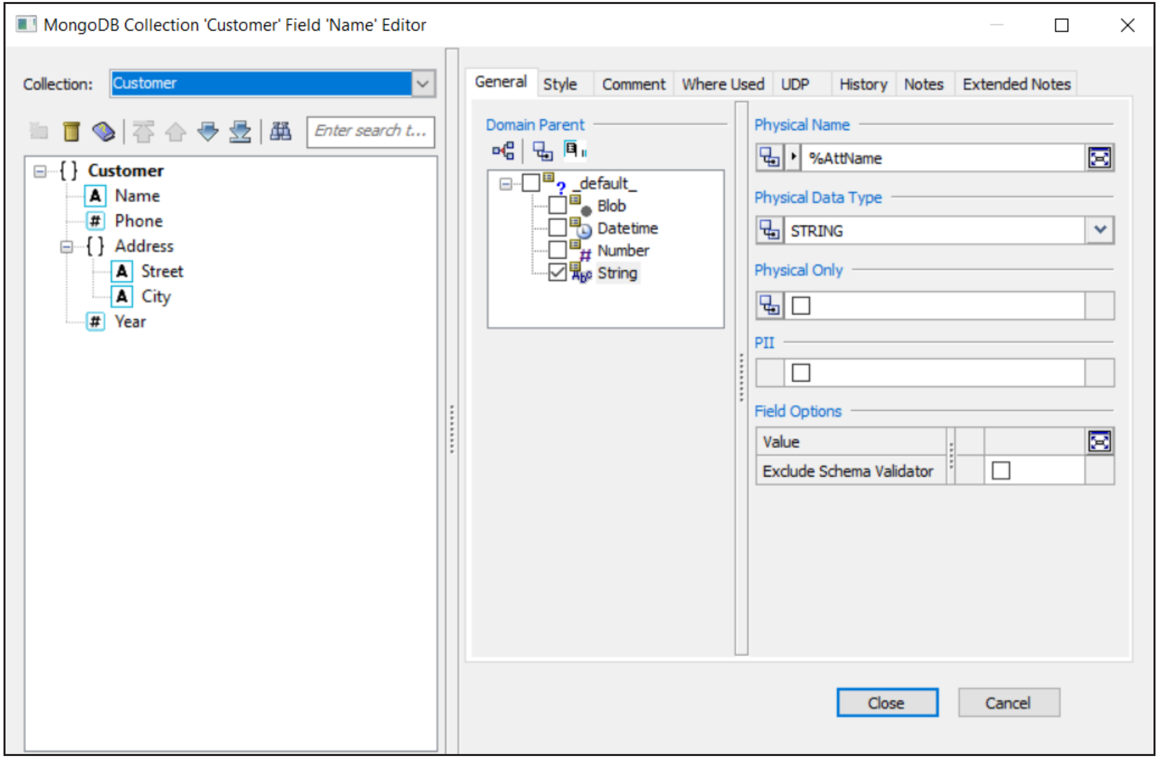
erwin Data Modeler 12.0 now supports schema validation for fields in a collection based on the field's data type. The scripts are not generated if the data in the fields are not inline with the assigned data type. If the validations are not met, application displays an error.

Also, you can exclude validations for one or more fields in a table when you generate a script using forward engineering and reverse engineering. This option overrides assigned data type of a field and generates script successfully.

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

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

The below screenshot displays the fields in a collection along with the assigned data types in the field property editor screen.


MongoDB Collection 'Customer' Field 'Name' Editor

Collection: Customer

- { } Customer
  - A Name
  - # Phone
  - { } Address
    - A Street
    - A City
  - # Year

General | Style | Comment | Where Used | UDP | History | Notes | Extended Notes

Domain Parent

- □ _default_
  - □ Blob
  - □ Datetime
  - □ Number
  - ☑ String

Physical Name: %AttName

Physical Data Type: STRING

Physical Only: □

PII: □

Field Options

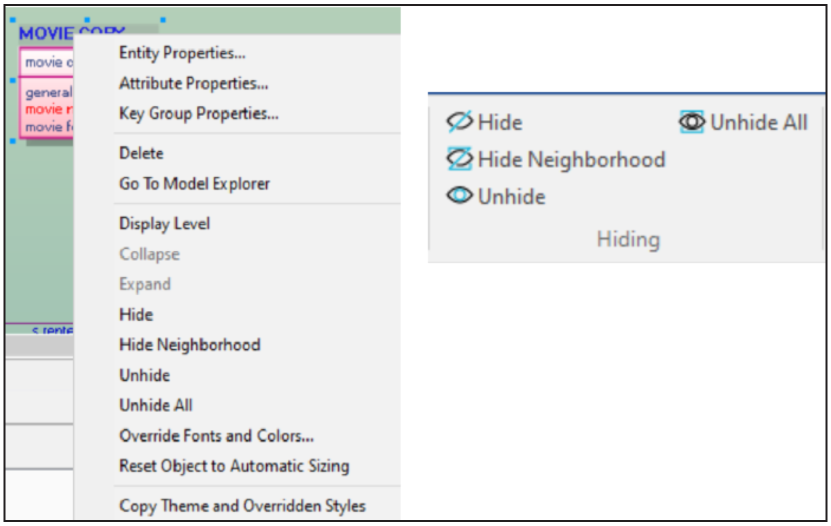
Value	
Exclude Schema Validator	□

To validate The below screenshot displays the fields in a collection along with the assigned data types in the field property editor screen.

This topic walks you through the steps to create associations between environments, business terms, and tables. Then, use the environment as a unique qualifier for association using an example.

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 layout. 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.

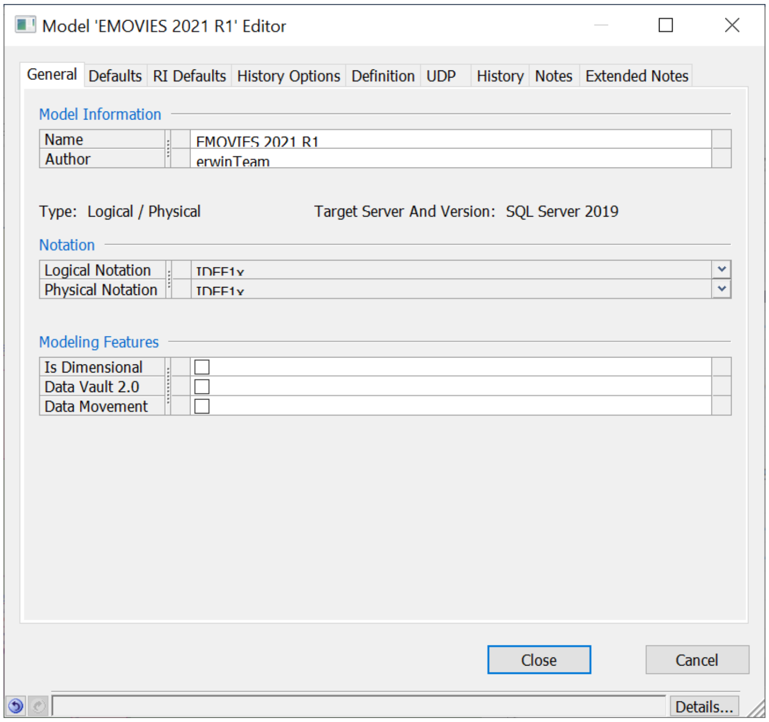
> You can also access these options by right-clicking the nodes.

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.



Name	EMOVIES 2021 R1
Author	erwinTeam

Type: Logical / Physical Target Server And Version: SQL Server 2019

Logical Notation	IDEF1x
Physical Notation	IDEF1x

Is Dimensional	□
Data Vault 2.0	□
Data Movement	□

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

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.

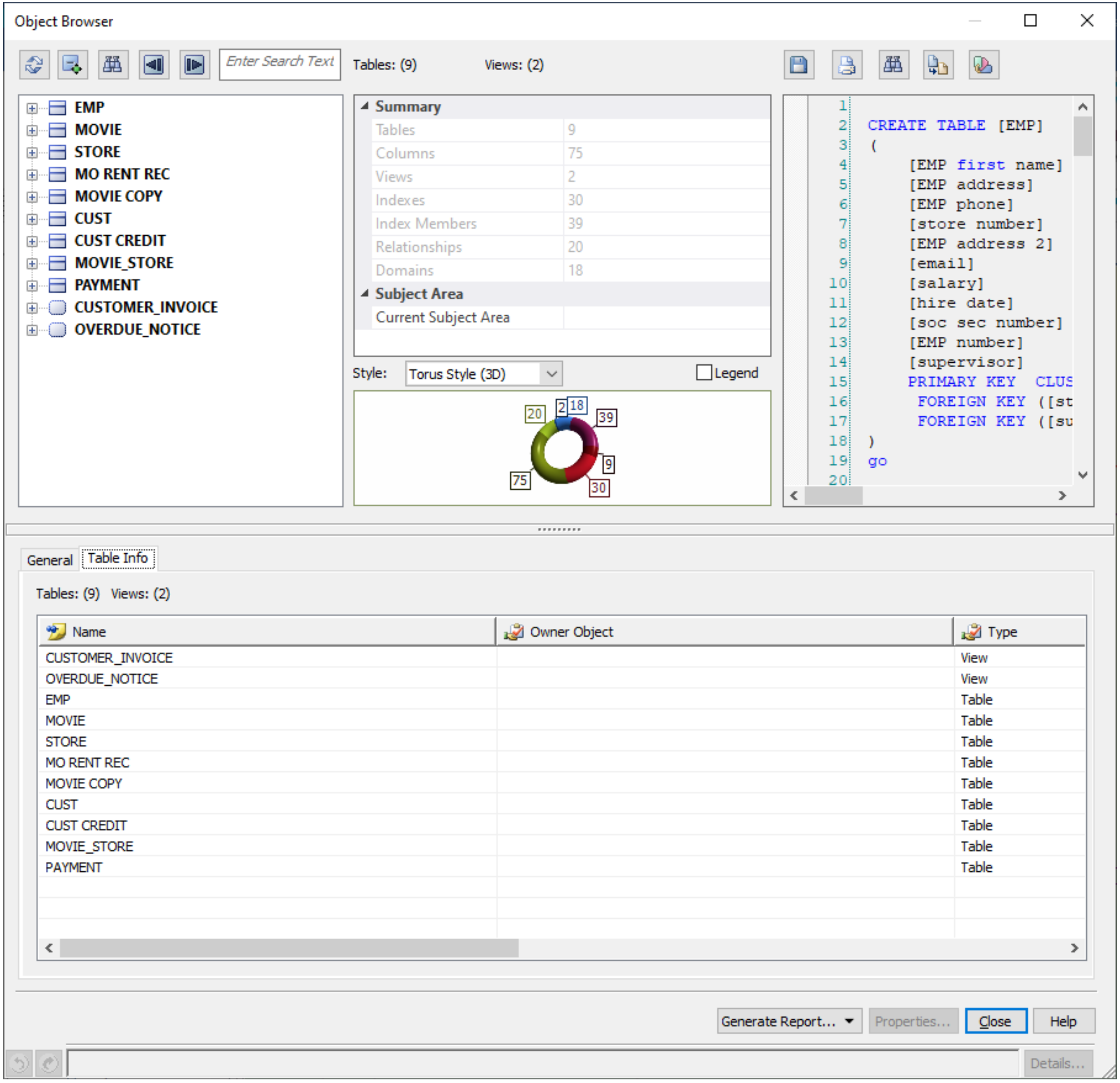
   If none of the neighboring objects exist in the source diagram, the selected object is copied and can be added to new model.
   If one or more selected objects exist in the target diagram, a message appears informing you that the objects are pasted as new objects in the model. The new objects follow 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.



Object Browser

Enter Search Text

Tables: (9) Views: (2)

- EMP
- MOVIE
- STORE
- MO RENT REC
- MOVIE COPY
- CUST
- CUST CREDIT
- MOVIE_STORE
- PAYMENT
- CUSTOMER_INVOICE
- OVERDUE_NOTICE

Summary	
Tables	9
Columns	75
Views	2
Indexes	30
Index Members	39
Relationships	20
Domains	18
Subject Area	
Current Subject Area	

Style: Torus Style (3D) □ Legend

```
CREATE TABLE [EMP]
(
      [EMP first name]
      [EMP address]
      [EMP phone]
      [store number]
      [EMP address 2]
      [email]
      [salary]
      [hire date]
      [soc sec number]
      [EMP number]
      [supervisor]
      PRIMARY KEY  CLUS
       FOREIGN KEY ([st
       FOREIGN KEY ([su
)
go
```

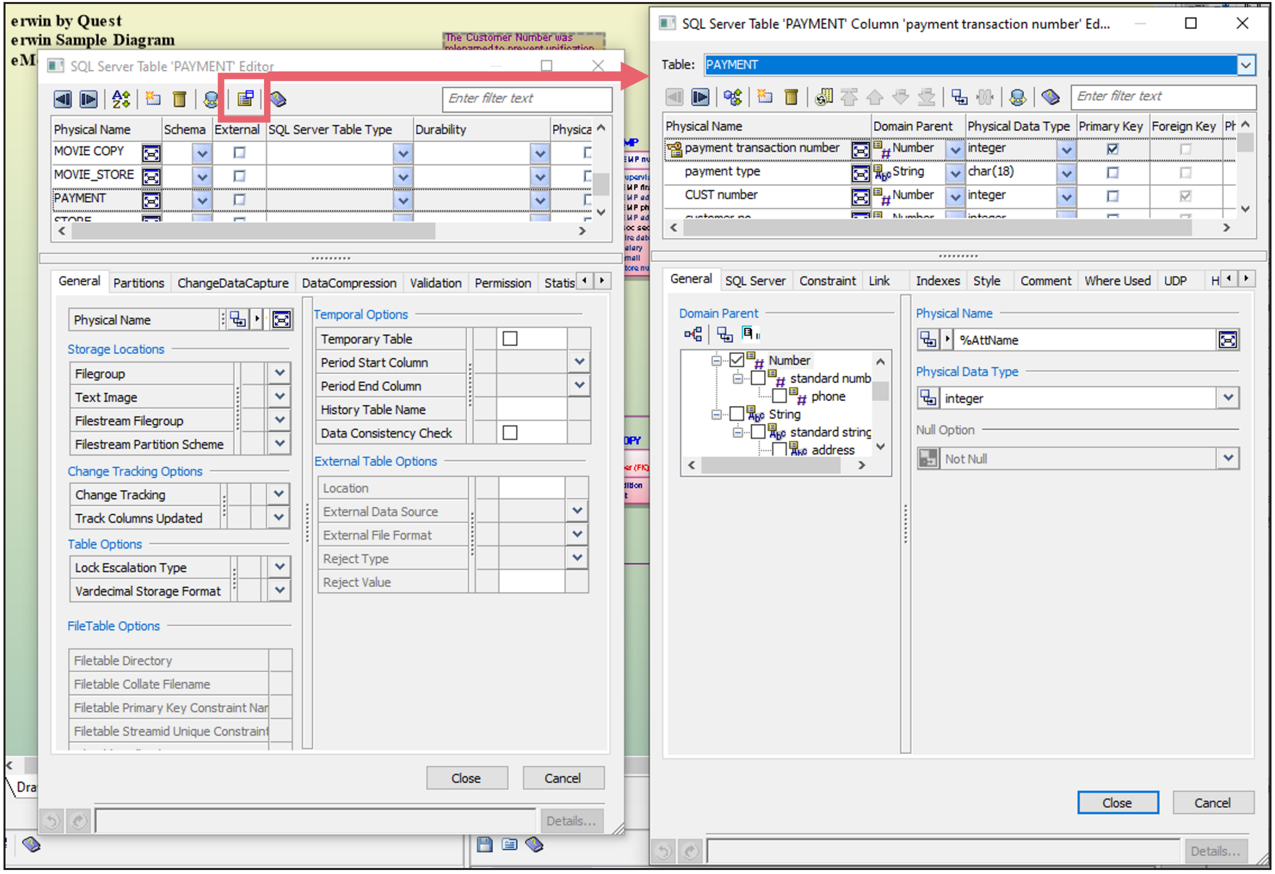
General | Table Info

Tables: (9) Views: (2)

Name	Owner Object	Type
CUSTOMER_INVOICE		View
OVERDUE_NOTICE		View
EMP		Table
MOVIE		Table
STORE		Table
MO RENT REC		Table
MOVIE COPY		Table
CUST		Table
CUST CREDIT		Table
MOVIE_STORE		Table
PAYMENT		Table

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.



erwin by Quest
erwin Sample Diagram

SQL Server Table 'PAYMENT' Editor

Physical Name	Schema	External	SQL Server Table Type	Durability
MOVIE COPY		□		
MOVIE_STORE		□		
PAYMENT		□		

SQL Server Table 'PAYMENT' Column 'payment transaction number' Ed...

Table: PAYMENT

Physical Name	Domain Parent	Physical Data Type	Primary Key	Foreign Key
payment transaction number	Number	integer	☑	□
payment type	String	char(18)	□	□
CUST number	Number	integer	□	☑

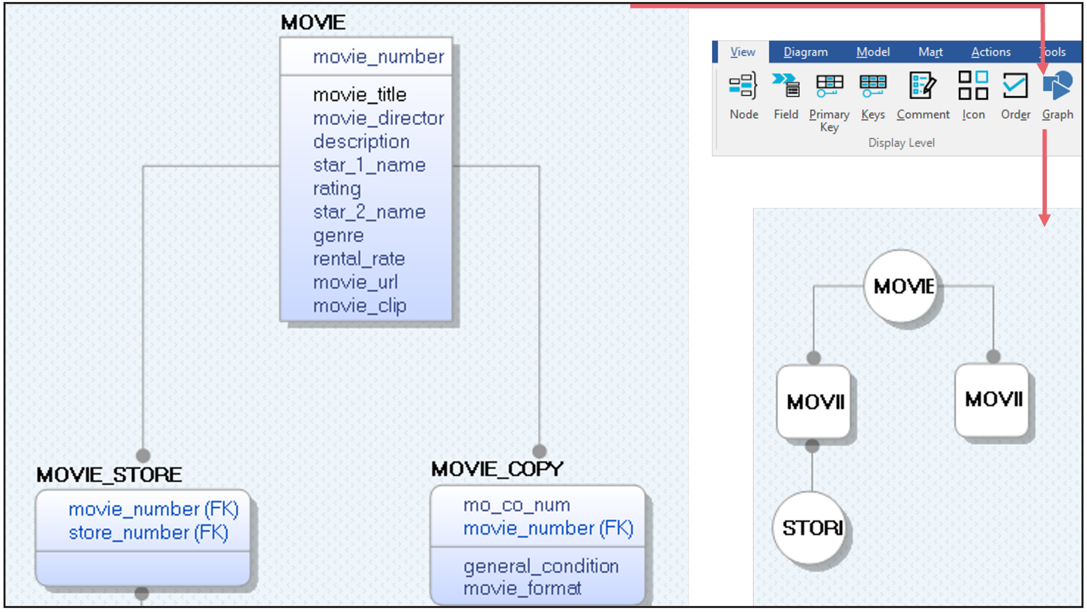
Physical Name: %AttName

Physical Data Type: integer

Null Option: Not Null

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:



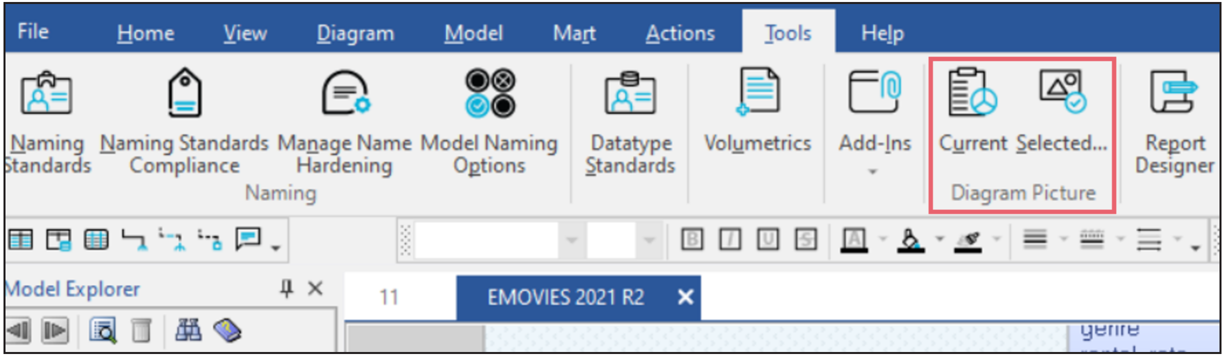
Generate Diagram Picture in Multiple Formats

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:

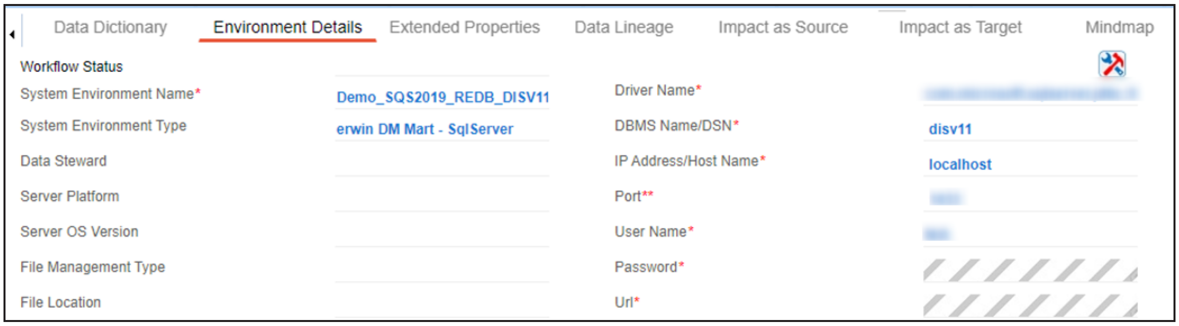
- Click **Current** to generate a single picture diagram of you current model. For more information, refer to the Generate Current Diagram Picture topic.
- Click **Selected** to generate multiple diagrams based on your selection. For more information, refer to the Generate Multiple Diagram Pictures topic.


File
Home
View
Diagram
Model
Mart
Actions
Tools
Help
Naming Standards
Naming Standards Compliance
Manage Name Hardening
Model Naming Options
Naming
Datatype Standards
Volumetrics
Add-Ins
Current
Selected...
Diagram Picture
Report Designer
Model Explorer
11
EMOVIES 2021 R2
genre

DM Connect for DI

DM Connect for DI has undergone several enhancements as follows:

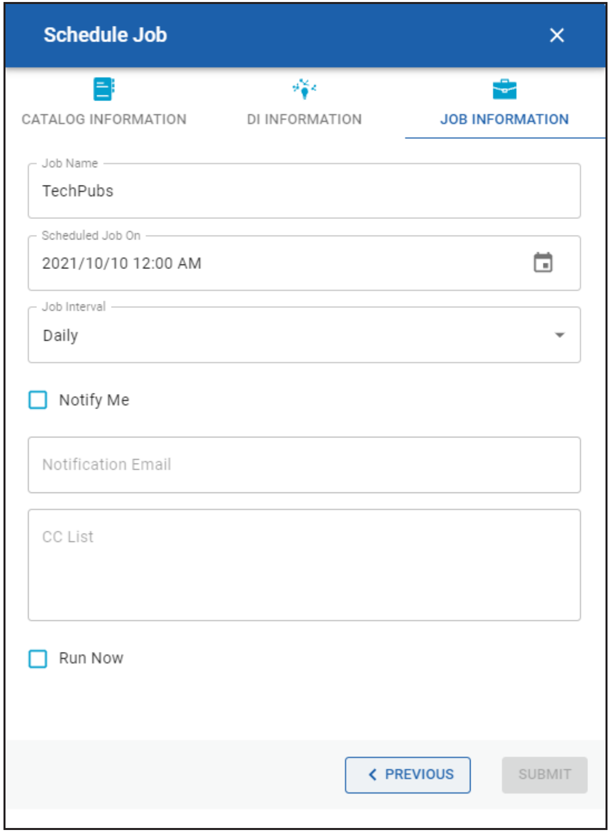
- The REDB process now stores the database connection parameters, such as DBMS Name/DSN, IP Address/Host Name, and Port under Environment Details in erwin Data Intelligence Suite.



Workflow Status

System Environment Name*	Demo_SQS2019_REDB_DISV11	Driver Name*	[redacted]
System Environment Type	erwin DM Mart - SqlServer	DBMS Name/DSN*	disv11
Data Steward		IP Address/Host Name*	localhost
Server Platform		Port**	[redacted]
Server OS Version		User Name*	[redacted]
File Management Type		Password*	[redacted]
File Location		Url*	[redacted]

- You can now run jobs immediately using the Run Now feature.



Schedule Job

Job Name: TechPubs

Scheduled Job On: 2021/10/10 12:00 AM

Job Interval: Daily

□ Notify Me

Notification Email

CC List

□ Run Now

- 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.

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